UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE CIENCIAS

ALGORITMOS PARA REORDENAR MATRICES RALAS

TESIS QUE PARA OBTENER EL TÍTULO
DE MAESTRO EN CIENCIAS (MATEMÁTICAS)
PRESENTA:

MAT. VIRGINIA ABRIN BATULE

OCTUBRE, 1985.

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

INDICE

INTRODUCCION

Dentro de la literatura matemática existen fundamentalmente dos concepciones, ambas independientes y a la vez relacionadas con el problema del ancho de banda. Una, con gráficas de puntos y líneas y otra con arreglos de números reales conocidos como matrices. Para una gráfica G con n puntos, el problema consiste en etiquetar los vértices v_i de G con enteros positivos distintos $f(v_i)$ para $i = 1,2,\ldots,n$, de tal manera que el valor máximo de $|f(v_i) - f(v_j)|$ tomado sobre todas las parejas de vértices adyacentes sea mínimo. Mientras que para una matriz A simétrica el problema del ancho de banda se traduce en encontrar una permutación simétrica A_0 de A, que minimice al máximo de $|i - j|$ sobre todos los elementos a'_{ij} de A_0 con $a'_{ij} \neq 0$. La equivalencia entre estos dos problemas queda establecida cuando para cada matriz A simétrica de orden n es posible asociarle una gráfica G^A con n vértices, en donde la pareja $\{i,j\}$ es una arista de G^A, si el elemento a_{ij} es distinto de cero para $i \neq j$.

La idea de reducir el ancho de banda de una matriz surge con los sistemas de ecuaciones cuya matriz de coeficientes contiene una gran cantidad de elementos iguales a cero; es decir, es rala, ya que para la solución del sistema se opera únicamente con los elementos de la banda.

En el siglo pasado, Gauss desarrolló técnicas para explotar la raleza en sistemas de este tipo. Posteriormente, en los años cincuenta, el tema cobra gran interés, principalmente, en ingeniería a través de los problemas de análisis estructural ya que de ahí se obtienen sistemas con un gran número de ecuaciones donde la matriz del sistema es rala, simétrica y definida positiva. Un mayor auge del tema se propicia con el desarrollo de las computadoras a mediados de los años sesenta.

Independientemente, el problema del ancho de banda en gráficas se empezó a trabajar en 1962, en el Jet Propulsion Laboratory de Pasadena,CA. En 1969 se da a conocer un algoritmo para reducir el ancho de banda de una gráfica debido a E. Cuthill y McKee [16], que con los años ha sido mejorada entre otros por A. George [17] y Gibbs *et al* [18]. Papadimitrion, en 1976 [27], demuestra que el problema de minimizar el ancho de banda es NP-Complete. En [9] se tiene un resumen y una bibliografía con resultados hasta 1981 sobre el problema del ancho de banda en gráficas. Por lo

que se refiere a matrices se puede consultar [11,17,33] para una información más amplia.

En el desarrollo de este trabajo se exponen aspectos importantes para el manejo de matrices ralas. En el capítulo I se muestran algunos esquemas específicos para almacenar matrices en banda y perfil así como reordenamiento del sistema $Ax = b$. En el capítulo II se establece la relación entre gráficas y matrices y se describen algoritmos para reducir el ancho de banda, seleccionar vértices iniciales, reducir el perfil y reducir el llenado. El capítulo III muestra resultados sobre los ordenamientos obtenidos, al discretizar la ecuación de Poisson usando 2 - 6 elementos descritos por A. George en [15] en donde 2 es el grado del polinomio y 6 es el número de nodos asociado con el elemento. En un segundo ejemplo se utiliza la misma ecuación, que se ha discretizado en una malla con dimensiones variables. Por último, en este capítulo se analiza una muestra de matrices generadas aleatoriamente, de orden 10, para comparar los resultados que sobre el ancho de banda proporcionan los algoritmos con el mínimo ancho de banda.

Una paquete computacional para la instrumentación de los algoritmos en el capítulo II se encuentra en el apéndice; con éste se probaron los ejemplos antes mencionados.

CAPITULO I

MANEJO DE SISTEMAS RALOS

Una matriz con un porcentaje pequeño de elementos distintos de cero se dice que es *rala*. En la práctica, una matriz de orden n (grande) es *rala* si tiene de dos a diez elementos distintos de cero en cada renglón, para n suficientemente grande. [33]

Consideremos un sistema de ecuaciones lineales

$$Ax = b$$

con A simétrica, definida positiva y rala de orden n. La manera de encontrar la solución del sistema resulta ser más eficiente si se aprovecha el que A es rala. Por ejemplo, se puede elegir un buen esquema para almacenar la matriz en

donde se guarde la menor cantidad posible de elementos iguales a cero. Por otra parte, el número de operaciones decrece si se evitan, en la medida de las posibilidades, las operaciones triviales, lo que reduce el tiempo de procesador. Estas consideraciones nos hacer pensar en la obtención de esquemas óptimos, con los que se obtenga la solución precisa, y que tanto el espacio en la memoria como el tiempo de procesador sean mínimos. Desde luego que estos tres requisitos son incompatibles, por lo que se debe restar importancia a algunos, para satisfacer otros, según nos interese.

En este capítulo se analizan dos aspectos importantes del manejo de matrices ralas:

a) Esquemas para almacenarlas.

b) Métodos para reordenarlas.

Que se emplearán posteriormente en la solución de sistemas.

$$Ax = b$$

En la primera parte se muestran esquemas para almacenar matrices ralas, junto con algunas de sus ventajas y desventajas respectivas. La segunda, se inicia con un ejemplo en el que se puede ver que, dado un sistema de la forma

$$Ax = b$$

con A simétrica, definida positiva y rala de orden n, antes que resolver directamente el sistema

$$Ax = b,$$

resulta preferible encontrar una matriz de permutación P, para luego resolver el sistema

$$PAP^tPx = Pb, \; y = P^tx$$

I.1 ESTRATEGIAS DE ALMACENAMIENTO

Las matrices grandes y ralas se pueden almacenar, en general, en arreglos donde sea posible localizar los elementos de la matriz por medio de subíndices. Desde luego, también es posible almacener dichos elementos en un arreglo de una dimensión, donde el renglón i se encuentra después del renglón i - 1 y antes del renglón i + 1; el elemento a_{ij}, se localiza en la posición ni + j - n, donde n es el orden de la matriz. A menudo este esquema de almacenamiento da lugar a un programa más eficiente que un arreglo de dos dimensiones.

Antes de continuar con los esquemas de almacenamiento, sería bueno preguntarse: Dada una matriz rala A, de orden n ¿cuándo es igual a cero el elemento a_{ij}?. Esta pregunta obedece a la preocupación de lograr una mayor eficiencia en el manejo de matrices grandes. (Vgr. de orden mayor que cien); en efecto, si sabemos cuando un elemento es cero, posiblemente se pueda evitar efectuar algunas de las operaciones triviales, para lograr asi un ahorro en el tiempo de procesador. Para matrices de orden menor que cien, esta pregunta tiene sentido; no así en otros casos; ya que tardaríamos demasiado tiempo en diferenciar sus elementos. Por ejemplo para una matriz rala de orden 10^{10}, no existe computadora alguna que pueda almacenar esta cantidad de elementos, y mucho menos resolver el sistema. [28]

En esencia, se tienen dos componentes básicas que se usan en los esquemas de almacenamiento. Una consiste en almacenar los elementos distinos de cero en un arrego unidimensional, al cual se le puede llamar un *arreglo primario*. Siempre que sea posible, se buscará que dichos elementos estén en un área determinada de la matriz. La otra componente la conforma un medio para reconocer qué elementos de la matriz son almacenados en un arreglo primario. Usualmente, tal estructura la integran uno o dos arreglos unidimensionales de identificadores, y generalmente se le

llama *arreglo secundario*.

A continuación se consideran algunos esquemas de almacenamiento para cualquier matriz rala. Más tarde veremos soluciones específicas, como son los esquemas en banda o de perfil.

ESQUEMA I [22]

La construcción de este esquema es muy simple y consiste en colocar a los elementos distintos de cero de la matriz, en un arreglo primario, mencionados por hileras y en orden de aparición; el secundario consta de una sucesión de ceros y unos que recorre el conjunto de posiciones de un arreglo primario (la *sombra* de la matriz) en el mismo orden. Por ejemplo, sea A la matriz de la figura 1.1.

$$A = \begin{bmatrix} 91 & 0 & 0 & 7 & 0 \\ 0 & 40 & 21 & 17 & 4 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 8 & 3.7 \end{bmatrix}$$

Figura 1.1.

El arreglo primario correspondiente a A es

{91 7 40 21 17 4 3 1 8 3.7}

y la sucesión binaria (1.1.1)

1er RENGLON	2º RENGLON	3er RENGLON	4º RENGLON	5º RENGLON
10010	01111	00100	00000	10011

constituye el arreglo secundario.

Según se ve, en esto también los elementos iguales a cero se encuentran en algún lugar del almacén. Así pues, este esquema no resulta muy recomendable pues es claro que resultan más eficientes los esquemas que no utilizan ningun espacio en el almacen para los elementos iguales a cero. Otra parte inoperante del esquema recien descrito es la gran dificultad para realizar operaciones con matrices ralas almacenadas de esta forma.

ESQUEMA II [7] [22]

Los elementos distintos de cero de una matriz quedan asignados en un arreglo primario, mientras que los subíndices se colocan en un par de arreglos secundarios. Como cada elemento queda completamente determinado por el número de renglón y el número de columna, es posible almacenarlos en cualquier orden. De acuerdo a esto la matriz de la figura 1.1 queda representada por el esquema siguiente:

```
ARREGLO REAL        A = {3.7 4 7 17 8 21 3 40 91 1 0}
ARREGLO DE ENTEROS IA = {  5 2 1  2 5  2 3  2  1 5 0}  (1.1.2)
ARREGLO DE ENTEROS JA = {  5 5 4  4 4  3 3  2  1 1 0}
```

Una de las ventajas de este tipo de empacamiento consiste en que si se quiere aumentar nuevos elementos distintos de cero, no es necesario mover los que ya se tienen; simplemente los nuevos elementos se añaden al final. Resulta conveniente incluir un elemento igual a cero en cada arreglo, para señalar el final de la matriz. De este esquema se desprende uno similar, que consiste en colocar los elementos de la diagonal al principio y en seguida el resto. Si la matriz resulta simétrica e.d. si

$$A = A^t,$$

bastará con almacenar los elementos de la diagonal y los de abajo de ésta.

Como ejemplo, consideremos la matriz de la figura 1.1; el esquema toma la forma siguiente:

```
ARREGLO REAL        A = {91 40 3 3.7 7 21 17 4 1 8 0}
ARREGLO DE ENTEROS IA = { 1  2 3   5 1  2  2 2 5 5 0}  (1.1.3)
ARREGLO DE ENTEROS JA = { 1  2 3   5 4  3  4 5 1 4 0}
```

En la práctica se ha visto que los esquemas que usan listas ligadas son muy convenientes, razón por la cual analizaremos algunos tipos de esquemas con esta propiedad.

ESQUEMA III [1] [22] [23]

Consideremos los arreglos, (ARREGLO REAL A, ARREGLO DE ENTEROS IA, ARREGLO DE ENTEROS JA) como antes y el ARREGLO ILG, que se construirá de tal manera que los elementos distintos de cero pueden rastrearse renglón por renglón.

Sea A la matriz

$$A = \begin{bmatrix} -1 & 0 & 0 & 4 & 0 \\ 0 & .5 & 0 & 2.5 & 0 \\ 0 & 0 & 7 & 8 & 0 \\ 4 & 2.5 & 8 & -5 & 0 \\ 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

Figura 1.2

DOMICILIOS	1	2	3	4	5	6	7	8	9	10	11	12
ARREGLO REAL A=	{-1	.5	7	-5	3	4	4	2.5	2.5	8	8	0}
ARREGLO DE ENTEROS IA=	{ 1	2	3	4	5	1	4	2	4	3	4	0}
ARREGLO DE ENTEROS JA=	{ 1	2	3	4	5	4	1	4	2	4	3	0}
ARREGLO DE ENTEROS ILG=	{ 6	8	10	5	0	2	9	3	11	7	4	0}

En este ejemplo podemos observar el funcionamiento del arreglo ILG ; para mayor claridad, tomemos el elemento a_{11}, este se encuentra en el domicilio 1, ILG (1) apunta al siguiente elemento distino de cero del renglón 1; en este caso es

a_{14} y su domicilio es 6. De la mismo forma ILG(6) apunta al siguiente elemento distinto de cero del renglón 1, que se encuentre después de a_{14}. Como los elementos que le siguen a a_{14} en el renglón 1 son todos cero, ILG se dirige al segundo renglón y se encuentra con a_{22} que es el primer elemento distinto de cero y que se localiza en el domicilio 2. Y así sucesivamente.

El esquema con listas ligadas es útil, si se puede ingresar a la cadena por varios de los elementos. Las direcciones de los elementos de la diagonal pueden usarse como entradas, de tal manera que para localizar el elemento a_{ij} la cadena se separa en el domicilio i, si $i < j$ o en $i - 1$ si $j < i$. Después se continúa a lo largo de la cadena, hasta obtener el elemento a_{ij} o se comprueba que a_{ij} es cero. Con este tipo de ligas, se puede recorrer la matriz renglón por renglón, de arriba hacia abajo o de abajo hacia arriba; desde luego que esto se puede hacer también para las columnas.

Entre los inconvenientes, encontramos que:

a) Es necesario tener un método para formar las ligas

b) Se requiere un espacio extra en el almacen.

c) El tiempo de procesador es mayor, pues es necesario revisar la dirección de las ligas.

d) A menos que la matriz esté enteramente en la memoria principal (memoria primaria o almacenamiento) se requiere de respaldos para ejecutar las operaciones con matrices que usan listas ligadas.

Como puede verse, el esquema III hace uso de cuatro arreglos además de las direcciones. En una matriz en donde se ha dado un orden a los elementos distintos de cero, no es necesario utilizar los índices de renglón y de columna para cada elemento; en estos casos basta con seguir un sentido en los renglones y o columnas y adoptar un solo arreglo, de renglón o columna; desde luego que es necesario indicar dónde empieza y donde termina un renglón o columna, para lo cual se emplearán otros índices. Si se decide almacenar los elementos de una matriz en el sentido de los renglones, entonces se omitirá el arreglo de índices de renglón. El esquema IV precisa estas ideas, junto con un ejemplo.

ESQUEMA IV [ ] [ ] [ ] [ ]

En este esquema se utilizan tres arreglos. El primero contiene los elementos distintos de cero de la matriz, el segundo incluye a los índices de columna de cada elemento y el último, es un arreglo de enteros que indica cual es el primer elemento distinto de cero de cada renglón. A este último le llamamos IPRIN y está construido de tal manera que

el número de elementos distintos de cero en el i-ésimo renglón es IPRIN(I + 1) - IPRIN(I). En una matriz de orden n, tiene n + 1 elementos IPRIN.

Consideremos la matriz de la figura 1.2; la manera de almacenarla según el esquema IV es la siguiente:

ARREGLO REAL A= {-1 4 .5 2.5 7 8 4 2.5 8 -5 3}
ARREGLO DE ENTEROS JA= { 1 4 2 4 3 4 1 2 3 4 5} (1.1.5)
ARREGLO DE ENTEROS IPRIN= { 1 3 5 7 11 12}

Se puede construir un esquema similar, si este se construye estableciendo una manera de rastrear la matriz por columnas e.d un sentido en las columnas.

Es bueno señalar que una de las ventajas del esquema IV, sobre el esquema III, es la supresión del arreglo de enteros IA.

En algunas ocasiones se utilizan elementos para indicar el principio y el final de una hilera en lugar del arreglo IPRIN ó en yuxtaposición con éste; a este tipo de elementos se les denomina *elementos mudos*. Dichos elementos pueden incluirse de diversas formas en el arreglo de índices de las columnas. Por ejemplo, la presencia de un elemento igual a cero en el arreglo de índices de columnas señala a un elemento mudo y este mismo puede indicar el nú

mero de renglón o denotar el final de la matriz. Usando elementos mudos, el esquema que usa el arreglo IPRIN para la matriz de la figura 1.2 se transforma en el siguiente:

```
ARREGLO REAL          A= { [1] -1 4 [2] .5 2.5 [3] 7 8 [4] 4 2.5 8 -5
ARREGLO DE ENTEROS JA= { [0]  1 4 [0]  2  4  [0] 3 4 [0] 1  2  3  4

                           [5] 3 [0] }
                           [0] 5 [0] }   (1.1.6)
```

Otra opción, para esquemas que usan elementos mudos, es la de incluir en el arreglo de índices de columna, el número de renglón precedido por un signo menos. De acuerdo a esto el arreglo anterior toma la forma:

```
ARREGO REAL          A= { [ x] -1 4 [ x] .5 2.5 [ x] 7 8 [ x]
ARREGLO DE ENTEROS  JA= { [-1]  1 4 [-2]  2  4  [-3] 3 4 [-4]

                           4 2.5 8 -5 [ x]  3 [ x] }
                           1  2  3  4 [-5]  5 [ 0] }   (1.1.7)
```

En algunas ocasiones y para efectos de realizar operaciones entre matrices, es conveniente usar enteros "muy grandes" como identificadores para elementos mudos; una forma de construirlos consiste en fijar una constante y sumarle el número de renglón; de esta forma el arreglo (1.1.7) se pue-

de expresar:

```
ARREGLO REAL          A= {x      -1 4  x     .5  2.5 x
ARREGLO DE ENTEROS   JA= {10001   1 4  10002  2   4   10003
                          7 8  x     4  2.58 -5  x     3
                          3 4  10004 1  2 3  4   10005 5
                          x   }
                          99999}
```

(1.1.8)

En este ejemplo el número 99999 señala el final de la matriz.

