Revista Facultad de Ingeniería No. 24. pp. 121-131. Diciembre, 2001

Fórmulas de direccionamiento en matrices triangulares

Roberto Flórez Rueda y Francisco J. Moreno**

Resumen

Las matrices dispersas —matrices que poseen muchos de sus elementos con valor cero— suelen representarse en vectores con el objetivo de ahorrar espacio. Al realizar tal transformación se debe garantizar que los algoritmos desarrollados para operar con ellas ofrezcan un buen rendimiento.

Se presenta a continuación un estudio de cuatro tipos de matrices dispersas triangulares, las cuales aparecen con bastante frecuencia en la práctica. Los algoritmos y las representaciones se aplican también a matrices simétricas [1].

Se incluyen los análisis y algoritmos para lograr las transformaciones deseadas, se analiza la eficiencia de cada uno de ellos y la forma como pueden mejorarse gradualmente hasta obtener algoritmos con orden de magnitud constante.

---------- *Palabras clave*: matrices triangulares, algoritmos, transformación, vectores.

Formulae for directing in triangular matrices

Abstract

Dispersed matrices —those that have many of their elements with value zero— usually are represented as vectors with the objective to save space. When making such transformation it has to be guaranteed that the developed algorithms to operate such matrices offer a good yield.

A study was done of four types of triangular dispersed matrices, which appear with enough frequency. The algorithms and the representations are applicable also to symmetrical matrices [1]. The analysis and algorithms are included to obtain the wanted transformations, to analyze the efficiency of each one of them and the way as they can be improved gradually until obtaining algorithms with constant order of magnitude.

---------- Key words: Triangular matrices, algoritms, transformation, vectors.

* Departamento de Ingeniería de Sistemas. Facultad de Ingeniería. Universidad de Antioquia.
rflorez@udea.edu.co.

Introducción

Las matrices triangulares aparecen con frecuencia en la práctica, especialmente en la solución de sistemas de ecuaciones en el álgebra [1] y la programación lineal [2]. Dichas matrices se consideran dispersas ya que más del 40% de sus elementos son cero. Así, por ejemplo, una matriz triangular de dimensión 50 x 50 tendrá al menos 1.225 elementos nulos. Se prefiere representar estas matrices en vectores, llevando sus elementos, ordenados ascendentemente por filas y dentro de la fila, por columnas, a posiciones fijas en un vector; se representan únicamente aquellos elementos diferentes de cero para lograr, así, un ahorro de memoria. Un ejemplo se observa en las figuras 1 y 2.

Dos problemas se deben resolver para lograr la correcta manipulación de dichas matrices.

- Dado un elemento de la matriz, el cual se identifica por fila y columna, hallar la posición que le corresponde en el vector.
- Dada una posición en el vector, determinar cuáles son las coordenadas del elemento de la matriz que allí se halla representado.

Las soluciones a estos dos problemas deben ser eficientes, pues de no ser así se degradaría significativamente el desempeño de los algoritmos correspondientes a las operaciones típicas con matrices.

Se analizan a continuación cuatro matrices triangulares: triangular inferior izquierda, triangular inferior derecha, triangular superior derecha y triangular superior izquierda (véanse figuras 1, 3, 5 y 6) y para cada una de ellas se expone la forma de obtener las soluciones a los dos problemas mencionados.

Matriz triangular inferior izquierda (*mtii*)

En una *mtii* el número de elementos diferentes de cero es: uno en la fila uno, más dos en la fila dos, más tres en la fila tres y así sucesivamente; se obtiene mediante:

$$\sum_{i=0}^{i=n} i = n(n+1)/2 \qquad (1)$$

donde n el orden de la matriz.

Esa fórmula es válida para cualquiera de las cuatro matrices triangulares descritas.

1	*2*	*3*	*4*	*5*	*6*	*7*
1						
2	3					
4	5	6				
7	8	9	10			
11	12	13	14	15		
16	17	18	19	20	21	
22	23	24	25	26	27	28

Figura 1 Matriz *mtii*

Sean i y j fila y columna de *mtii* y *pos* la posición del vector en la cual quedará almacenado el elemento de la fila i y columna j de *mtii*.

En todas las matrices analizadas se cumplen: $1 \le i \le n$ y $1 \le j \le n$

Para encontrar una fórmula que permita conocer *pos* para un elemento de la fila i y la columna j se utilizará el método de la inducción matemática.