Otra opción en el uso de elementos mudos consiste en incluir enteros que correspondan al número de elementos distintos de cero del renglón siguiente; éstos serán incluidos en el arreglo de índices de columna. Cuando los elementos de un renglón son todos iguales a cero no es necesario indicar el número de renglón. Desde luego, esto no se da al resolver sistemas de ecuaciones lineales.

El siguiente ejemplo muestra la aplicación del esquema con elementos mudos, en donde se indica, en el arreglo de columnas, el número de elementos distintos de cero del renglón siguiente.

Considérese como A a la matriz

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & .5 & 7 & 0 \\ 3 & 1 & 4 & 0 \\ 1.7 & 0 & 0 & 3.5 \end{bmatrix}$$

ARREGLO REAL A= {x x .5 7 x 3 1 4 x 1.7 3.5}

ARREGLO DE ENTEROS JA= {0 2 3 4 3 1 2 3 2 1 4 } (1.1.9)

Recordemos los esquemas que usan elementos mudos; entre estos, el primero usa los elementos mudos para señalar el número de renglón. Los elementos mudos que se usan en los arreglos de reales se pueden omitir, excepto en el esquema que se muestra en 1.1.6. Esto se hace para ahorrar espacio en el almacen. Desde luego que al quitarlos se pierde la correspondiencia uno a uno entre los índices de columnas y los elementos de la matriz, perdiendo también los domicilios de estos.

Los elementos distintos de cero de una matriz, también se pueden almacenar en un arreglo de una dimensión; en la construcción de dicho arreglo se incluyen identificadores de renglón o de columna según se haya elegido recorrer la matriz por renglones o columnas.

El siguiente es un ejemplo, en donde el esquema que se usa para almacenarla, es de una dimensión y contiene identificadores de renglón.

$$A = \begin{bmatrix} .3 & 0 & 1 & 0 \\ 0 & .3 & 0 & 1 \\ 1 & 0 & .3 & 0 \\ 0 & 1 & 0 & .3 \end{bmatrix}$$

ARREGLO REAL A= {-1 [1.3] [3 1] -2 [2.3] [4 1] -3 [1 1] [3.3]

-4 [2 1] [4.3] 0} (1.1.10)

Observemos que, en este esquema, cada elemento distinto de cero está precedido por el índice de columna y que cada renglón se índica con el índice correspondiente con signo menos.

ESQUEMA V [22]

En este esquema se describe una manera de almacenar dos tipos especiales de matrices: las matrices triangulares inferiores y las de Hessenberg.

Una matriz triangular inferior L de orden n, se puede almacenar en un arreglo de una dimensión con $n(n+1)/2$ elementos. Si la matriz se almacena por renglones, la sucesión de almacenamiento está dada por:

ARREGLO $L = \{\ell_{11}/\ell_{21}\ell_{22}/\ell_{31}\ell_{32}\ell_{33}/\ldots/\ell_{1n}\ldots\ell_{nn}\}$;

en donde la ubicación del elemento ℓ_{ij}, es (i/2) (i-1) + j.

Una matriz *superior de Hessenberg*, es una matriz triangular superior, en donde los elementos de la *subdiagonal principal* pueden ser distintos de cero.

Sea H una matriz de Hessenberg de orden n. El arreglo unidimensional que contiene a los elementos de H, almacenados por renglones, tiene (n/2)(n + 3)-1 elementos y el arreglo toma la forma siguiente:

ARREGLO $H = \{h_{11}\ldots h_{1n}/h_{21}\ldots h_{2n}/h_{32}\ldots h_{3n}/\ldots/h_{n,n-1}\ h_{nn}\}$.

El elemento h_{ij} de la matriz H se localiza en el lugar $(i/2)(2 + 3 - i) - n + j - 1$.

ESQUEMA VI [10] [22] [33]

Consideremos ahora una matriz simétrica de orden n. Para cada renglón i, se puede definir el ancho de banda como sigue: para cada $1 \leq i \leq n$, sea

$$\beta_i = \max_{j<i} |i - j|;$$

a este número se le llama el *i-ésimo ancho de banda de la* matriz A. El *ancho de banda* de A está dado por

$$\beta = \max_{1\leq i\leq n} \beta_i$$

En algunos trabajos [22] el i-ésimo ancho de

banda es considerado como $2\beta_i + 1$, con igual convención para el ancho de banda de A. En lo sucesivo adoptaremos esta convención al tratar con matrices no simétricas.

Sea A una matriz simétrica de orden n, en donde el ancho de banda para cada renglón es el mismo y sea este $2\beta + 1$. Como A es simétrica, unicamente se almacenan los elementos a_{ij} con $j \leq i$. Si $b = \beta + 1$, entonces el arreglo tendrá $(b/2)(2n - b + 1)$ elementos y a_{ij} se encuentra en el lugar $(i/2)(i - 1) + j$ si $i \leq b$ o en el lugar $(i - b/2)(b - 1) + j$ si $i \geq b$.

Una forma de almacenar a A consiste en emplear elementos mudos; como un ejemplo de ello, consideremos la matriz de la figura 1.3.

$$A = \begin{bmatrix} 1 & 2 & 1/2 & 0 & 0 & 0 \\ 2 & 1 & 0 & 1/4 & 0 & 0 \\ 1/2 & 0 & 1 & 0 & 7 & 0 \\ 0 & 1/4 & 0 & 1 & 0 & \sqrt{2} \\ 0 & 0 & 7 & 0 & 1 & 8 \\ 0 & 0 & 0 & \sqrt{2} & 8 & 1 \end{bmatrix}$$

figura 1.3

Esta matriz es simétrica de orden 6 y su ancho de banda es 5; el esquema 1.1.11 nos muestra el empleo de elementos mudos para almacenar la matriz A y el elemento a_{ij} se ubica en el lugar $ib - i + j$.

ARREGLO A = {x x 1|x21|1/2 0 1|1/4 0 1|7 0 1|$\sqrt{2}$ 8 1}

(1.1.11)

Otra opción para almacenar matrices con el mismo ancho de banda en cada renglón, consiste en emplear un arreglo de dos dimensiones de orden $n \times b$. Cuando el arreglo incluye elementos mudos, el elemento a_{ij} con $j \leq i$, se puede localizar en el lugar cuyos índices corresponden a(i, $b - i + j$).

En el caso de una matriz tridiagonal, en donde el ancho de banda es tres, generalmente se puede almacenar en un arreglo unidimensional de orden n, junto con dos arreglos de orden $n - 1$. Desde luego que, si la matriz es simétrica, solo se utiliza uno de estos arreglos.

Los esquemas de este tipo son de gran utilidad, pues no requieren de arreglos secundarios, ni tampoco tiempo para procesarlos.

Para matrices con ancho de banda local variable, se puede usar un esquema similar al anterior, en donde será necesario conocer el número de elementos distintos de cero de cada renglón o la posición de los elementos de la diagonal.

Como un ejemplo consideremos a la matriz siguiente:

$$A = \begin{bmatrix} 3 & .5 & 0 & 0 & 4 & 0 & 1 \\ .5 & 3 & 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 3 & 1 & 9 & 4 & 1 \\ 0 & 0 & 1 & 3 & 7 & .2 & 0 \\ 4 & 5 & 9 & 7 & 1 & 1 & 1 \\ 0 & 0 & 4 & .2 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

A es simétrica de orden 7. El siguiente esquema para almacenar A consta de dos arreglos. El primero contiene a los elementos de la diagonal y a todos los elementos a_{ij} con $j \leq i$, a partir del primer elemento distinto de cero de cada renglón; el segundo arreglo es un vector de posiciones para los elementos de la diagonal.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
ARREGLO A = {3/	.5	3/	3/	1	3/	4	5	9	7	1/	4	.2	1	1/	1	0	1	0	1	0	1/}	
ARREGLO DE ENTEROS																						
AD = {1		3	4		6					11					15							22}

Cada elemento a_{ij} con $j \leq i$ contenido en el arreglo se localiza en el lugar AD(i) + j - i y el número de elementos del renglón i que se encuentran en el almacenes AD(i) - AD(i-1) para $i > 1$.

Este esquema se usa generalmente para matrices simétricas ralas y definidas positivas; desde luego que es útil también para matrices tringulares inferiores y superiores.

I.2 REORDENAMIENTO DEL SISTEMA $Ax = b$

Consideremos un sistema de ecuaciones

$$Ax = b,$$

en donde A es una matriz simétrica, definida positiva, rala y de orden n.

Una manera de resolver el sistema consiste en aplicarle el método de Cholesky, que busca factorizar a la matriz A como

$$A = LL^t;$$

la matriz L resulta ser triangular inferior y con los elementos de la diagonal mayores que cero[ ]Hecho esto, resolver el sistema

$$Ax = b,$$

es equivalente a resolver los sistemas

$$Ly = b \quad y \quad L^t x = y,$$

que son fáciles de resolver por substitución directa e inversa, respectivamente [20].

Resulta claro que estos sistemas son más fáciles de resolver si la parte baja de la diagonal de la matriz L contiene una gran cantidad de elementos iguales a cero. Desafortunadamente, el hecho de que la matriz A sea rala no garantiza que la matriz

$$L + L^t$$

también lo sea. En efecto, sea A_0 la matriz siguiente, cuyos elementos distintos de cero se denotan mediante una x:

$$A_0 = \begin{bmatrix} x & x & x & x & x \\ x & x & & & \\ x & & x & & \\ x & & & x & \\ x & & & & x \end{bmatrix}$$

se puede ver que A_0 es rala; sin embargo el factor L_0 de Cholesky de A_0 tiene la forma siguiente, en la cual los elementos señalados con (x) pueden ser distintos de cero:

$$A_0 = \begin{bmatrix} x & & & & \\ x & x & & & \\ x & x & x & & \\ x & \textcircled{x} & \textcircled{x} & x & \\ x & \textcircled{x} & \textcircled{x} & \textcircled{x} & x \end{bmatrix}$$

Con este sencillo ejemplo se puede ver, que la ventaja de tener una gran cantidad de ceros en A, solo simplifica el procedimiento para encontrar el factor L de Cholesky y que no necesariamente reduce la complejidad de los sistemas

$$Ly = b \quad y \quad L^t x = y.$$

Sin embargo, en algunas ocasiones se puede encontrar un sistema

$$\bar{A}\bar{x} = \bar{b}$$

equivalente al sistema

$$Ax = b,$$

de tal manera que al factorizar $\bar{A}$ como

$$\bar{A} = \bar{L}\,\bar{L}^t,$$

la matriz $\bar{L}$ tenga a lo más tantos elementos distintos de cero como la parte inferior de $\bar{A}$. [14]

Por lo pronto, observemos que reordenar las ecuaciones equivale a premultiplicar ambos miembros del sistema

$$Ax = b$$

por una matriz P de permutaciones, obteniendo

$$PAx = Pb$$

A su vez, reordenar x con la misma permutación, equivale a reemplazarla por Px. Dado que

$$P\,P^t = I = P^tP$$

(por ser las columnas de P las mismas que las de I, solo que permutadas) es posible reescribir el sistema reordenado en la forma

$$(PAP^t)(Px) = Pb$$

Asi pues, una vez encontrado un ordenamiento P adecuado, se permutan los elementos de b y las hileras de A según P y se reordenan las columnas de A según la misma permutación; luego se resuelve el sistema resultante, que

será

$$\bar{A}\bar{x} = \bar{b},$$

en donde

$$\bar{A} = PAP^t, \quad \bar{b} = Pb$$

La solución x del sistema original se obtiene aplicando a $\bar{x}$ la reordenación inversa, dada por P^t.

Sea $A_0x = b_0$ como en el ejemplo anterior y considérese la matriz de permutaciones

$$P = \begin{bmatrix} \cdot & \cdot & 1 & \cdot & \cdot \\ \cdot & 1 & \cdot & \cdot & \cdot \\ 1 & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & 1 & \cdot \\ \cdot & \cdot & \cdot & \cdot & 1 \end{bmatrix}$$

De lo anterior se obtiene que resolver el sistema $A_0x = b_0$ es equivalente a resolver

$$\bar{A}_0\bar{x} = b_0, \quad \bar{x} = Px$$

donde

$$\bar{A}_0 = PA_0P^t = \begin{bmatrix} x & . & x & . & . \\ . & x & x & . & . \\ x & x & x & x & x \\ . & . & x & x & \\ . & . & x & . & x \end{bmatrix}$$

La factorización de Cholesky de $\bar{A}_0$ tiene como primer factor a

$$\bar{L}_0 = \begin{bmatrix} x & & & & \\ . & x & & & \\ x & x & x & & \\ . & . & x & x & \\ . & . & x & \textcircled{x} & x \end{bmatrix}$$

según se verifica facilmente. Como el número de elementos de $\bar{L}_0$ iguales a cero puede ser mayor que el de L_0, resulta en general más fácil resolver los sistemas

$$\bar{L}_0 y = Pb_0 \text{ y } \bar{L}_0^t Px_0 = y$$

que los sistemas

$$L_0 y = b \text{ y } L_0^t x = y,$$

respectivamente.

Para describir la propiedad de la matriz A que asegura la presencia de una gran cantidad de ceros en la parte inferior del factor L de A, se introducirán algunos conceptos.

En la primera sección de este capítulo (I.1), esquema VI, hablamos del ancho de banda de una matriz; la definición siguiente precisa este concepto.

Definición

Sea $A = (a_{ij})$ una matriz. El ancho de banda de A es

$$\beta(A) = \max_{1<i<n} \beta_i(A), \text{ donde}$$

$$\beta_i(A) = \max\{|i - j| : a_{ij} \neq 0\} \quad i = 1,2,\ldots, n.$$

En general, se tiene el siguiente resultado, cuya validez es clara al aplicar eliminación gaussiana a la matriz A para obtener L^t. Para una demostración amplia ver A. George [7].

Proposición 1

Sean A una matriz simétrica definida positiva y L tal que $A = LL^t$. Entonces

$$\beta(L) = \beta(A).$$

De esta forma concluimos que si existe una matriz P de permutaciones tal que

$$\text{si } \beta(PAP^t) < \beta(A) \text{ entonces } \beta(\bar{L}) < \beta(L),$$

donde

$$PAP^t = \bar{L}\bar{L}^t \quad y \quad A = LL^t.$$

Dado que el número de elementos iguales a cero puede ser considerablemente mayor en $\bar{L}$ que en L, la solución del sistema

$$Ax = b,$$

se facilita resolviendo el sistema equivalente

$$(PAP^t)\,Px = Pb,$$

encontrando una matriz P de permutaciones adecuada, que reduzca el ancho de banda de la matriz A. Para esto, se dispone de técnicas basadas en la teoría de gráficas que son de gran utilidad [9]. Dedicaremos el capítulo siguiente a presentar algunas de dichas técnicas.

CAPITULO II

ALGORITMOS DE REORDENAMIENTO

El contenido de este capítulo está dado en cuatro secciones. La primera parte trata sobre la relación entre gráficas y matrices. En la segunda se muestra el algoritmo de E. Cuthill y Mc. Kee, para reducir el ancho de banda de una matriz. La parte tercera contiene el R.C.M (el algoritmo con inversión de E. Cuthill y Mc. Kee), algoritmo que sirve para reducir el perfil de una matriz y es debido a Allan George. Por último, presentamos el algoritmo de deficiencia mínima, el cual permite disminuir el llenado de una matriz al aplicar eliminación gaussiana y fue propuesto por J. Rose.

II.1 GRAFICAS Y MATRICES [1, 2, 3, 5, 19]

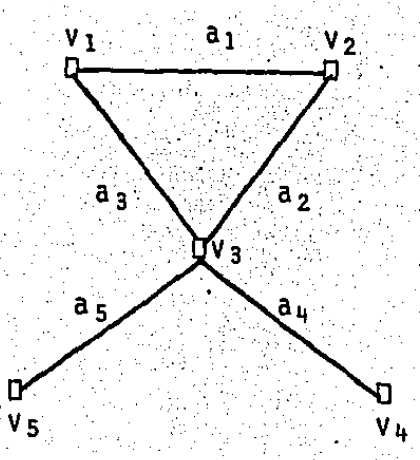
Existe una correspondencia entre matrices de orden n y gráficas con n puntos que resulta de lo más conveniente para nuestros fines, pues gracias a ella podemos expresar adecuadamente algunos algoritmos para reordenar las hileras y las columnas de una matriz rala y simétrica. En esencia, a una matriz A tal de orden n se le asocia una gráfica de n vértices numerados del 1 al n, de los cuales dos, (el i y el j) estarán conectados por una arista si y sólo si $a_{ij} \neq 0$. Para precisar lo anterior, conviene recordar algunos conceptos elementales de la teoría de gráficas, los cuales nos servirán para expresar otros como ancho de banda, perfil, llenado etc. que tienen su origen en la teoría de matrices.

Una gráfica G consta de dos conjuntos V(G), A(G) y una función

$$\varphi_G : A(G) \longrightarrow 2^{V(G)}$$

de tal manera que $|\varphi_G(a)| = 2$ para toda *arista* en A(G) (toda arista tiene dos y solo dos vértices). El conjunto V(G) es no vacio y a sus elementos se les llama *vértices* de G.

EJEMPLO



G

figura 2.1.

La gráfica G de la figura 2.1 tiene como conjunto de vértices a:

$$V(G) = \{v_1, v_2, v_3, v_4, v_5\};$$

de aristas al conjunto

$$A(G) = \{a_1, a_2, a_3, a_4, a_5\}$$

y la función que le asigna a cada arista sus extremos, que da descrita por

$$\varphi(a_1) = \{v_1, v_2\} \quad \varphi(a_2) = \{v_2, v_3\} \quad \varphi(a_3) = \{v_1, v_3\}$$

$$\varphi(a_4) = \{v_3, v_4\} \quad y \quad \varphi(a_5) = \{v_3, v_5\}$$

Dos aristas son *incidentes* si tienen un extremo en común. Dos vértices son *adyacentes*, si existe alguna arista que los une.

El *grado* de un vértice v es el número de aristas que lo tienen como extremo. Se denota por gr(v).

Una *trayectoria* en una gráfica es una sucesión alternada de vértices y aristas de tal manera que ningún vértice se repite y por lo tanto ninguna arista.

Una gráfica G es *conexa* si para todo par de vértices existe una trayectoria que los une.

EJEMPLOS

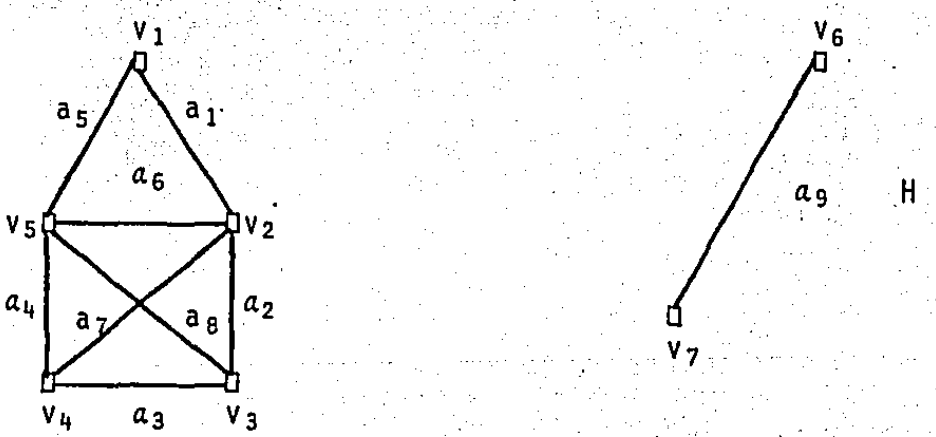


figura 2.2.

En la figura 2.2. las aristas a_1 y a_2 son incidentes en v_2, los vértices v_4 y v_2 son adyacentes; los une la arista a_7; $gr(v_2) = gr(v_5) = 4$ y $gr(v_4) = gr(v_3) = 3$

La sucesión $v_1\ a_1\ v_2\ a_2\ v_3\ a_3\ v_4\ a_4\ v_5$ es una trayectoria. Los vértices $\{v_1, v_2, v_3, v_4, v_5\}$ junto con el conjunto de aristas $\{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8\}$ constituya una gráfica conexa, mientras que la gráfica formada por $\{v_1, v_2, v_3, \ldots, v_7\}$ y $\{a_1, a_2, \ldots, a_9\}$ no lo es.

Las gráficas que se utilizarán en este trabajo serán conexas. De hecho no hay ninguna perdida de generalidad en ello, pues si la gráfica no es conexa los resultados se pueden aplicar a cada una de sus componentes por separado

Sea A una matriz simétrica de orden n. Construimos la gráfica asociada a A, G^A, de la siguiente manera:

Sus vértices son 1, 2,..., n y sus aristas son parejas no ordenadas tomadas del conjunto de vértices; {i,j} es una arista de G^A si y solo si $a_{ij} \neq 0$, $i \neq j$

$$(i,j) \in A(G^A) \iff a_{ij} \neq 0,\ i \neq j$$

EJEMPLO

$$A = \begin{bmatrix} x & x & x & x & x \\ x & x & & & \\ x & & x & & x \\ x & & & x & x \\ x & & x & x & x \end{bmatrix}$$

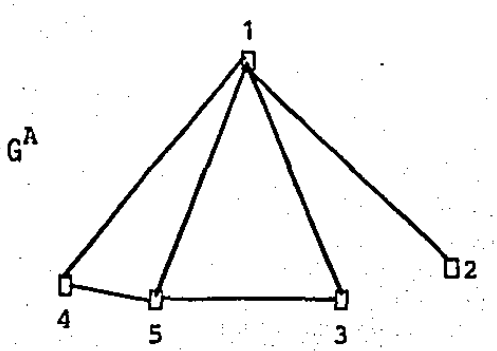


figura 2.3

Para nuestros fines, es de gran interés renumerar las hileras y las columnas de una matriz simétrica A. Es claro que esta operación no altera la gráfica correspondiente y unicamente permuta los nombres de los vértices.

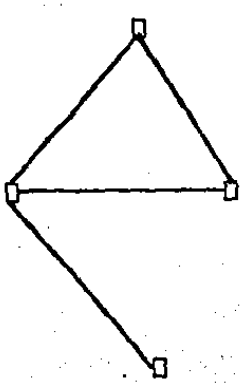
Definición

Sea G una gráfica con n vértices. Una *numeración* de G es una función biyectiva

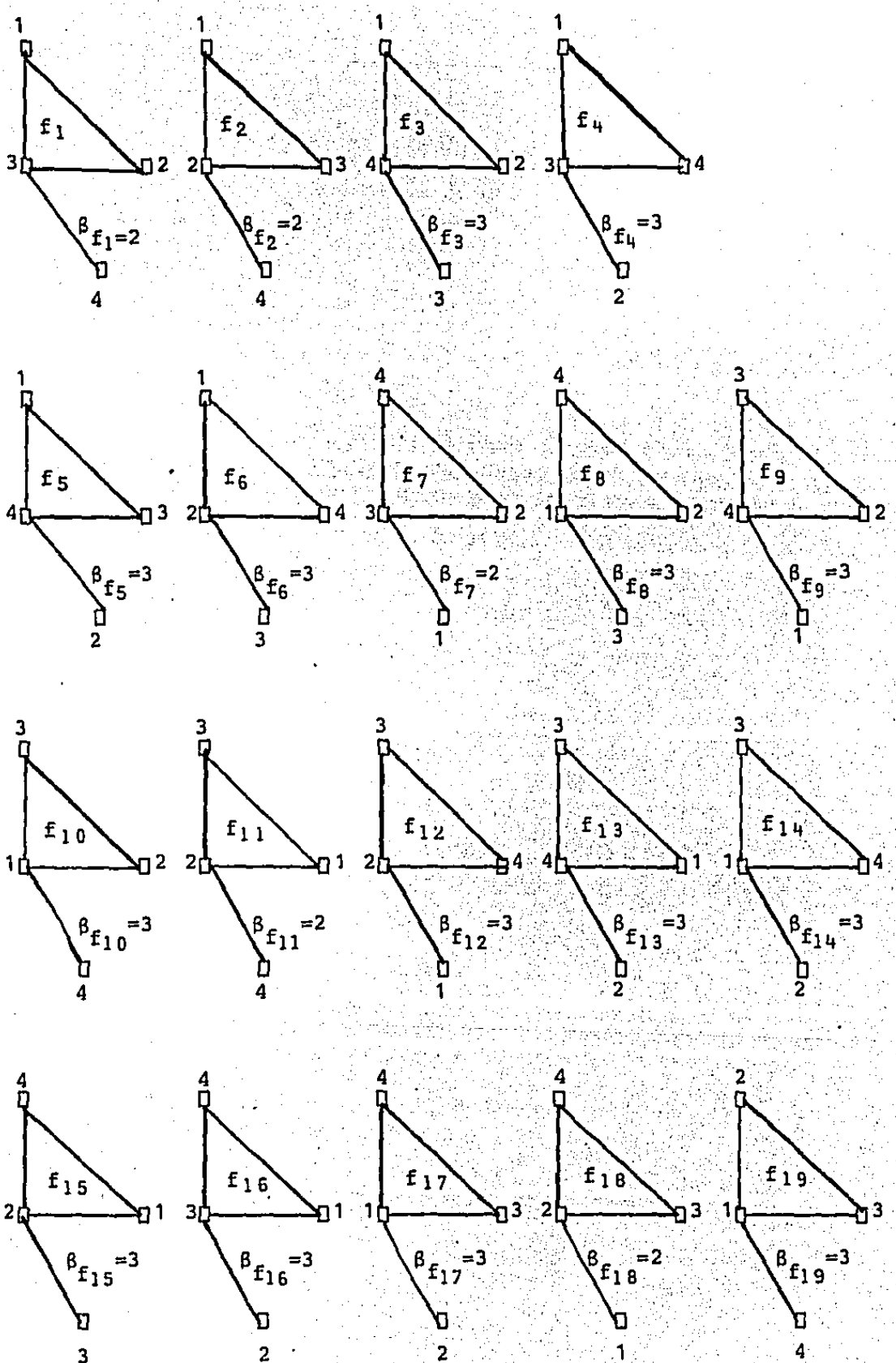
$$f: V(G) \longrightarrow \{1, \ldots, n\}$$

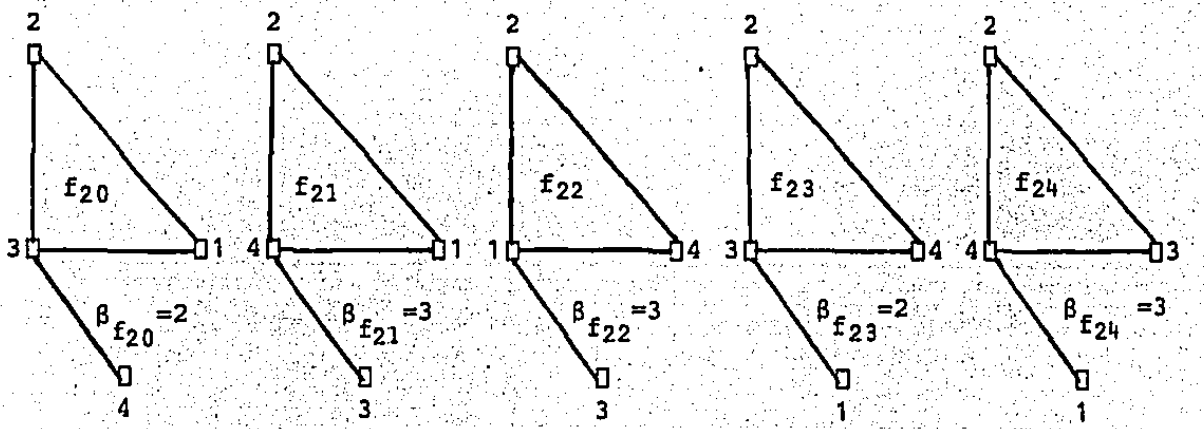
EJEMPLO

Sea G la gráfica



Las numeraciones posibles de G son 24 y están dadas por:


1
f1
3
2
βf1=2
4
1
f2
2
3
βf2=2
4
1
f3
4
2
βf3=3
3
1
f4
3
4
βf4=3
2
1
f5
4
3
βf5=3
2
1
f6
2
4
βf6=3
3
4
f7
3
2
βf7=2
1
4
f8
1
2
βf8=3
3
3
f9
4
2
βf9=3
1
3
f10
1
2
βf10=3
4
3
f11
2
1
βf11=2
4
3
f12
2
4
βf12=3
1
3
f13
4
1
βf13=3
2
3
f14
1
4
βf14=3
2
4
f15
2
1
βf15=3
3
4
f16
3
1
βf16=3
2
4
f17
1
3
βf17=3
2
4
f18
2
3
βf18=2
1
2
f19
1
3
βf19=3
4



Es claro que la correspondencia

$$A \longrightarrow G^A$$

no es uno a uno, pues para construir la gráfica asociada a A unicamente requerimos la sombra de esta:

$$\text{sombra } (A) = \{(i, j) \in \{1, \ldots, n\}^2 : a_{ij} \neq \phi\}$$

Dadas dos matrices de orden n, podemos construir la gráficas correspondientes, G^A y G^B. Es inmediato que

$$G^A = G^B \iff \text{sombra } (A) = \text{sombra } (B)$$

Observación

Estos conceptos se aplican también si las matrices no son simétricas. Sin embargo, la forma en que construimos la correspondencia

$$A \longrightarrow G^A \quad \text{es tal que,}$$

en ese caso,

$$G^A = G^B \iff \text{sombra } (A + A^t) = \text{sombra } (B + B^t)$$

En este punto conviene aclarar que hemos utilizado la siguiente convención:

$$\text{sombra } (A + B) := \text{sombra } A \cup \text{sombra } B$$

En otras palabras, las cancelaciones aritméticas no cuentan al calcular la sombra de una suma de matrices.

Conceptos como ancho de banda aplicados a matrices simétricas se traducen facilmente al lenguaje de las gráficas.

Definición

Sea G una gráfica y f una numeración de sus vértices. El *ancho de banda de* G *relativo a* f, $\beta_f(G)$, se define como

$$\beta_f(G) = \max\{|f(v_i) - f(v_j)|: \{v_i, v_j\} \in \varphi_G(A(G))\}$$

Definición

El ancho de banda de G, $\beta(G)$, se define como:

$$\beta(G) = \min_{f \in N_G} \beta_f(G)$$

donde N_G denota al conjunto de todas las numeraciones de G.

En el ejemplo anterior, $\beta(G) = 2$ y las numeraciones f_1, f_2, f_7, f_{11}, f_{12}, f_{18}, f_{20}, f_{23} son óptimas.

En general, si lo que se desea es reducir el ancho de banda $\beta_f(G)$ de G, se dirá de una numeración f^* que es *óptima* si

$$\beta_{f^*}(G) = \beta(G)$$

Reducir el ancho de banda de una gráfica es de gran interés, ya que existe una correspondencia entre matrices simétricas de orden n y gráficas de orden n, como lo describimos anteriormente.

Claramente se tiene que si A es una matriz y G^A su gráfica asociada, entonces

$$\beta(A) = \beta_I(G^A),$$

donde I es la numeración de G^A dada por

$$I(v_i) = i \text{ con } i = 1,2,\ldots,n$$

Proposición 2.

Sea A una matriz simétrica y G^A la gráfica correspondiente. Para cada numeración f de G^A, la matriz de permutaciones

$P_f = (P_{ij})$ dada por:

$$P_{ij} = \begin{cases} 1 \text{ si } f(v_j) = i \\ 0 \text{ si } f(v_j) \neq i \end{cases}$$

es tal que $\beta(P_f\, A\, P_f^t) = \beta_f(G^A)$.

El resultado es claro, pues la manera de definir P_f hace que la permutación de renglones y columnas de A sea la misma permutación de los vértices de G^A definida por la numeración f. Así pues, si queremos encontrar una matriz P de permutaciones tal que

$$\beta(PAP^t) < \beta(A)$$

basta con encontrar una numeración f de G^A tal que $\beta_f(G^A) < \beta_I(G^A)$.

He aqui el origen de nuestro interés por encontrar renumeraciones de los vértices de una gráfica que reduzcan su ancho de banda.

En la práctica no es frecuente que pueda lograrse una renumeración óptima, pero sí es factible dar renumeraciones subóptimas y efectivas. Una manera de encontrarlas es utilizando el algoritmo de E. Cuthill y Mc. Kee, que es una herramienta muy poderosa para dar renumeraciones buenas que reducen el ancho de banda. Existen algunos otros algoritmos como los que presentan Norman E. Gibbs, William G. Pool Jr. y Paul K. Stockemeyer en [ ]. Apoyados en el algoritmo de E. Cuthill y Mc. Kee y utilizando conceptos de la teoría de gráficas estos autores logran encontrar renumeraciones más eficaces.