El análisis correspondiente se presenta en la tabla 1.

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28

Figura 2 Vector con el cual se representa *mtii*

Tabla 1 Análisis para determinar pos en *mtii*

i	*j*	*(Los almacenados antes) +*	*?*	*pos*
1	1	0	1	1
2	1	1	1	2
2	2	1	2	3
3	1	3	1	4
3	2	3	2	5
3	3	3	3	6
4	1	6	1	7
4	2	6	2	8
4	3	6	3	9
4	4	6	4	10
5	1	10	1	11
5	2	10	2	12
5	3	10	3	13
5	4	10	4	14
5	5	10	5	15
6	1	15	1	16
6	2	15	2	17
6	3	15	3	18
6	4	15	4	19
6	5	15	5	20
6	6	15	6	21
7	1	21	1	22
7	2	21	2	23
7	3	21	3	24
7	4	21	4	25
7	5	21	5	26
7	6	21	6	27
7	7	21	7	28

Para un elemento perteneciente a la fila i y columna j se observa que su posición *pos* en el vector se obtiene con base en la suma de los elementos de las $(i - 1)$ filas anteriores más la columna j, o sea:

$$pos = \sum_{k=1}^{i-1}(k) + j$$

Sumatoria cuyo valor es $i(i - 1)/2$, según (1)

Por consiguiente:

$$pos = i(i - 1)/2 + j \quad (2)$$

El mismo resultado fue expuesto en [3] y [4].

Cuando se utilizan esas fórmulas se debe especificar para cuáles valores de i y j son válidas.

El (2) será válida sólo para aquellos elementos que pertenezcan a la matriz triangular inferior, es decir:

$$\forall i, j / i \geq j$$

Una aplicación del uso de (2) se puede observar en el algoritmo de multiplicación de matrices, en el que las matrices se representan en vectores de acuerdo con la figura 2 y se deja el resultado en un tercer vector que cumple la misma representación.

El algoritmo de la tabla 2 tiene orden de magnitud cúbica, $O(n^3)$, el cual es un orden típico para la multiplicación de matrices [6].

Tabla 2 Algoritmo que multiplica dos matrices triangulares inferiores, representadas en vectores, haciendo uso de (2)

```
//A, B: vectores de entrada que representan matrices
   por multiplicar.
//n: Orden de las matrices por multiplicar.
//C: vector de salida que representa la matriz resultado.
for i = 1 to n do
   k = i(i - 1)/2
   C(k) = 0
   for j = 1 to i do
      for m = j to i do
         C(k + j) = C(k + j) + (A(k + m))(B(m(m - 1)/2 + j))
      end(for)
   end(for)
end(for)
```

Ahora se considerará el proceso inverso; es decir, conocido *pos*, n y usando (2) interesa determinar

la fila y columna representadas en dicha posición (véanse tablas 3 y 4).

Para ello se puede proceder de diversas formas:

Algoritmos cuadráticos (dos versiones)

Tabla 3 Algoritmo, con orden de magnitud cuadrática, para determinar fila y columna conocidos *pos*, *n* y (2) en *mtii* (versión 1)

```
// pos y n son parámetros de entrada
// fila y columna son parámetros de retorno
   for k=1 to n
      for i = 1 to k
         if (k(k - 1)/2 + i) = pos
            fila = k
            columna = i
            return
      end(if)
   end(for)
```

Tabla 4 Algoritmo, con orden de magnitud cuadrática, para determinar fila y columna conocidos *pos*, *n* y (2) en *mtii* (versión 2)

```
//pos y n son parámetros de entrada
//fila y columna parámetros de retorno
aux = 1
for k = 1 to n
   for i = 1 to k
    if aux = pos then
      fila = k
      columna = I
      return
   end(if)
   aux = aux + 1
   end(for)
end(for)
```

En el peor de los casos el orden de magnitud de los algoritmos de las tablas 3 y 4 es $O(n^2)$, ya que cuando *pos* sea el último elemento de la matriz se tendrán que ejecutar los ciclos completamente.

Algoritmo lineal

Es una mejora del algoritmo anterior, el cual se ha denominado lineal. Véase tabla 5.