II.2 REDUCCION DEL ANCHO DE BANDA [6,7,8,9]

En esta sección se dará una descripción del algoritmo de E. Cuthill y Mc. Kee para reducir el ancho de banda. También se incluye un ejemplo en el que se muestra el funcionamiento de éste. Para ello es importante introducir el

concepto de estructura de nivel de una gráfica.

Definición

Sea G una gráfica y $v \in V(G)$.

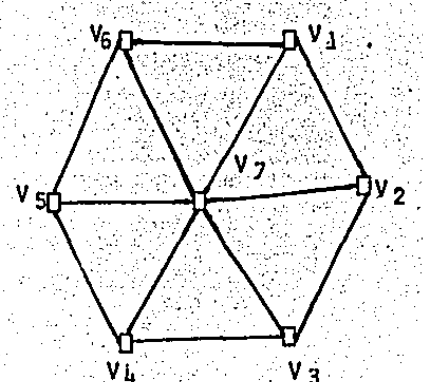
Una *vecindad* $V(v)$ consiste en el conjunto de vértices $u \in V(G)$ tales que v es adyacente a u. De otra forma

$$V(v) = Ady(v) = \{u | u \in V(G) \text{ y } \{u,v\} \in A(G) \text{ y } v \neq u\}.$$

Dado un subconjunto V' de vértices de G, definimos

$$Ady(V') = \bigcup_{v \in V'} Ady(v) - V'$$

En la gráfica siguiente



la vecindad de v_7 es: $V(v_7) = \{v_1,\ldots,v_6\}$

y si $V' = \{v_1, v_3\}$

$$Ady(V') = \{v_2, v_6, v_7, v_4\}$$

Definición

Dada un vértice $v \in V(G)$, la estructura de nivel cimentada en el vértice v, es una partición de $V(G)$

$$N(v) = \{N_0(v), N_1(v),\ldots, N_{e(v)}(v)\}$$

en donde

$$N_0(v) = \{v\}, \quad N_1(v) = Ady\,(N_0\{v\})$$

y

$$N_i(v) = Ady(N_{i-1}(v)) - N_{i-2}(v) \text{ con } i = 2,3,\ldots,e(v)$$

El entero $e(v)$ se llama la *profundidad* de la estructura $N(v)$; a $N_0(v),\ldots, N_{e(v)}(v)$ se les llaman *niveles* y al vértice v la *raiz*.

Al número de elmentos del nivel $N_i(v), a_i(N) := |N_i|$ se le denomina la *anchura* del nivel i. Para una estructura de nivel N dada con raiz en v,

$$a(N) := \max\{a_i : i = 1, \ldots, n\}$$

se define como *el ancho* (o la anchura) de dicha estructura de nivel.

DESCRIPCION DEL ALGORITMO

Supongamos que la gráfica es conexa, pues de otra forma el algoritmo se puede aplicar a cada una de las componentes de la gráfica.

A. Generar la estructura de nivel en cada vértice v cuyo grado sea menor o igual que

$$\max\{\min\{(gr_{max} + gr_{min})/2 \quad gr_{promedio} - 1\}, gr_{min}\},$$

en donde grado máximo significa, el grado más grande entre todos los vértices de la gráfica; de manera semejante se tiene el grado promedio y el grado mínimo.

B. Para cada estructura de nivel de anchura mínima generada en A, numerar la gráfica nivel por nivel, con enteros consecutivos, de la siguiente manera:

a) El vértice donde se cimentó la estructura de nivel, asignarle el número 1.

b) Para los niveles sucesivos empezando por el nivel dos, numerar los vértices adyacentes al vértice 1, en orden creciente de los grados. Los empates se pueden romper de manera arbitraria. Los vértices restantes adyacentes al vértice numerado con el menor entero positivo del nivel anterior se numeran después, en orden creciente de los grados. Se continua con este proceso en todos los vértices del nivel en cuestión hasta que todos estén numerados; se procede luego a numerar los del nivel siguiente. El proceso termina cuando todos los vértices han sido numerados.

C. Para cada numeración f producida en Bb calcular el ancho de banda $\beta f(G)$ correspondiente. Seleccionar la numeración que produce el ancho de banda mínimo.

El ejemplo siguiente muestra el funcionamiento del algoritmo.

Sean A una matriz simétrica de orden 8 y G^A la gráfica asociada figura 2.4.

$$\begin{bmatrix} x & x & x & . & . & x & x & x \\ x & x & . & . & . & . & x & x \\ x & . & x & . & . & x & . & . \\ . & . & . & x & . & x & . & x \\ . & . & . & . & x & . & x & . \\ x & . & x & . & x & x & . & x \\ x & x & . & . & x & . & x & . \\ x & x & . & x & . & x & . & x \end{bmatrix}$$

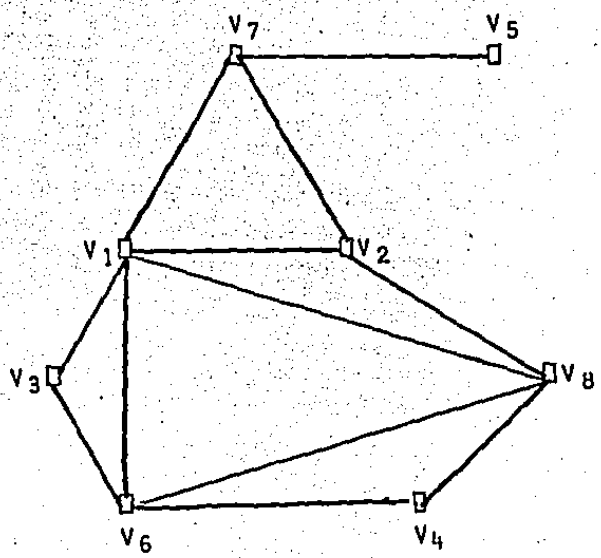
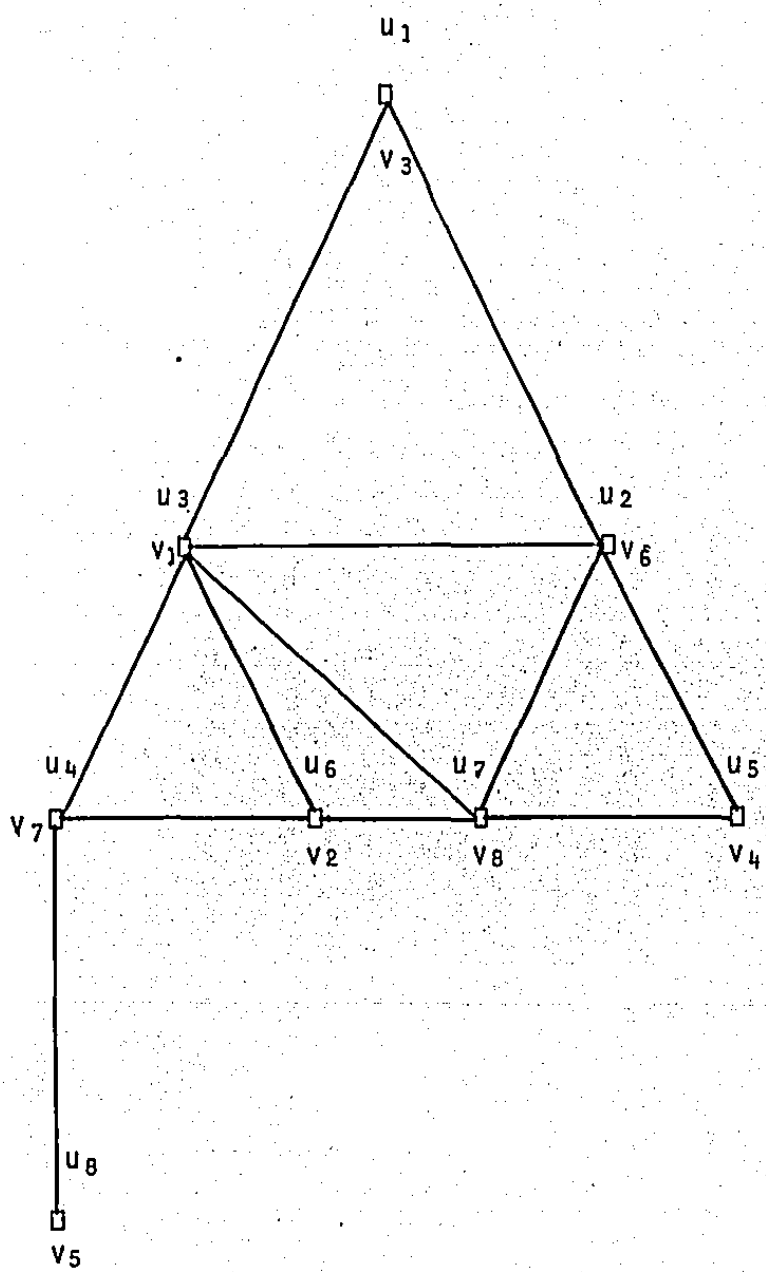


figura 2.4.

Los vértices de grado bajo son v_5, v_4, v_3. Las estructuras de nivel cimentadas en los vértices v_3, v_4, v_5 se muestran en las figuras 2.5, 2.6 y 2.7 respectivamente.



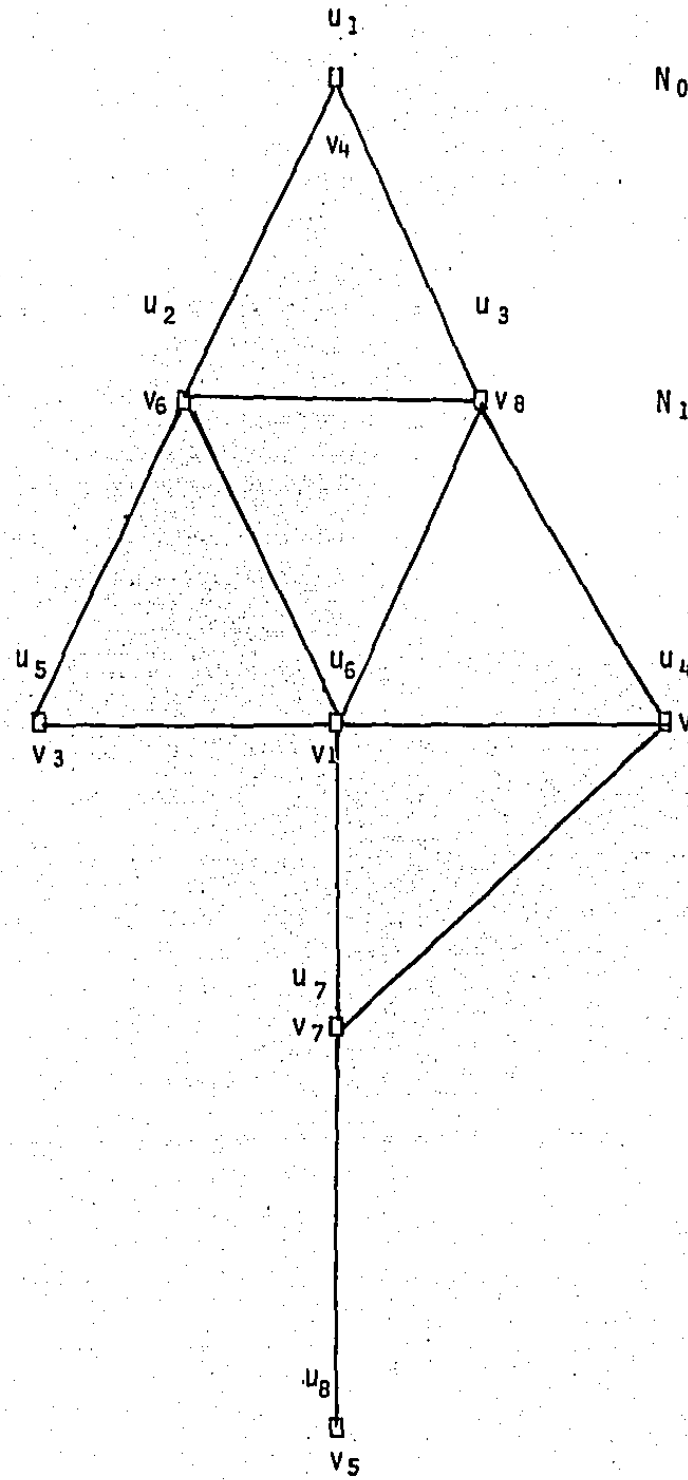
$N_0(v_3) = \{v_3\}$

$N_1(v_3) = \{v_1, v_6\}$

$N_2(v_3) = \{v_7, v_2, v_8, v_4\}$

$N_3(v_3) = \{v_5\}$

Figura 2.5



$N_0(v_4) = \{v_4\}$

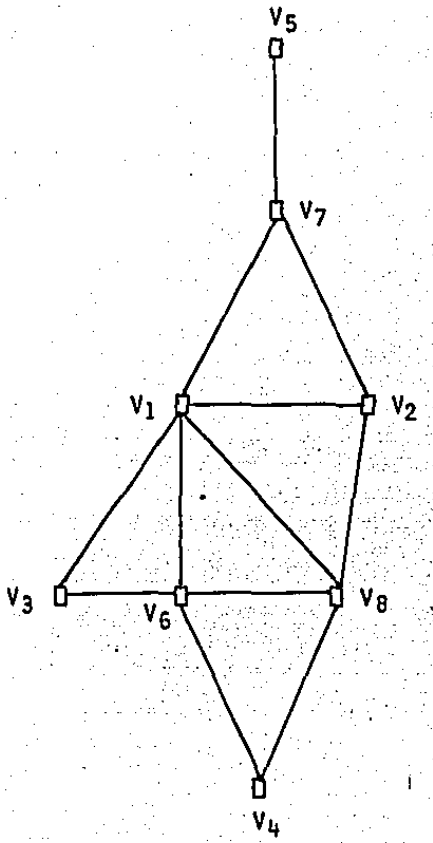
$N_1(v_4) = \{v_6, v_8\}$

$N_2(v_4) = \{v_3, v_1, v_2\}$

$N_3(v_4) = \{v_7\}$

$N_4(v_4) = \{v_5\}$

Figura 2.6



$N_0(v_5) = \{v_5\}$

$N_1(v_5) = \{v_7\}$

$N_2(v_5) = \{v_1, v_2\}$

$N_3(v_5) = \{v_3, v_6, v_8\}$

$N_4(v_5) = \{v_4\}$

Figura 2.7

La numeración obtenida por el algoritmo de E. Cuthill y Mc. Kee., está dada por las u_i, $i = 1, 2..., 8$, obsérvese que los anchos de banda producidos por estas numeraciones tomando como raíces a v_3, v_4 y v_5 son de 5, 4 y 3 respectivamente, por lo que podemos elegir la numeración que tiene a v_5 como vértice inicial. La matriz que se obtiene se muestra en la figura 2.8.

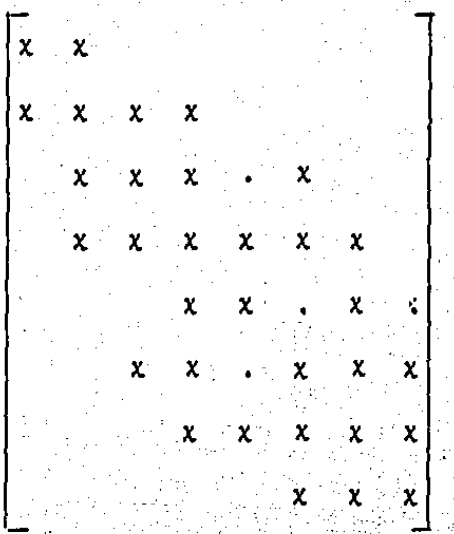


figura 2.8

Observemos que esta renumeración ha reducido el ancho de banda de la matriz dada, de 7 que era a 3, con las ventajas que de ello se desprenden en lo que toca a almacenar A y a realizar eliminación gaussiana sobre ella.

II.3 REDUCCION DEL PERFIL

La banda de una matriz, se define como

$$BANDA(A) = \{a_{ij} \mid 0 < i-j \leq \beta(A)\}.$$

Consideremos una matriz A la cual ha sido permutada para reducir su ancho de banda. El conjunto de elementos que se

encuentran en la Banda (A) contiene muchos elementos iguales a cero. Pensaríamos en quitarlos de alguna manera para facilitar, entre otras cosas, la solución del sistema asociado a $Ax = b$; esto es posible en cierta medida, utilizando para cada renglón i su ancho de banda local, de tal manera que todos los elementos iguales a cero que se encuentran a la izquierda del primer elemento distinto de cero de cada renglón quedan fuera de este nuevo conjunto. Definamos a este conjunto de elementos como la *envolvente* de A; de una manera más precisa:

$$ENV(A) = \{a_{ij} \mid 0 < i - j \leq \beta_i(A)\}$$

Como podemos observar la envolvente de A es un subconjunto de la banda de A. También debemos hacer notar que en algunos casos, resulta más conveniente reducir la cardinalidad de la envolvente o *perfil*, pues ésta contiene tantos o menos elementos que la banda.

En esta parte nos referimos al algoritmo R.C.M (el algoritmo con inversión de E. Cuthill y Mc. Kee), que basa gran parte de su desarrollo en el algoritmo descrito en la sección 2 de este capítulo y cuyas modificaciones fueron hechas por A. George [17].

El siguiente resultado, caracteriza a la envolvente

de una matriz A por medio de una propiedad en la gráfica asociada G^A.

Sea A una matriz simétrica, definida positivamente de orden n. Para el i-ésimo renglón de A definimos

$$f_i(A) = \min \{j | a_{ij} \neq 0\}$$

para $i = 1, 2, \ldots, n$

El número $f_i(A)$, es el índice de la columna en la que se encuentra el primer elemento distinto de cero del renglon i.

Teorema

Para $i<j$, $a_{ij} \in ENV(A)$ si y solo si $v_j \in Ady(\{v_1, v_2, \ldots, v_i\})$.

Demostración:

Supongamos que $v_j \in Ady(\{v_1, v_2, \ldots, v_i\})$,

entonces $a_{jk} \neq 0$ para alguna $k \leq i$. En consecuencia

$$f_j(A) \leq i \text{ y } a_{ij} \in ENV(A).$$

Reciprocamente, si $f_j(A) \leq i < j$, existe $k \leq i$ para la cual $a_{jk} \neq 0$; de esta forma los nodos j y k son adyacentes en la gráfica G^A, y entonces se tiene que

$$v_j \in Ady\ (\{v_1, v_2, \ldots, v_i\}).$$

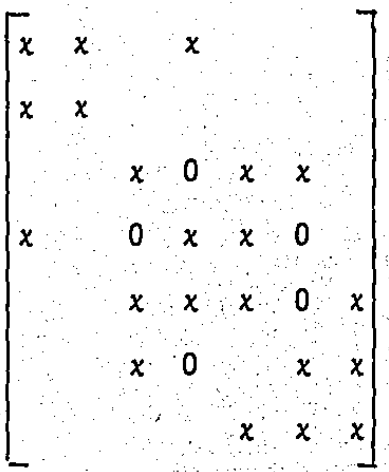
El siguiente resultado es un corolario del teorema anterior. Para demostrarlo introducimos la definición de ancho de frente de una matriz.

Definición

Para una matriz A, el i-ésimo ancho de frente de A se define como

$$w_i(A) = |\{k | k > i \ \text{ y } \ a_{k\ell} \neq 0 \ \text{ para alguna } \ \ell \leq i\}|$$

Antes de demostrar el corolario antes mencionado, ilustraremos los conceptos que se acaban de definir mediante la figura 2.9



i	$w_i(A)$	$f_i(A)$
1	2	1
2	1	1
3	3	3
4	2	1
5	2	3
6	1	3
7	0	5

figura 2.9

Corolario

Para $i = 1, 2, \ldots, n$ $w_i(A) = |Ady(\{v_1, v_2, \ldots, v_i\})|$

En efecto, de la definición de w_i se tiene que

$$w_i(A) = |\{j > i | \{i, j\} \in ENV(A)\}|$$

el resultado se sigue del teorema.

Definición

En una gráfica G, al conjunto $Ady(\{v_1, v_2, \ldots, v_i\})$ se le llama el *i-ésimo frente de la gráfica* y a su tamaño el i-ésimo ancho de frente.

DESCRIPCION DEL R.C.M. [17, 25]

A.- Determinar un nodo inicial v y asignarle la etiqueta $v_1 = v$

B.- Para $i = 1, 2, \ldots, n$, encontrar todos los vecinos del nodo v_i que no estén numerados y numerarlos en orden creciente de los grados.

C.- El ordenamiento inverso de E. Cuthill Mc. Kee está dado por $u_1, u_2, \ldots, u_n$ donde

$$u_i = v_{n-i+1}, \quad \text{para } i = 1, 2, \ldots, n.$$

En este algoritmo esta contenido el de E. Cuthill Mc. Kee; podriamos decir que los dos primeros pasos del R.C.M. es el algoritmo para reducir el ancho de banda, pues la selección de un nodo inicial, involucra el uso de estructuras de nivel.

Después de dar un ejemplo para ilustrar el R.C.M. haremos mención de algoritmos que seleccionan vértices iniciales.

	a	b	c	d	e	f	g	h	i
a	x	.	.	.	.	x	.	.	x
b	.	x	.	.	.	.	.	x	.
c	.	.	x	.	x	.	.	.	x
d	.	.	.	x	.	x	.	.	x
e			x	.	x	.	x	.	.
f	x	.	.	x	.	x			
g		x	.	.	x	.	x	x	
h		x	.	.	.	.	.	x	
i	x	.	x	x	.	.	.	.	x

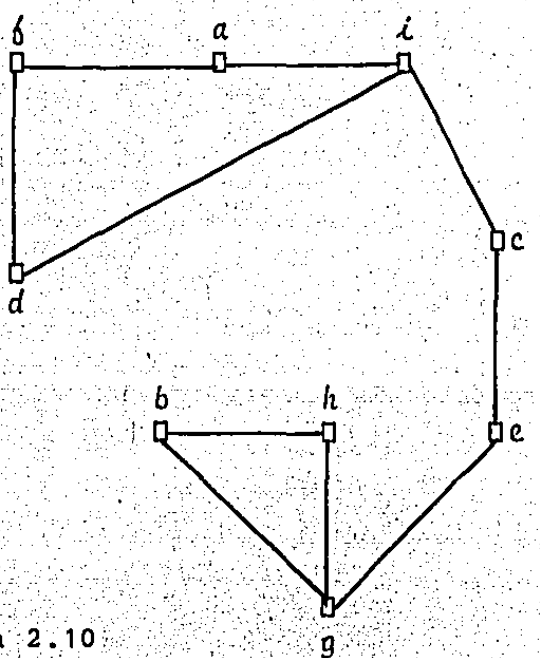


Figura 2.10

La figura 2.10 muestra una matriz y su gráfica asociada; la tabla siguiente ejemplifica la manera de numerar los vértices según el R.C.M. y en las figuras 2.11 y 2.12 se puede observar los cambios en el matriz al dar las nuevas numeraciones a la gráfica.

i	nodo V_i	vecinos sin numerar en orden creciente	numeración invertida
1	g	b, e, h	9
2	b	-	8
3	e	c	7
4	h	-	6
5	c	i	5
6	i	a, d	4
7	a	f	3
8	d	-	2
9	f	-	1

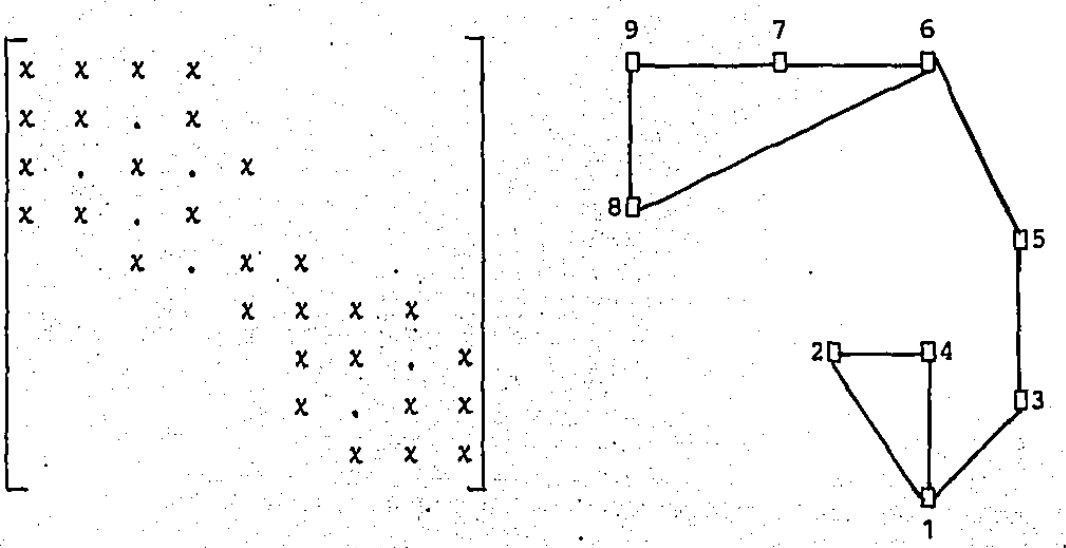


figura 2.11

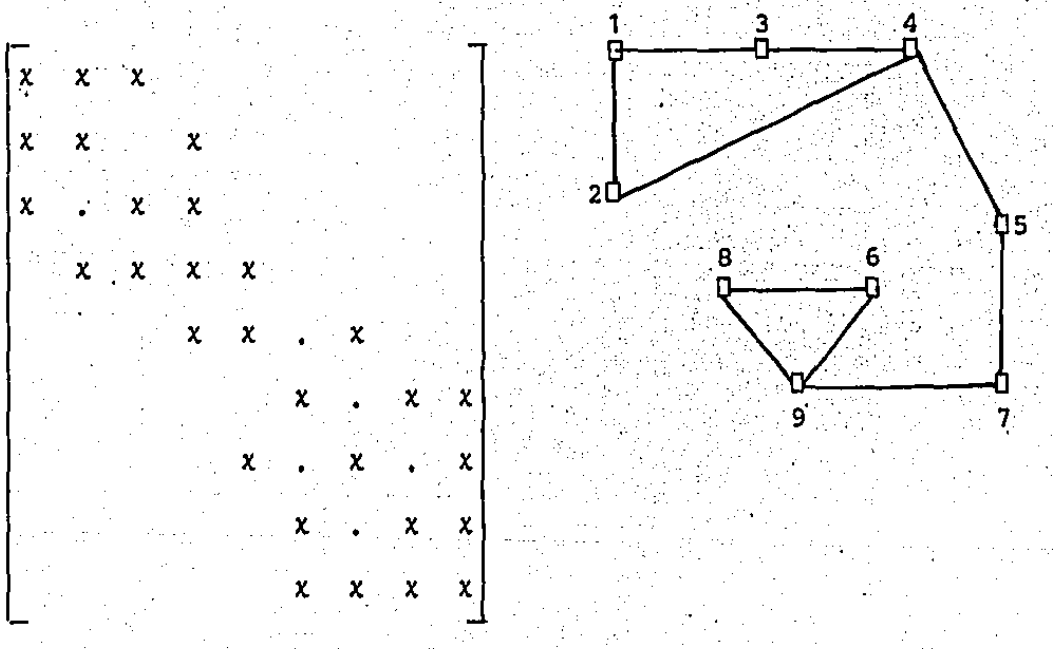


figura 2.12

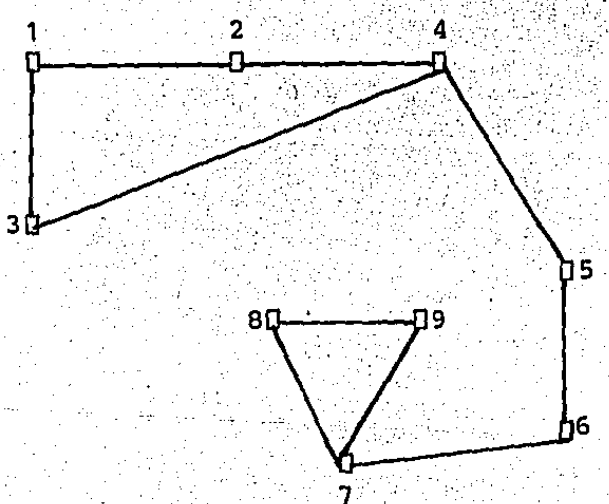
En este ejemplo se puede ver claramente que los perfiles de las matrices de las figuras 2.10, 2.11 y 2.12 son de 26, 14 y 13 respectivamente mientras que el ancho de banda respectivo es de 8, 3 y 3.

Consideremos ahora a f como nodo inicial:

i	nodo v_i	vecinos sin numerar en orden creciente
1	f	a, d
2	a	i
3	d	-
4	i	c
5	c	ℓ
6	ℓ	g
7	g	b, h
8	b	-
9	h	-

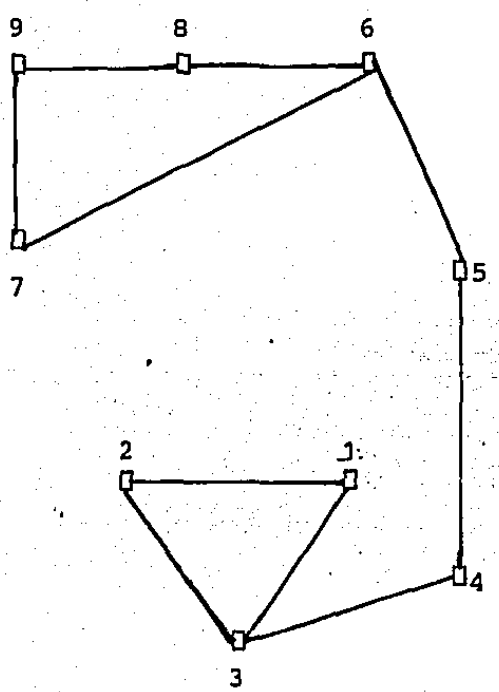
	1	2	3	4	5	6	7	8	9
1	x	x	x						
2	x	x	.	x					
3	x	.	x	x					
4		x	x	x	x				
5				x	x	x			
6					x	x	x		
7						x	x	x	x
8							x	x	x
9							x	x	x

Matriz después del paso B.



Gráfica con la nueva numeración

Figura 2.13 (a)



$$\begin{bmatrix}
x & x & x & & & & & & & \\
x & x & x & & & & & & & \\
x & x & x & x & & & & & & \\
 & & x & x & x & & & & & \\
 & & & x & x & x & & & & \\
 & & & & x & x & x & x & & \\
 & & & & & x & x & & x & \\
 & & & & & x & . & x & x & \\
 & & & & & & x & x & x &
\end{bmatrix}$$

La gráfica y la matriz al invertir la numeración mediante $U_i = U_n - i + 1$.

Figura 2.13 (b)

La efectividad de los algoritmos para ordenamientos depende en gran parte de la elección de un nodo inicial. En el ejemplo que acabamos de considerar podemos ver los cambios que resultan al numerar al nodo f con el número 1(figura 2.13 ,*a* y *b*)el perfil de la matriz resulta ser de 11. De esta observación se desprende la necesidad de tener algún criterio para seleccionar vértices iniciales, de tal manera que el utilizarlos nos garantice estar más cerca del perfil mínimo. Enseguida se definen algunos conceptos necesarios para describir el algoritmo que proporciona vértices iniciales.

La *longitud* de una trayectoria es el número de aristas que contiene. En general, para dos vértices dados en una gráfica hay más de una trayectoria que los conecta. La *distancia* entre dichos vértices se define como la mínima longitud de alguna trayectoria que los une. En otras palabras, para $v, w \in V(G)$, $d(v, w) = \ell$ si de todas las trayectorias que unen a v y w hay al menos una de longitud ℓ y ninguna de longitud menor.

La *excentricidad* de un nodo $v \in V(G)$ se define por

$$e(v) = \max \{d(v, u): u \in V(G)\}$$

El diámetro de una gráfica G está dado como:

$$\delta(G) = \max \{e(v) \mid v \in V(G)\}$$

$$= \max \{d(u, v) \mid u, v \in V(G)\}.$$

De un nodo $v \in V(G)$ se dice que es *periférico*, si su excentricidad es igual al diámetro, e.c.

$$e(v) = \delta(G).$$

En la figura 2.14 se ilustran los conceptos recién descritos.

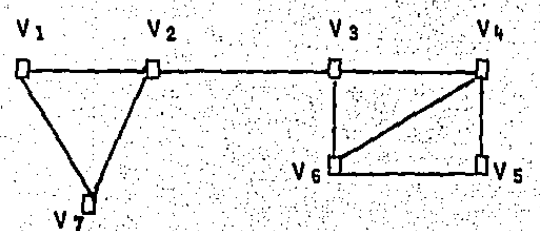


Figura 2.14

Una trayectoria T de v_1 a v_6 es:

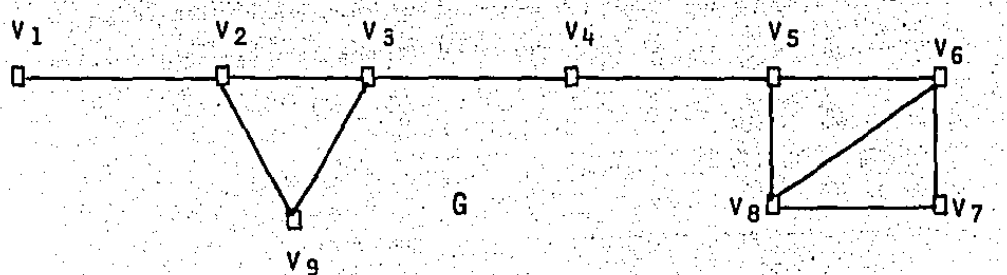
$T_{v_1-v_6} = v_6 = (v_1, v_2, v_3, v_4, v_6)$ y tiene longitud cuatro. De hecho, la distancia v_5 a v_7 es cuatro; la de v_2 a v_7 es uno, la excentricidad de v_3 es dos y el diámetro de la gráfica es cuatro; los nodos periféricos son v_1, v_5, v_7 y

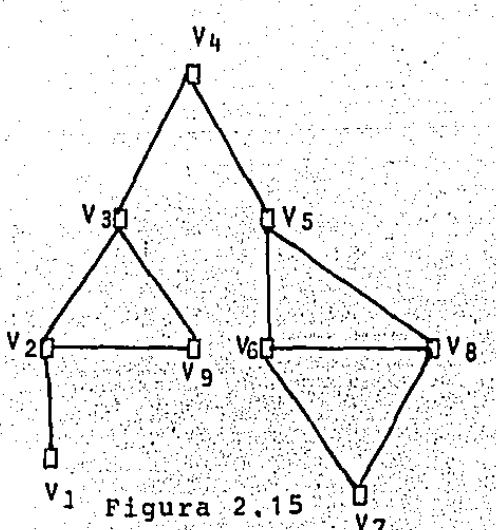
$$e(v_1) = e(v_5) = e(v_7) = \delta(G) = 4.$$

Si $N(v)$ es la estructura de nivel cimentada en v, a la excentricidad del nodo v se le conoce también como la *longitud* o *profundidad* de $N(v)$. La anchura de $N(v)$ queda expresada como

$$a(v) = \max \{|N_i(v)| : 0 \leq i \leq e(v)\}.$$

La figura 2.15 muestra una gráfica G y la estructura de nivel enraizada en v_4. Se puede observar que la excentricidad de v_4 es tres y la anchura de la estructura de nivel es cuatro.