Tabla 5 Algoritmo, con orden de magnitud lineal, para determinar fila y columna, conocidos *pos*, *n* y (2) en una *mtii*

```
//pos es parámetro de entrada
//fila y columna son parámetros de retorno
k = 1
while (k(k + 1)/2 ) < pos do
   k = k+1
end(while)
fila = k
columna = pos - k(k - 1)/2
```

El orden de magnitud del algoritmo de la tabla 5 es $O(n)$, ya que en el peor de los casos, cuando *pos* está en la última fila, la *k* se incrementa $(n - 1)$ veces.

Algoritmo logarítmico

Se puede mejorar, haciendo uso de la idea de la búsqueda binaria [5], logrando un algoritmo con orden de magnitud $O(\log_2 n)$.

Dicho algoritmo se presenta en la tabla 6.

Tabla 6 Algoritmo, con orden de magnitud logarítmico, para determinar fila y columna conocidos *pos*, *n* y (2) en *mtii*

```
//pos y n son parámetros de entrada.
//fila y columna son parámetros de salida.
ki = 1
kf = n
while ki < = kf do
```

Tabla 6 (continuación)

```
    km = ( ki +kf )/2
    vf = km(km +1)/2
    vi = vf – km
    casos
       : pos > vf:
          ki = km +1
       : pos < = vi:
          kf = km – 1 :
       else:
          fila = ki
          columna = pos – fila(fila – 1)/2
          break
    fin(casos)
end(while)
```

Algoritmo constante. Pero lo ideal es obtener fila y columna sin realizar ciclos, obteniendo así un algoritmo con orden de magnitud O(1). Se propone a continuación una manera de lograrlo.[1]

De (*f*) se tiene: $pos = i(i - 1)/2 + j$

La variable *j* en dicha fórmula se utiliza para determinar la posición *pos* en el vector de una fila dada; por consiguiente, si se elimina esta variable de la fórmula de direccionamiento se estará ubicado en la fila *i*.

La expresión $i(i - 1)/2$ representa $\sum_{k=1}^{i-1} k$

Es decir, si se tiene la fórmula $pos = i(i - 1)/2$ se puede despejar *i* y obtener la fila correspondiente a una posición *pos* dada.

La variable *i* se despeja haciendo la siguiente transformación:

$2pos = i(i - 1)$

$2pos = i^2 - i$

$i^2 - i - 2pos = 0$

la cual es una expresión cuadrática, cuya solución es:

$$i = (1 + SQRT(1 + 8pos))/2 \qquad (3)$$

Los resultados obtenidos, despejando *i* con la fórmula obtenida anteriormente, se presentan en la tabla 7, columna 2, para una *mtii* de orden 7.

Tabla 7 Análisis de resultados para determinar el valor de una fila, conocidos *pos*, *n* y (2) para una *mtii* de orden 7

pos	*Valor de i (columna 2)*	*Valor de i (columna 3)*
1	2,00	1,00
2	2,56	2,00
3	3,00	2,56
4	3,37	3,00
5	3,70	3,27
6	4,00	3,70
7	4,27	4,00
8	4,53	4,27
9	4,77	4,53
10	5,00	4,77
11	5,22	5,00
12	5,42	5,22
13	5,62,	5,42
14	5,82	5,62
15	6,00	5,82
16	6,18	6,00
17	6,35	6,18
18	6,52	6,35
19	6,68	6,52
20	6,84	6,68
21	7,00	6,84
22	7,15	7,00
23	7,30	7,15
24	7,45	7,30
25	7,59	7,45
26	7,73	7,59
27	7,87	7,73
28	8,00	7,87

1 De acuerdo con [7] y hasta donde se logró establecer, este algoritmo no ha sido presentado previamente en referencia alguna.

La *i* de la columna 3 se despejó de la fórmula que a continuación se explica: dada una posición *pos* despejando la *i* y *TRUNC*ándola, se obtiene la fila correspondiente a la posición *pos*, excepto para la diagonal principal, en la cual la *i* queda ubicada una fila más allá. Para corregir esto, se resta una unidad a *pos*, antes de despejar la *i*; o sea, despejamos *i* a partir de:

$$i(i - 1)/2 = pos - 1 \qquad (3')$$

despejando *i* se obtiene

$$i = TRUNC(1 + SQRT(8pos - 7)/2) \qquad (4)$$

y la columna *j* se obtiene de (2):

$$j = pos - i(i - 1)/2$$

El algoritmo tiene orden de magnitud O(1) y se presenta en la tabla 8.

Tabla 8 Algoritmo, con orden de magnitud constante, para determinar fila y columna, conocidos *pos*, *n* y (2) para una *mtii*

```
//pos parámetro de entrada
//fila y columna parámetros de retorno.
fila = TRUNC(1 + SQRT(8pos - 7)/2)
columna = pos – fila(fila - 1)/2
```

Se presenta a continuación una tabla resumen que muestra la mejora progresiva de los algoritmos presentados (tabla 8.1).

Tabla 8.1 Resumen de los órdenes de magnitud de los algoritmos para determinar fila y columna, conocidos *pos*, *n* y (2) en una *mtii*

Tipo de algoritmo	*Orden de magnitud*
Algoritmo cuadrático	$O(n^2)$
Algoritmo lineal	$O(n)$
Algoritmo logarítmico	$O(\log_2 n)$
Algoritmo constante	O(1)

Matriz triangular superior derecha (*mtsd*) (véase figura 3)

1	*2*	*3*	*4*	*5*	*6*	*7*
1	2	3	4	5	6	7
	8	9	10	11	12	13
		14	15	16	17	18
			19	20	21	22
				23	24	25
					26	27
						28

Figura 3 Matriz *mtsd*

Con las mismas convenciones se tiene los resultados que se muestran en la tabla 9.

Tabla 9 Análisis para determinar *pos* en *mtsd*

i	*j*	*(Los almacenados antes)* +	?	*pos*
1	1	0	1	1
1	2	0	2	2
1	3	0	3	3
1	4	0	4	4
1	5	0	5	5
1	6	0	6	6
1	7	0	7	7
2	2	7	1	8
2	3	7	2	9
2	4	7	3	10
2	5	7	4	11
2	6	7	5	12
2	7	7	6	13
3	3	13	1	14
3	4	13	2	15
3	5	13	3	16
3	6	13	4	17
3	7	13	5	18
4	4	18	1	19
4	5	18	2	20
4	6	18	3	21
4	7	18	4	22
5	5	22	1	23
5	6	22	2	24
5	7	22	3	25
6	6	25	1	26
6	7	25	2	27
7	7	27	1	28

Para un elemento situado en la fila i se tiene que sumar el número de elementos presentes en las i - 1 filas anteriores así:

Si $i = 5$, se debe sumar 7 + 6 + 5 + 4

Esa sumatoria se expresa así: $\sum_{K=1}^{i-1}(n-k+1)$

Ahora se requiere determinar la componente ? de la tabla 9 para hallar la posición *pos* en el vector.

Observando la tabla 9 resulta:

Para la fila 2 columna 3 se debe sumar 2.

Para la fila 3 columna 7 se debe sumar 5.

Para la fila 4 columna 3 se debe sumar 2.

Para la fila 5 columna 7 se debe sumar 3.

El término ? se obtiene mediante: $j - i + 1$

O sea que: $\sum_{k=1}^{i-1}(n-k+1)+(j-1+1)$

la cual, desarrollando las sumatorias y factorizando se reduce a:

$$pos = (i - 1)(n - i/2) + j \qquad (6)$$

$\forall i, j / j \geq i$

Nótese que la división es real (no se *TRUNC*a ni se redondea).

Un resultado y análisis similar puede verse en [4].

Para esta representación también se requiere desarrollar el proceso inverso; es decir, conocidos *pos, n* y la (6), determinar la fila y columna correspondientes a esa posición *pos.*

Se propone a continuación un algoritmo con orden de magnitud constante para lograrlo.[2]

Partiendo de $\sum_{k=1}^{i-1}(n-k+1)+(j-i+1)$

El término que interesa para obtener la fila es el de la sumatoria, ya que el término $(j - i + 1)$ sólo sirve para ubicar la posición *pos*, de una fila dada, en el vector.

Se tiene que: $\sum_{k=1}^{i-1}(n-k+1)=(i-1)(n-i/2+1)$

Por tanto $pos = (i - 1)(n - i/2 + 1)$ (7)

Utilizando nuevamente la fórmula para resolver una ecuación cuadrática se despeja i de (7) y para una *mtsd* de orden 7 se obtiene la tabla 10.