$N_0 = \{v_4\}$

$N_1 = \{v_3, v_5\}$

$N_2 = \{v_2, v_9, v_6, v_8\}$

$N_3 = \{v_1, v_7\}$

Figura 2.15

Regresando a la tarea de encontrar nodos inciales para el algoritmo con inversión de E. Cuthill y Mc. Kee, se tiene la experiencia de que es necesario encontrar un par de nodos que se encuentren a distancia máxima o cerca de estarlo, afirmación que implica que los nodos deberán tener excentricidad grande. Buenos candidatos resultan ser los nodos periféricos, pero en general resulta tardado y costoso localizarlos, ya que para obtenerlos es necesario encontrar para cada vértice el número de trayectorias que lo contienen, calcular la longitud de cada una y seleccionar alguna de longitud máxima, después elegir de entre ellas alguna que coincida con el diámetro de la gráfica. Estas consideraciones nos hacen pensar que en una gráfica en donde el número de nodos es grande, el número de iteraciones es elevado. Con respecto a esto diremos que el mejor algoritmo que encuentra nodos periféricos es del orden de $O(|v||A|)$, Smyth [V7] y que en la mayoría de las aplicaciones de éste en matrices ralas esta cota llega a ser $O(|v|^2)$. Desde luego que la idea de encontrar nodos periféricos resulta ser muy atractiva y la razón para ello es convincente, pues al generar una estructura de nivel en un nodo periférico, el número de niveles es mayor que el número que se obtiene si el vértice no lo es. Entre más niveles se tengan, la cardinalidad de cada uno de ellos disminuye, trayendo consigo la reducción del ancho de banda de la gráfica y consecuentemente el de la matriz.

En el algoritmo RCM se emplearán nodos de excentridad alta como nodos iniciales; nos referiremos a ellos como nodos *pseudoperiféricos*. El algoritmo encuentra nodos pseudoperiféricos, es heurístico y ha demostrado que solamente en situaciones especiales los nodos periféricos resultan ser mejores que los encontrados por él. Este es debido a Gibbs et al. [18]. [26]

· ALGORITMO PARA SELECCIONAR NODOS INICIALES

A. Escoger un nodo arbitrario $v \in V$.

B. Generar la estructura de nivel cimentada en v:

$$N(v) = \{N_0(v), N_1(v), \ldots Ne_{(v)}(v)\}.$$

C. Elegir un nodo u de grado mínimo en el último nivel de la estructura

D. Generar la estructura de nivel enraizada en u

$$N(u) = \{N_0(u), N_1(u), \ldots, Ne_{(u)}(u)\}.$$

b) Si $e(u) > e(v)$ asignarle a $v = u$ y regresar a C.

E. El nodo u es un nodo pseudoperiférico

EJEMPLO

Sea G la gráfica siguiente

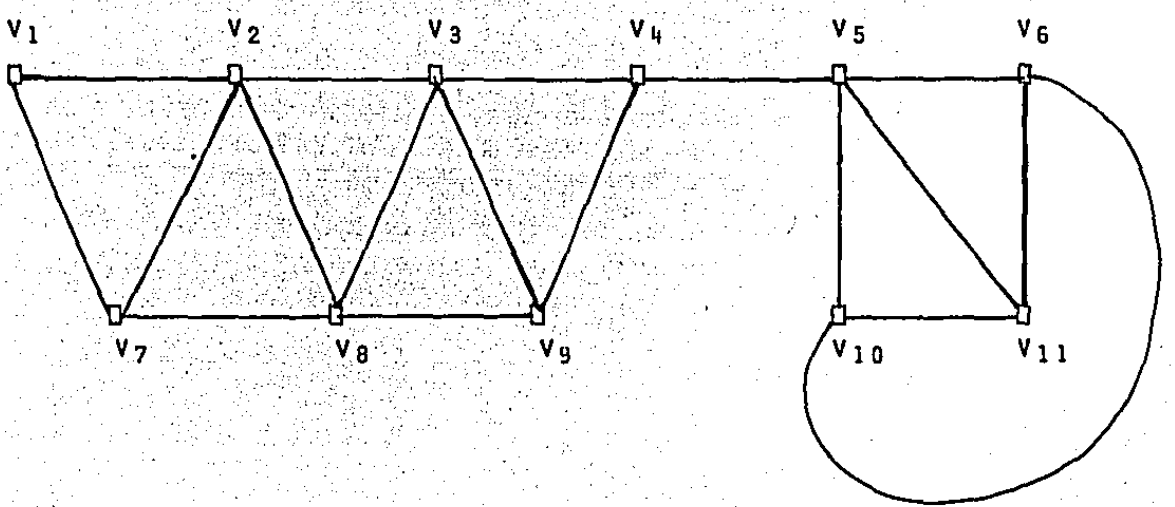
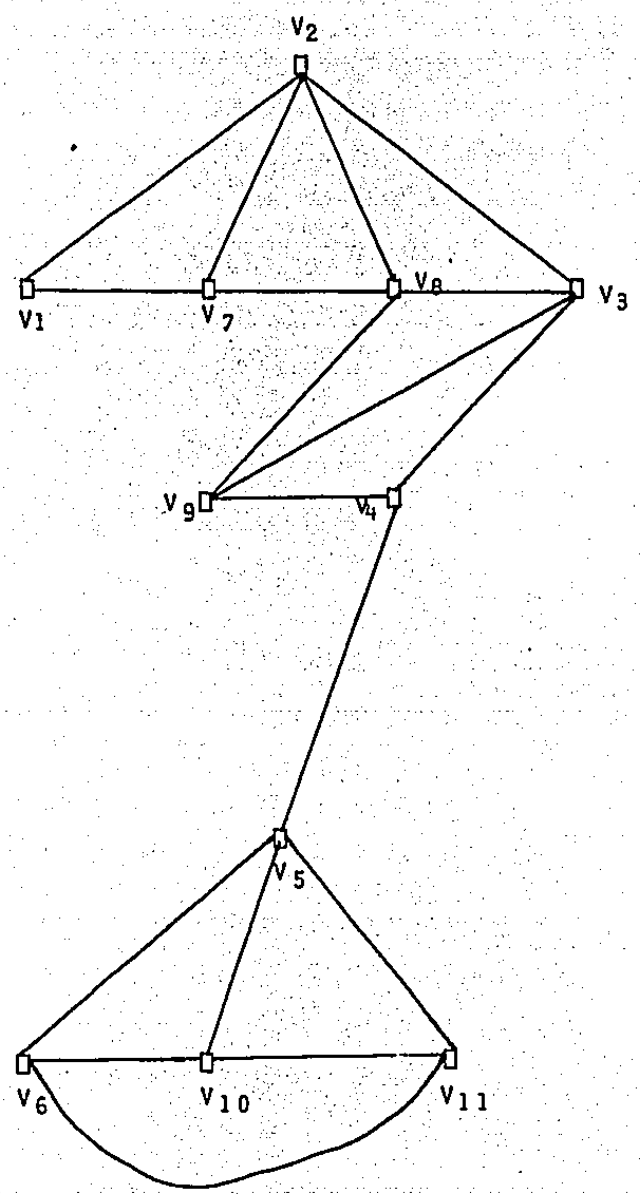


Figura 2.16

A. Tómese V_2 como nodo inicial $v = v_2$

B. Estructura de nivel cimentada en v_2



$N_0(v_2) = \{v_2\}$

$N_1(v_2) = \{v_1, v_7, v_8, v_3\}$

$N_2(v_2) = \{v_9, v_4\}$

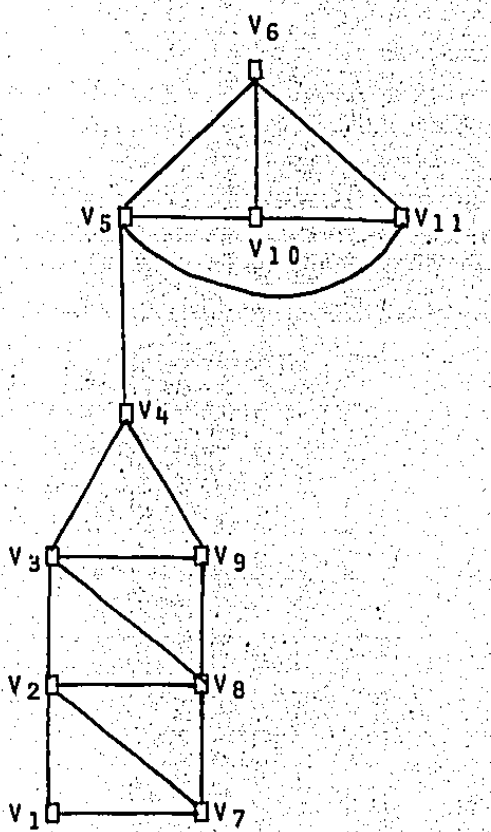
$N_3(v_2) = \{v_5\}$

$N_4(v_2) = \{v_6, v_{10}, v_{11}\}$

Figura 2.17

C. v_6 tiene grado mínimo entre los vértices del nivel más profundo.

D. *a)* Estructura de nivel cimentada en v_6



$N_0(v_6) = \{v_6\}$

$N_1(v_6) = \{v_5, v_{10}, v_{11}\}$

$N_2(v_6) = \{v_4\}$

$N_3(v_6) = \{v_3, v_9\}$

$N_4(v_6) = \{v_2, v_8\}$

$N_5(v_6) = \{v_1, v_7\}$.

Figura 2.18

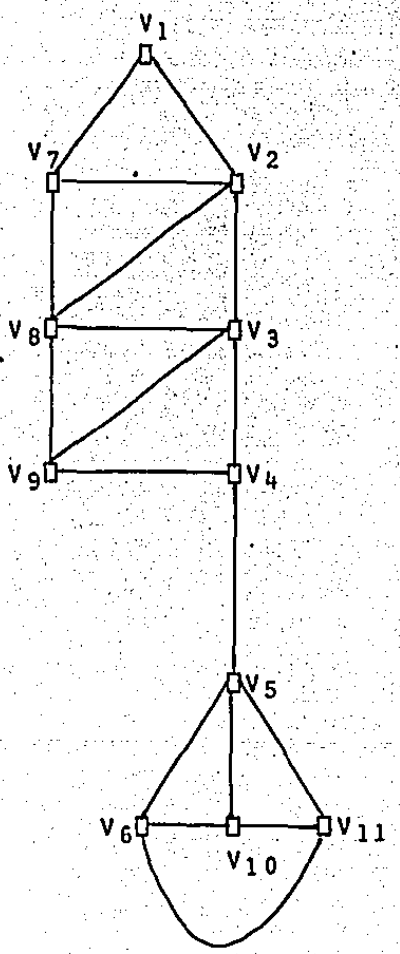
b) En este caso $e(v_6) = 5$ y $e(v_2) = 4$

por lo tanto $e(v_6) > e(v_2)$

y de aquí que hagamos $v = v_6$ y regresamos al paso C

C. Escoger un vértice del nivel $N_{e(v_6)}(v_6)$ que tenga grado mínimo; sea éste v_1.

D. Generar la estructura de nivel cimentada en v_1



$N_0(v_1) = \{v_1\}$

$N_1(v_1) = \{v_7, v_2\}$

$N_2(v_1) = \{v_8, v_3\}$

$N_3(v_1) = \{v_9, v_4\}$

$N_4(v_1) = \{v_5\}$

$N_5(v_1) = \{v_6, v_{10}, v_{11}\}$

Figura 2.19

Como $e(v_1) = e(v_6) = 5$, se tiene que:

E) v_1 es un nodo pseudoperiférico. Por lo tanto el algoritmo termina.

En las figuras 2.10, 2.11 y 2.12 con las que ilustramos el algoritmo con inversión, hicimos notar la necesidad de tener un criterio que nos permitiera obtener vértices iniciales; ahora que ya tenemos una forma de encontrarlos, haremos una descripción de éste y de sus efectos en la gráfica y la matriz, utilizando los criterios para la elección de nodos pseudoperiféricos.

Consideremos la matriz siguiente y su gráfica asociada (figura 2.20). Observemos que la gráfica es la misma que la de la figura 2.16. De aquí que podemos utilizar el vértice v_1 como nodo inicial, ya que éste fué elegido con el criterio que nos permite encontrar nodos pseudoperiféricos. La figura 2.19 muestra la estructura de nivel cimentada en v_1, y en la figura 2.21 se observa la gráfica con la nueva numeración y la matriz correspondiente. En la figura 2.22 está la gráfica renumerada invirtiendo el orden, junto con la matriz resultante.

	1	2	3	4	5	6	7	8	9	10	11
1	x	x					x				
2	x	x	x				x	x			
3		x	x	x				x	x		
4			x	x	x				x		
5				x	x	x				x	x
6					x	x				x	x
7	x	x	.	.	.	.	x	x			
8		x	x	.	.	.	x	x	x		
9			x	x	.	.	.	x	x		
10					x	x	.	.	.	x	x
11					x	x	.	.	.	x	x

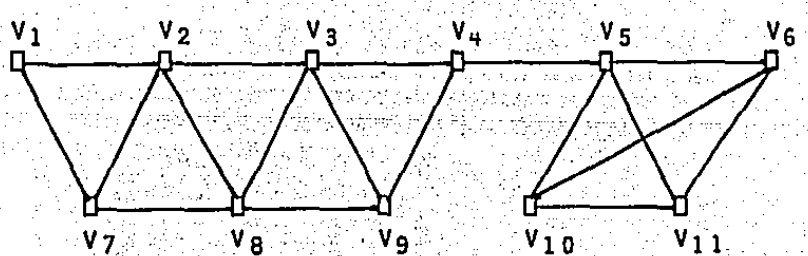
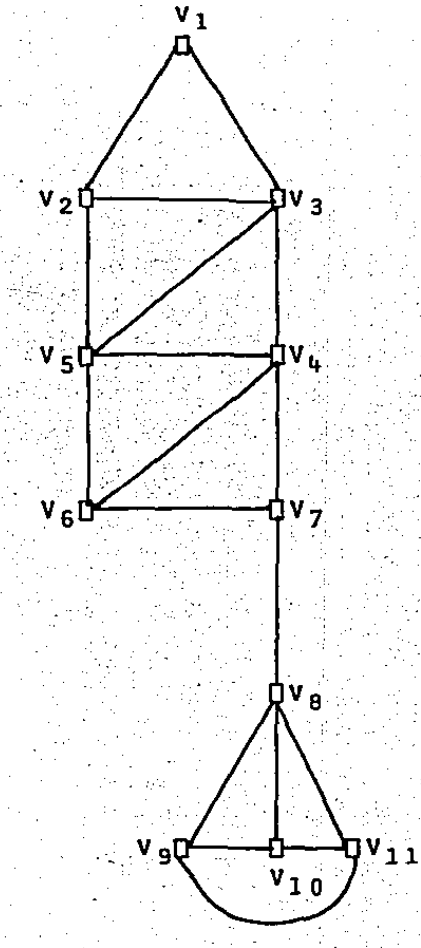
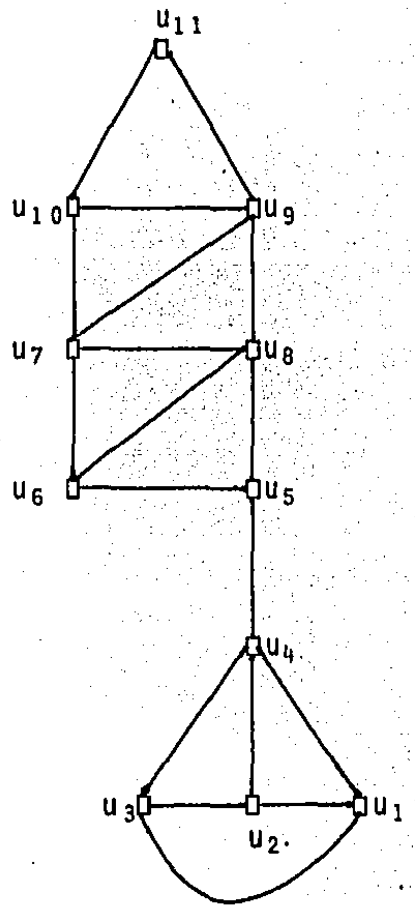