Tabla 10 Resultados para determinar el valor de una fila, conocidos *pos*, *n* y (6) para una *mtsd*

Pos	*Valor de i (columna 2)*	*Valor de i (columna 3)*
1	1,13	1,00
2	1,27	1,13
3	1,41	1,27
4	1,55	1,41
5	1,70	1,55
6	1,85	1,70
7	2,00	1,85
8	2,16	2,00
9	2,31	2,16
10	2,48	2,31
11	2,65	2,48
12	2,82	2,65
13	3,00	2,82
14	3,18	3,00
15	3,38	3,18
16	3,58	3,38
17	3,78	3,58
18	4,00	3,78
19	4,23	4,00
20	4,47	4,23
21	4,73	4,47
22	5,00	4,73
23	5,30	5,00
24	5,63	5,30
25	6,00	5,63
26	6,44	6,00
27	7,00	6,44
28	8,00	7,00

2 De acuerdo con [7] y hasta donde se logró establecer, este algoritmo no ha sido presentado previamente en referencia alguna.

Al solucionar la ecuación cuadrática resulta que las dos soluciones de la ecuación son positivas; por consiguiente, sólo se toma la menor, ya que ésta cumple $i \leq n$.

Como se presenta un desfase en una fila para los elementos situados en la última columna, similarmente como en la *mtii*, se despeja i pero después de restar una unidad a la posición *pos* para corregir el desfase.

La tercera columna de la tabla 10 se obtiene, entonces, despejando i de la fórmula

$$pos - 1 = (i - 1)(n - i/2 + 1) \qquad (8)$$

La forma cuadrática que se resulta es:

$$i^2 - i(3 + 2n) + 2(pos + n) = 0$$

Sea $W = 3 + 2$, entonces:

$$i = TRUNC(W - SQRT(W^2 - 8(pos + n)))/2)$$

y de (6)

$$j = pos - (i - 1)(n - i/2)$$

El algoritmo completo se presenta en la tabla 12.

Tabla 12 Algoritmo, con orden de magnitud constante, para determinar fila y columna, conocidos *pos*, *n* y (6) en una *mtsd*

```
//pos y n son parámetros de entrada
//fila y columna son parámetros de retorno
W = 3 + 2n
Fila = TRUNC(W - SQRT(W² - 8(pos + n)))/2)
Columna = pos - (fila - 1)(n - fila/2)
```

Un método alterno, con el cual se llega al mismo resultado, es el de establecer la simetría con la triangular inferior izquierda (*mtii*) y se desarrolla a continuación.

Para determinar la correspondencia se enumera la matriz *mtsd* de abajo hacia arriba (véase figura 4).

1	2	3	4	5	6	7
28	27	26	25	24	23	22
	21	20	19	18	17	16
		15	14	13	12	11
			10	9	8	7
				6	5	4
					3	2
						1

Figura 4 Matriz *mtsd* enumerada de abajo hacia arriba

El número de elementos diferentes de cero en una matriz triangular es $s = n(n + 1)/2$ por (1). Dada una posición *pos* cualquiera, perteneciente a *mtsd*, si se resta de s y se aplica la fórmula (4) se obtiene el complemento n de la fila que se desea hallar; por tanto, si este valor se resta a n se obtendrá la fila real a la que corresponde esa posición en la *mtsd*. Despejar la columna será como en el caso anterior (última instrucción de la tabla 12).

En la tabla 13 se presenta el algoritmo.

Tabla 13 Algoritmo, con orden de magnitud constante, para determinar fila y columna, conocidos *pos*, *n* y (6) en una *mtsd*

```
//pos y n son parámetros de entrada
//fila y columna son parámetros de retorno
    pos i = n(n + 1)/2 - (pos - 1)
    x = TRUNC (1 + SQRT (8 pos i – 7)/2)
    fila = n – x + 1
    columna = pos - (fila - 1)(n – fila/2)
```

Interesa demostrar la equivalencia de las fórmulas utilizadas en los algoritmos de las tablas 12 y 13 para determinar la fila.