Figura 2.20



	1	2	3	4	5	6	7	8	9	10	11
1	x	x	x								
2	x	x	x	.	x						
3	x	x	x	x	x						
4		.	x	x	x	x	x				
5		x	x	x	x	x	.				
6				x	x	x	x				
7				x	.	x	x	x			
8							x	x	x	x	x
9								x	x	x	x
10								x	x	x	x
11								x	x	x	x

Figura 2.21



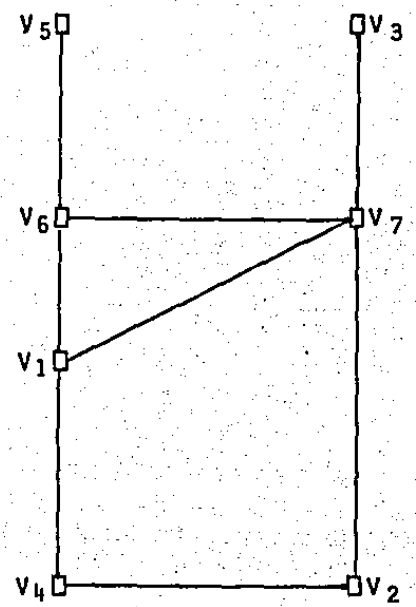
	1	2	3	4	5	6	7	8	9	10	11
1	x	x	x	x							
2	x	x	x	x							
3	x	x	x	x							
4	x	x	x	x	x						
5				x	x	x	.	x			
6					x	x	x	x			
7					.	x	x	x	x	x	
8					x	x	x	x	x		
9							x	x	x	x	x
10							x	.	x	x	x
11									x	x	x

Figura 2.22

Del ejemplo que acabamos de analizar podemos observar que, al aplicar el algoritmo de E. Cuthill y McKee a la matriz de la figura 2.20 obtenemos que el perfil es de 19 (figura 2.21) que es el mismo para la matriz de la figura 2.20 al aplicar el algoritmo R.C.M (figura 2.22).- Analicemos ahora un ejemplo en donde esto no pase; e.d. en donde los perfiles dados por estos ordenamientos sean diferentes.

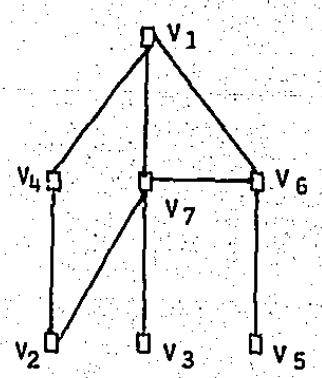
Sean A una matriz simétrica y G^A su gráfica asociada como se muestran enseguida:

$$A = \begin{bmatrix} x & . & . & x & . & x & x \\ . & x & . & x & . & . & x \\ . & . & x & . & . & . & x \\ x & x & . & x & . & . & . \\ . & . & . & . & x & x & . \\ x & . & . & . & x & x & x \\ x & x & x & . & . & x & x \end{bmatrix}$$



Elegimos a v_1 como nodo inicial y construyamos su estructura de nivel $N(v_1)$.

$N(v_1)$



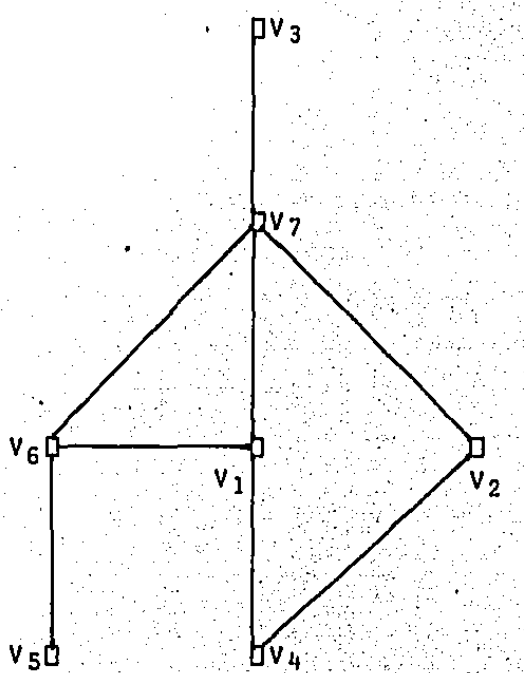
$N_0(v_1) = \{v_1\}$

$N_1(v_1) = \{v_4, v_7, v_6\}$

$N_2(v_1) = \{v_2, v_3, v_5\}$

Ahora elegimos un nodo con grado mínimo del último nivel. Sea este v_3 y construyamos su estructura $N(v_3)$.

$N(v_3)$



$N_0(v_3) = \{v_3\}$

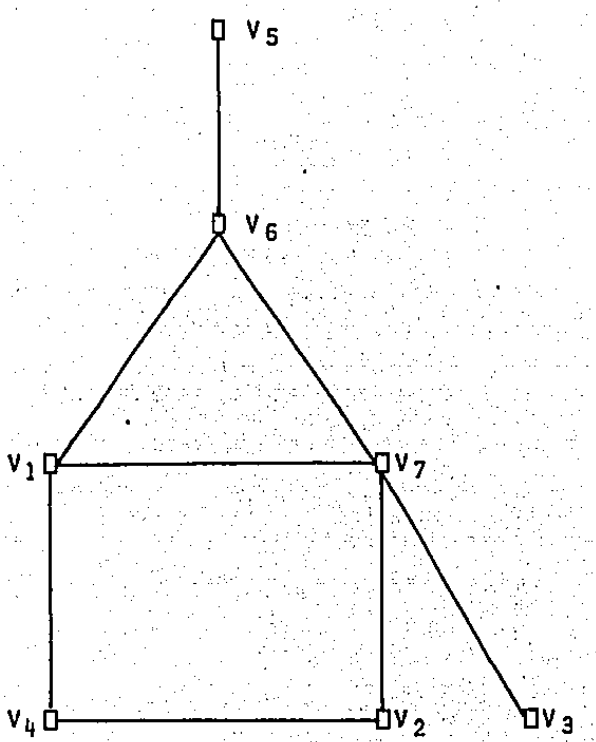
$N_1(v_3) = \{v_7\}$

$N_2(v_3) = \{v_6, v_1, v_2\}$

$N_3(v_3) = \{v_5, v_4\}$

Como la exentricidad de v_3 es 3 y de v_1 es dos procedemos a elegir un nodo de grado mínimo de $N_3(v_3)$ y generar su estructura de nivel. En este caso v_5 tiene grado menor que v_4 por lo que su estructura $N(v_5)$ queda como:

$N(v_5)$



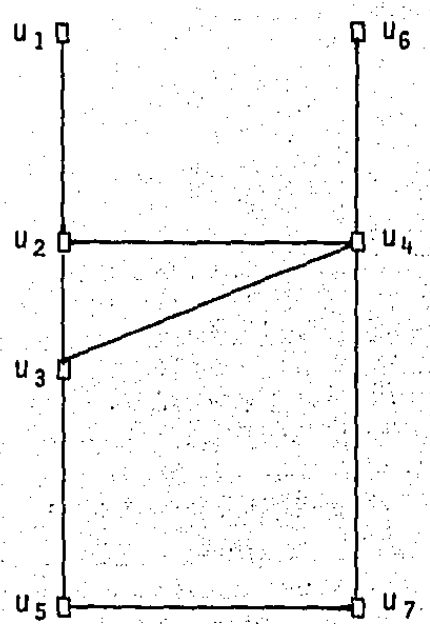
$N_0(v_5) = \{v_5\}$

$N_1(v_5) = \{v_6\}$

$N_2(v_5) = \{v_1, v_7\}$

$N_3(v_5) = \{v_4, v_2, v_3\}$

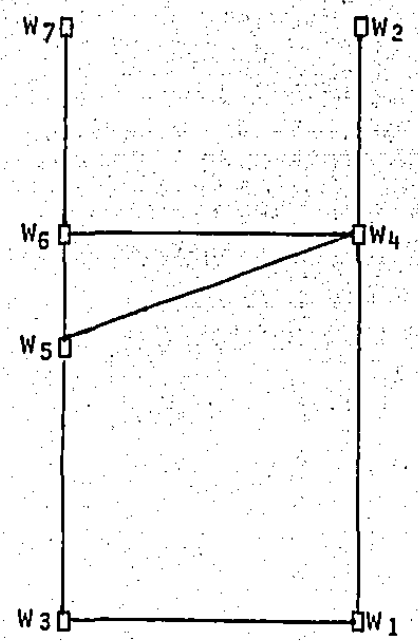
La exentricidad de v_5 y de v_3 es tres por lo que v_5 y v_3 son nodos pseudoperiféricos y de aquí que ya tengamos nodos iniciales, que nos den un mejor ordenamiento al aplicar el algoritmo de E. Cuthill y McKee. Demos la numeración a G^A obtenida por este utilizando como nodo inicial a v_5. La nueva numeración para la gráfica está dada por $u_1, u_2, \ldots, u_7$.



y esta permutación transforma a la matriz A en la matriz siguiente

$$A' = \begin{bmatrix} x & x & . & . & . & . & . \\ x & x & x & x & . & . & . \\ . & x & x & x & x & . & . \\ . & x & x & x & . & x & x \\ . & . & x & . & x & . & x \\ . & . & . & x & . & x & . \\ . & . & . & x & x & . & x \end{bmatrix}$$

Ahora demos a la gráfica el ordenamiento obtenido por R.C.M. Sea este $w_1, w_2, \ldots, w_1$. La gráfica con esta numeración y la matriz A reordenada son, como se muestra a continuación:



$$A'' = \begin{bmatrix} x & . & x & x & . & . & . \\ . & x & . & x & . & . & . \\ x & . & x & . & x & . & . \\ x & x & . & x & x & x & . \\ . & . & x & x & x & x & . \\ . & . & . & x & x & x & x \\ . & . & . & . & . & x & x \end{bmatrix}$$

De este ejemplo podemos observar que el perfil de A' es de 11; mientras que el de A'' es de 10. Recordemos que para la matriz de la figura 2.20, los perfiles coinciden bajo los dos ordenamientos; CM y RCM. En general se tiene que al reordenar una matriz mediante el primero de estos, la envolvente de la parte superior resulta tener menos o tantos elementos que la envolvente inferior. La permutación dada por $y_i = x_{n-i+1}$ transforma la envolvente superior en la envolvente inferior y viceversa. De aquí que en muchos casos este ordenamiento reduzca el perfil. Todas estas ideas estan contenidas en la definición y teorema siguientes.

Definición

La *envolvente superior* de una matriz A, es el conjunto $ENV_s(A) = \{a_{ij}: 0 < j - i \leq \beta_s^i(A)\}$

donde $\beta_s^i = f_s^i - i$, $f_s^i = \max \{a_{ij}: a_{ij} \neq 0, i \neq j\}$

para $i, j \in \{1,2,\ldots,n\}$.

Teorema

Sea A una matriz simétrica de orden n. Si A_0 es la matriz que se obtiene de A a través del ordenamiento de Cuthill-McKee, entonces:

$$|ENV_s(A_0)| \leq |ENV(A_0)|$$

Demostración:

Supongamos que $a_{ij} \in ENV_s(A_0)$ y $a_{ij} \neq 0$; entonces $a_{ji} \in ENV(A_0)$. Si $a_{ij} = 0$, sea N la estructura de nivel con la cual se construyó el ordenamiento. Como $a_{ij} \in ENV_s(A_0)$ existe j' tal que $a_{ij'} \neq 0$ y $j < j'$. Demostraremos que $a_{ji} \in ENV(A_0)$.

Dado que los vértices i y j' son adyacentes es posible encontrar un entero positivo ℓ tal que:

a) $i, j' \in N_\ell$

b) $i \in N_\ell$ y $j' \in N_{\ell+1}$ [1]

De (a) se tiene que $i < j < j'$, $j \in N_\ell$ y de aqui, es posible encontrar una $k \in N_{\ell+1}$ con las propiedades de que $k < i$ y $j \in V(k)$. El elemento $a_{jk} \neq 0$ por lo tanto $a_{ji} \in ENV(A)$.

De la parte (b) se concluye que si $j \in N_\ell$ entonces se reduce al caso (a). Cu ndo $j \in N_{\ell+1}$, en N_ℓ se encuentra un vértice k tal que $k < i$ y $j \in V(k)$; ya que si $i < k$, tendríamos un ordenamiento en los nodos que no fué el dado; por lo tanto $k < i < j$, $a_{jk} \neq 0$ y $a_{ji} \in ENV(A_0)$, con lo cual el teorema queda demostrado.

II. 4 REDUCCION DEL LLENADO

Sean A una matriz de orden n, simétrica, definida positiva y rala, L el factor de Cholesky de A y $F = L + L^t$. Definimos el *llenado* de A como el número de elementos de F distintos de cero que se encuentran en posiciones que en A son cero.

En esta última sección se hace en principio una descripción de la relación entre gráficas, matrices y eliminación gaussiana y después se muestra el algoritmo de deficiencia mínima debido a J. Rose [15,31] que tiene como objeto, reducir el llenado de la matriz A, renumerando los vértices de la gráfica asociada G^A. Para ello se construye una sucesión de gráficas $\{G_i\}_{i=0}^{n}$ en donde n + 1 es el

número de vértices de G^A. La gráfica $G_i = (V_i, A_i)$ se obtiene eliminando el vértice v_i de la gráfica G_{i-1} las aristas que lo tienen como extremo y añadiendo las aristas que se originan con la eliminación. La gráfica F y su gráfica G^F quedan determinadas mediante la sucesión $\{G_i\}_{i=0}^{n}$.

Ejemplo

Sea A_0 una matriz simétrica, definida positiva, rala y G^{A_0} su gráfica asociada, como se muestra en la figura 2.23. Las sucesiones $\{A_i\}_{i=0}^{4}$ y $\{G_i\}_{i=0}^{4}$ se obtienen según las ideas anteriores y se muestran en la figuras 2.24 (a) y 2.24 (b)

$$A_0 = \begin{bmatrix} x & x & . & x & . \\ x & x & . & . & x \\ . & . & x & x & x \\ x & . & x & x & x \\ . & x & x & x & x \end{bmatrix}$$

$G^{A_0} = G_0$

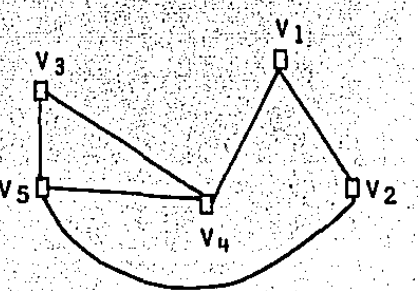


Figura 2.23

$$A_1 = \begin{bmatrix} x & x & x & x & . \\ . & \begin{bmatrix} x & . & (x) & x \\ . & x & x & x \\ x & x & x & x \\ x & x & x & x \end{bmatrix} \end{bmatrix}^{*}$$

G_1

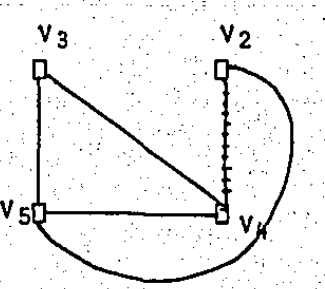
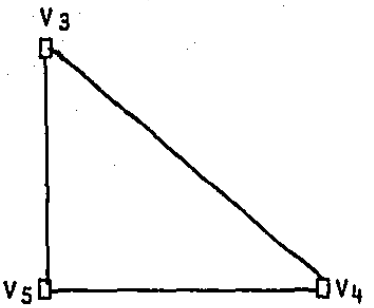


Figura 2.24 (a)

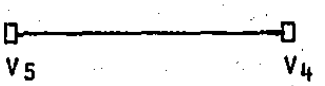
$$A_2 = \begin{bmatrix} x & x & . & x & . \\ . & x & . & x & x \\ . & . & x & x & x \\ . & . & x & x & x \\ . & . & x & x & x \end{bmatrix}$$

G_2



$$A_3 = \begin{bmatrix} x & x & . & x & . \\ . & x & . & x & x \\ . & . & x & x & x \\ . & . & . & x & x \\ . & . & . & x & x \end{bmatrix}$$

G_3



$$A_4 = \begin{bmatrix} x & x & . & x & . \\ . & x & . & x & x \\ . & . & x & x & x \\ . & . & . & x & x \\ . & . & . & . & x \end{bmatrix}$$

G_4

Figura 2.24 (b)

* A los elementos que pueden ser cero y que se originan durante la eliminación se denotan por (x). Las nuevas aristas se indican con líneas punteadas.