De la tabla 12 se tiene que

$$fila = TRUNC(3 + 2n - SQRT(4n^2 - 8pos + 4n + 9)/2) \qquad (8')$$

y de la tabla 13 que

$$fila = (n + 1) - TRUNC(1 + SQRT(4n^2 - 8pos + 1 + 4n)/2) \qquad (8'')$$

Para establecer la equivalencia, se requiere entrar el término $(n + 1)$ dentro de la función *TRUNC*, para ello se debe entrar como $(n + 2)$ y restar una unidad al término *pos* dentro del radical.

$$fila = TRUNC((n + 2) - (1 + SQRT(4n^2 - 8(pos - 1) + 1 + 4n)/2)$$

$$fila = TRUNC(2(n + 2) - (1 + SQRT(4n^2 - 8pos + 8 + 1 + 4n))/2)$$

la cual, al simplificarla se obtiene

$$fila = TRUNC(3 + 2n - SQRT(4n^2 - 8pos + 4n + 9)/2) \qquad (9)$$

que es (8')

Un ejemplo para una *mtsd* de orden 7 se presenta en la tabla 14, haciendo uso de (8'), (8'') y (9).

Matriz triangular superior izquierda *(mtsi)* (véase figura 5)

1	*2*	*3*	*4*	*5*	*6*	*7*
1	2	3	4	5	6	7
8	9	10	11	12	13	
14	15	16	17	18		
19	20	21	22			
23	24	25				
26	27					
28						

Figura 5 Matriz *mtsi*

Consideremos la tabla 15 para determinar la fórmula de direccionamiento de *mtsi*.

Para un elemento perteneciente a la fila i y columna j se observa que su posición *pos* en el vector se obtiene con base en la suma de los elementos de las $(i - 1)$ filas anteriores más la columna j.

Lo anterior se puede expresar así:

$$pos = \sum_{k=1}^{i-1} (n - k + 1) + j$$

Tabla 14 Comprobación de validez de algoritmos en tablas 12 y 13

Pos	*Valor fila dado por (8')*	*Valor fila dado por (8'')*	*Valor fila dado por (9)*
1	*TRUNC*(1) = 1	8 - *TRUNC*(7,865) = 1	*TRUNC*(9-8) = *TRUNC*(1) = 1
2	*TRUNC*(1,135) = 1	8 - *TRUNC*(7,728) = 1	*TRUNC*(9-7,865) = *TRUNC*(1,135) = 1
3	*TRUNC*(1,271) = 1	8 - *TRUNC*(7,588) = 1	*TRUNC*(9-7,728) = *TRUNC*(1,272) = 1
4	*TRUNC*(1,411) = 1	8 - *TRUNC*(7,446) = 1	*TRUNC*(9-7,588) = *TRUNC*(1,412) = 1
5	*TRUNC*(1,553) = 1	8 - *TRUNC*(7,300) = 1	*TRUNC*(9-7,446) = *TRUNC*(1,554) = 1
6	*TRUNC*(1,699) = 1	8 - *TRUNC*(7,152) = 1	*TRUNC*(9-7,300) = *TRUNC*(1,7) = 1
7	*TRUNC*(1,847) = 1	8 - *TRUNC*(7) = 1	*TRUNC*(9-7,152) = *TRUNC*(1,848) = 1

Nótese la equivalencia entre las columnas 2, 3 y 4 de la tabla.

Tabla 15 Análisis para determinar *pos* en *mtsi*

i	*j*	*(Los almacenados antes)* +	?	*pos*
1	1	0	1	1
1	2	0	2	2
1	3	0	3	3
1	4	0	4	4
1	5	0	5	5
1	6	0	6	6
1	7	0	7	7
2	1	7	1	8
2	2	7	2	9
2	3	7	3	10
2	4	7	4	11
2	5	7	5	12
2	6	7	6	13
3	1	13	1	14
3	2	13	2	15
3	3	13	3	16
3	4	13	4	17
3	5	13	5	18
4	1	18	1	19
4	2	18	2	20
4	3	18	3	21
4	4	18	4	22
5	1	22	1	23
5	2	22	2	24
5	3	22	3	25
6	1	25	1	26
6	2	25	2	27
7	1	27	1	28

Resolviendo la sumatoria se obtiene:

$$pos = (i - 1)(n - i/2 + 1) + j \qquad (10)$$

$$\forall (i, j)/(j \leq n - i + 1)$$

De nuevo la división es real (no se *TRUNC*a ni se redondea), el análisis del proceso inverso se presenta más adelante.

Matriz triangular inferior derecha *(mtid)* (véase figura 6)

1	*2*	*3*	*4*	*5*	*6*	*7*
						1
					2	3
				4	5	6
			7	8	9	10
		11	12	13	14	15
	16	17	18	19	20	21
22	23	24	25	26	27	28

Figura 6 Matriz *mtid*

Considérese la tabla 16 para determinar la fórmula de direccionamiento de *mtid*.

Tabla 16 Análisis para determinar pos en *mtid*

i	*j*	*(Los almacenados antes)* +	?	*pos*
1	7	0	1	1
2	6	1	1	2
2	7	1	2	3
3	5	3	1	4
3	6	3	2	5
3	7	3	3	6
4	4	6	1	7
4	5	6	2	8
4	6	6	3	9
4	7	6	4	10
5	3	10	1	11
5	4	10	2	12
5	5	10	3	13
5	6	10	4	14
5	7	10	5	15
6	2	15	1	16
6	3	15	2	17
6	4	15	3	18
6	5	15	4	19
6	6	15	5	20
6	7	15	6	21
7	1	21	1	22

Tabla 16 (continuación)

i	*j*	*(Los almacenados antes)* + ?		*pos*
7	2	21	2	23
7	3	21	3	24
7	4	21	4	25
7	5	21	5	26
7	6	21	6	27
7	7	21	7	28

Para un elemento situado en la fila i se tiene que sumar el número de elementos presentes en las (i - 1) filas anteriores, así: Si $i = 5$, se debe sumar 1 + 2 + 3 + 4; es decir: $\sum_{k=1}^{i-1} k$

Ahora se requiere determinar la componente ? de la tabla 16 para determinar la posición *pos* en el vector.

Observando la tabla 16 se tiene:

Para la fila 2 columna 7 se debe sumar 2.

Para la fila 3 columna 5 se debe sumar 1.

Para la fila 4 columna 6 se debe sumar 3.

Para la fila 5 columna 7 se debe sumar 5.

El término *?* se obtiene mediante: $i + j - n$

por tanto:

$$pos = \sum_{k=1}^{i-1}(k) + (i + j - n)$$

la cual, al desarrollar la sumatoria y simplificar, se obtiene:

$$pos = i(i + 1)/2 + j - n \qquad (11)$$

$$\forall (i, j)/(j \geq n - i + 1)\cdot$$

El análisis del proceso inverso para las matrices *mtsi* y *mtid* es similar al de las dos primeras matrices:

- Se observa que hay una correspondencia entre las matrices *mtsi* y *mtsd*. La fórmula para la fila es la misma (segunda instrucción de la tabla 12), sólo cambia la de la columna, la cual se obtiene a partir de (10).

$$columna = pos - (fila - 1)(n - fila/2 + 1) \qquad (12)$$

- Para la matriz *mtid* se obtiene la fila con la misma fórmula que para la matriz *mtii* (primera instrucción de la tabla 8); y la columna se obtiene a partir de (11).

$$columna = pos - fila(fila + 1)/2 + n \qquad (13)$$

Conclusiones

Se han presentado fórmulas eficientes que facilitan la trasformación de coordenadas entre matrices triangulares y su representación en memoria contigua, mediante el uso de vectores.

Dicha transformación garantiza que se pueda llevar a cabo el trabajo operativo típico con matrices; es decir, sumas, restas, multiplicaciones, etc. Sin perder eficiencia en los algoritmos y ahorrando una cantidad significativa de memoria.

En escritos posteriores se presentarán análisis para otros tipos de matrices, en los cuales se espera aportar igualmente fórmulas y métodos eficientes de transformación.

Referencias bibliográficas

1. Grossman, S. I. *Álgebra Lineal.* McGraw-Hill Interamericana. 1996.
2. Hillier, F. S. y Lieberman, G. *Introducción a la investigación de operaciones.* McGraw-Hill Interamericana. 1989.
3. Horowitz, E. y Sahni, S. *Fundamentals of data structures.* Computer Sciencie. 1983.
4. Cairo, O. y Guardati, S. *Estructuras de datos y algoritmos.* McGraw-Hill Interamericana 1993.
5. Aho, A., Hopcroft, J. E. y Ullman, J. D. *Estructuras de datos y algoritmos.* Sistemas Técnicos de Edición. 1988.
6. Baase, S. y Van Gelder, A. *Computer Algoritms: Introduction to Design & Analysis.* Addison-Wesley. 2000.
7. Sahni, S. *Comunicación privada.* Mayo de 2001.