En el ejemplo que acabamos de considerar observemos que cada matriz A_i $i = 1, 2, 3, 4,$ puede ser construida de la matriz A_{i-1}. La relación entre ellas queda expresada en el siguiente párrafo [13]

Si Q_1, Q_2, Q_3, Q_4 son matrices con la propiedad de que:

$$Q_i A_{i-1} = A_i \quad \text{para} \quad i = 1, 2, 3, 4$$

y definimos

$$T = Q_4 Q_3 Q_2 Q_1,$$

entonces T es no singular, además de que

a) $$TA_0 = A_4 \quad y \quad A_0 = T^{-1}A_4$$

en donde

b) T^{-1} es triangular inferior y A_4 es triangular superior.

Conviene recordar que no siempre es cierto que $T^{-1} = L$ y $A_4 = L^t$.

De las sucesiones de matrices y gráficas que se muestra en la figura 2.24 se concluye que la matriz F y su gráfica asociada G^F, quedan como se muestran en la figura 2.25.

$$F=L+L^t=\begin{bmatrix} x & x & . & x & . \\ x & x & . & x & x \\ . & . & x & x & x \\ x & x & x & x & x \\ . & x & x & x & x \end{bmatrix}$$

G^F

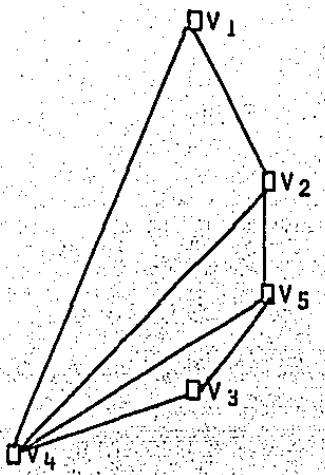


Figura 2.25

Una pregunta que surge al analizar el ejemplo anterior es la siguiente: ¿qué pasaría si eliminamos v_1, v_2, v_3, v_4 y v_5 en otro orden?. Convendría pues, tener una manera para elegir nodos que introduzcan el menor número de aristas durante el eliminación, ya que esto contribuye a disminuir el llenado de la matriz A: como sabemos, ésta se obtiene con las nuevas aristas que se forman durante el proceso de eliminación. Es claro que si v es un vértice y u,w son adyacentes a él, al eliminar v de la gráfica se obtienen nuevas aristas si u y w no son adyacentes. Para formalizar esta idea definimos la *deficiencia* de un vértice v como el número de parejas $\{u,w\}$ que tienen a v como vecino pero que u y w no lo son entre sí, más precisamente:

$$D(v) = |\{\{u,w\}: u, w \in V(v), w \notin V(u)\}|$$

De la figura 2.26 observamos que la deficiencia de v_1 es 3 y la de v_4 es 2

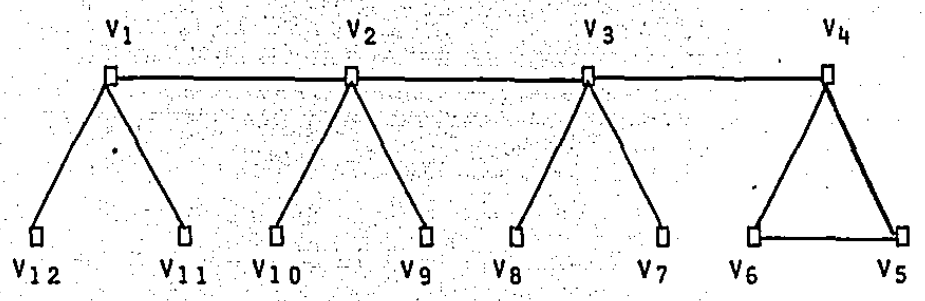


Figura 2.26

ALGORITMO DE ELIMINACION CON DEFICIENCIA MINIMA

1. Sea G_0 la gráfica correspondiente a la matriz A

2. Para $i = 1, \ldots, n$

   a) asígnese el número i al vértice en G_{i-1} que tiene mínima deficiencia

b) Elimínese dicho vértice, las aristas que lo contienen y anádanse las aristas nuevas, obteniéndose así la gráfica G_1.

3. Si todos los vértices de la gráfica han sido numerados, el proceso termina; de otra forma, considérese el conjunto de vértices restantes y elíjase alguno con deficiencia mínima, constrúyase la gráfica de eliminación correspondiente y asígnesele el menor entero positivo que no haya sido asignado.

Ejemplo

Consideremos la siguiente matriz A y su gráfica asociada G^A en la figura 2.27. El vértice de mínima deficiencia es v_7 por lo que damos a éste el número 1 y además $D(v_7) = 0$. Observamos que la gráfica de eliminación G_1 no contiene aristas nuevas y los elementos de la matriz que antes eran cero lo siguen siendo.

En la gráfica G_1, el vértice v_6 tiene deficiencia $D(v_6) = 0$; damos a v_6 el número 2. La gráfica G_2 se construye a partir de G_1 quitando a v_6 y a las aristas que lo tienen como extremo. En la gráfica G_2 vemos que $D(v_4) = 0$, por la que v_4 tendrá el número 3 en la nueva numeración; la gráfica G_3 se construye como las anteriores. La deficiencia de los vértices en G_3 es uno para todos, así es que da lo mismo eliminar cualquiera de ellos; elimi

nemos v_1 para construir G_4. Las gráficas G_5 y G_6 se pueden observar a continuación de G_4. A los vértices v_1, v_2, v_3 y v_5 se les dan los números 4, 5, 6 y 7 respectivamente. La matriz del llenado F y gráfica G^F se muestran en la figura 2.29 también.

Como lo expresamos al principio de esta sección, el algoritmo de deficiencia mínima es un algoritmo heurístico, que encuentra ordenamientos subóptimos para los nodos de la gráfica G^A y con ello reduce el llenado de la matriz A. La idea fundamental de este consiste en encontrar, en cada paso de la eliminación, vértices cuya deficiencia sea mínima. La búsqueda de este tipo de vértices en general resulta costosa, pues para cada vértice v, el cálculo de su deficiencia involucra $(|V(v)||V(v)|+1|)/2$ pruebas y estos cálculos consumen mucho tiempo de procesador. Por ello es conveniente utilizar otro tipo de algoritmos, como el de grado mínimo J. Rose [16,17] que elimina en cada paso nodos de grado mínimo; esta operación reduce considerablemente el tiempo en los cálculos.

$$A = \begin{bmatrix} x & x & x & x & . & . & . \\ x & x & . & . & x & . & . \\ x & . & x & x & x & x & . \\ x & . & x & x & . & x & . \\ . & x & x & . & x & . & x \\ . & . & x & x & . & x & . \\ . & . & . & . & x & . & x \end{bmatrix}$$

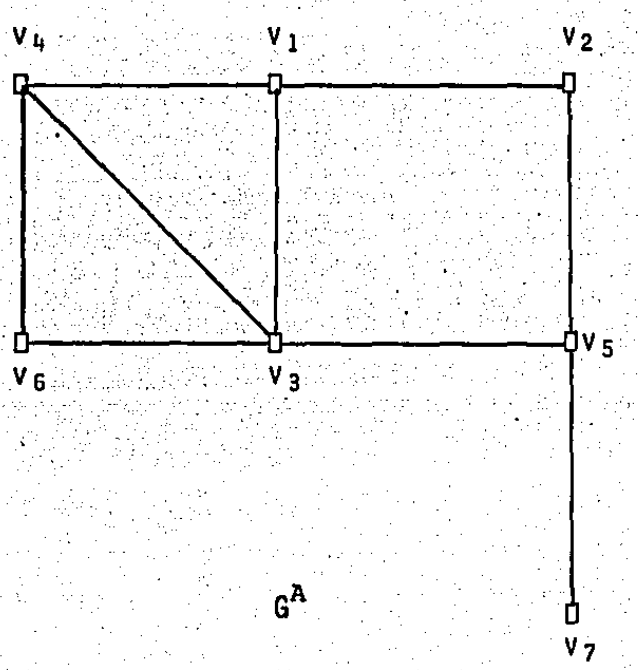
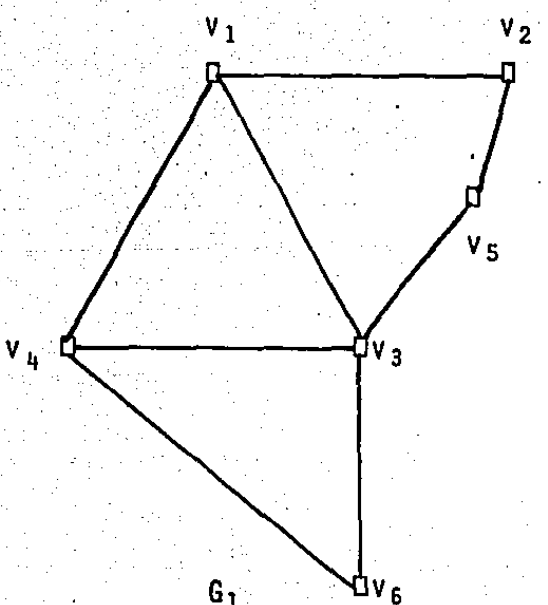


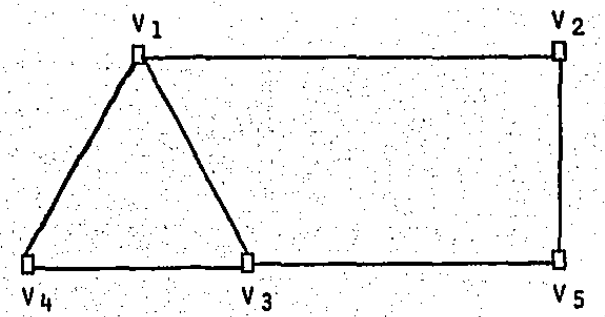
Figura 2.27

$$A_1 = \begin{bmatrix} x & x & x & x & . & . \\ x & x & . & . & x & . \\ x & . & x & x & x & x \\ x & . & x & x & . & x \\ . & x & x & . & x & . \\ . & . & x & x & . & x \end{bmatrix}$$



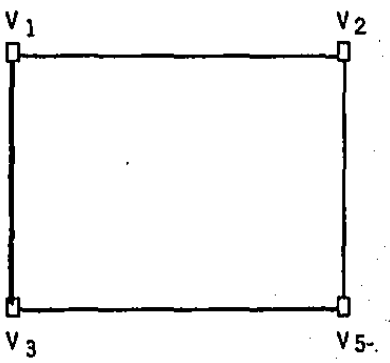
$$A_2 = \begin{bmatrix} x & x & x & x & . \\ x & x & . & . & x \\ x & . & x & x & x \\ x & . & x & x & . \\ . & x & x & . & x \end{bmatrix}$$

G_2



$$A_3 = \begin{bmatrix} x & x & x & . \\ x & x & . & x \\ x & . & x & x \\ . & x & x & x \end{bmatrix}$$

G_3



$$A_4 = \begin{bmatrix} x & x & x & x \\ x & x & x & x \\ x & x & x & x \end{bmatrix}$$

G_4

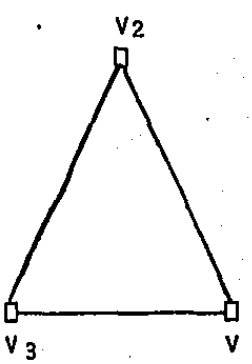
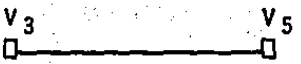


Figura 2.28

$$A_5 = \begin{bmatrix} x & x \\ x & x \end{bmatrix}$$

G_5

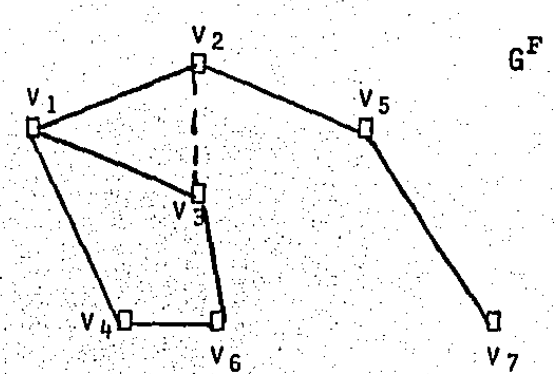


$$A_6 = \begin{bmatrix} x \end{bmatrix}$$

G_6

v_5

$$F = \begin{bmatrix} x & x & x & x & . & . & . \\ x & x & x & . & x & . & . \\ x & x & x & . & x & x & . \\ x & . & . & x & . & x & . \\ . & x & x & . & x & . & x \\ . & . & x & x & . & x & . \\ . & . & . & . & x & . & x \end{bmatrix}$$



G^F

Figura 2.29

A continuación consideramos la matriz del ejemplo anterior (figura 2.27), a la cual se le aplicará eliminación gaussiana; las matrices $A_1', \ldots, A_7'$ corresponden a la matriz que se obtiene en cada paso de la eliminación, mientras que A^{LL} es la matriz del llenado. Observemos que mediante este proceso el llenado de A es de 13 mientras que al aplicar el algoritmo de deficiencia mínima el llenado de A es de 2.

$$A_1' = \begin{bmatrix} x & x & x & x & . & . & . \\ . & x & x & x & x & . & . \\ . & x & x & x & x & x & . \\ . & x & x & x & . & x & . \\ . & x & x & . & x & . & x \\ . & . & x & x & . & x & . \\ . & . & . & . & x & . & x \end{bmatrix}$$

$$A_2' = \begin{bmatrix} x & x & x & x & . & . & . \\ . & x & x & x & x & . & . \\ . & . & x & x & x & x & . \\ . & . & x & x & x & x & . \\ . & . & x & x & x & . & x \\ . & . & x & x & . & x & . \\ . & . & . & . & x & . & x \end{bmatrix}$$

$$A_3' = \begin{bmatrix} x & x & x & x & . & . & . \\ . & x & x & x & x & . & . \\ . & . & x & x & x & x & . \\ . & . & . & x & x & x & . \\ . & . & . & x & x & x & x \\ . & . & . & x & x & x & . \\ . & . & . & . & x & . & x \end{bmatrix}$$

$$A_4' = \begin{bmatrix} x & x & x & x & . & . & . \\ . & x & x & x & x & . & . \\ . & . & x & x & x & x & . \\ . & . & . & x & x & x & . \\ . & . & . & . & x & x & x \\ . & . & . & . & x & x & . \\ . & . & . & . & x & . & x \end{bmatrix}$$

$$A_5' = \begin{bmatrix} x & x & x & x & . & . & . \\ . & x & x & x & x & . & . \\ . & . & x & x & x & x & . \\ . & . & . & x & x & x & . \\ . & . & . & . & x & x & x \\ . & . & . & . & . & x & x \\ . & . & . & . & . & x & x \end{bmatrix}$$

$$A_6' = \begin{bmatrix} x & x & x & x & . & . & . \\ . & x & x & x & x & . & . \\ . & . & x & x & x & x & . \\ . & . & . & x & x & x & . \\ . & . & . & . & x & x & x \\ . & . & . & . & . & x & x \\ . & . & . & . & . & . & x \end{bmatrix}$$

$$A^{LL} = \begin{bmatrix} x & x & x & x & . & . & . \\ x & x & x & x & x & . & . \\ x & x & x & x & x & x & . \\ x & x & x & x & x & x & . \\ . & x & x & x & x & x & x \\ . & . & x & x & x & x & x \\ . & . & . & . & x & x & x \end{bmatrix}$$

CAPITULO III

En este capítulo se dan ejemplos que ilustran los resultados al aplicar los algoritmos del capítulo II. Algunos de ellos muestran las ventajas entre los ordenamientos iniciales y los obtenidos posteriormente; los otros se seleccionado con el fin de señalar dónde pueden fallar tales algoritmos; e.d los resultados no son los deseados. Esto se debe a que los algoritmos son heurísticos y de aquí que los resultados en algunas ocasiones estén lejos de ser óptimos.

III. 1 La ecuación de Poisson discretizada en una región rectangular con elementos triangulares.

Para esta región se usaron 2-6 elementos descritos en [15]. Se puede observar que el número de nodos es de 81. La matriz del sistema original (figura 3.1) tiene un ancho de banda de 56. Su gráfica asociada G se muestra en la figura 3.2. Observemos que el nuevo ordenamiento para los nodos de G (figura 3.3) produce un ancho de banda de 12 y el perfil es de 386.

Ahora consideremos la matriz de la figura 3.5; ésta corresponde al sistema de ecuaciones antes descrito en donde se han numerado los nodos en distinto orden. Aquí el ancho de banda es de 76. El ordenamiento dado a los nodos de su gráfica asociada (figura 3.6) es el de la figura 3.3; de aquí que el ancho de banda y el perfil coincidan con los dados al reordenar la matriz de la figura 3.1.

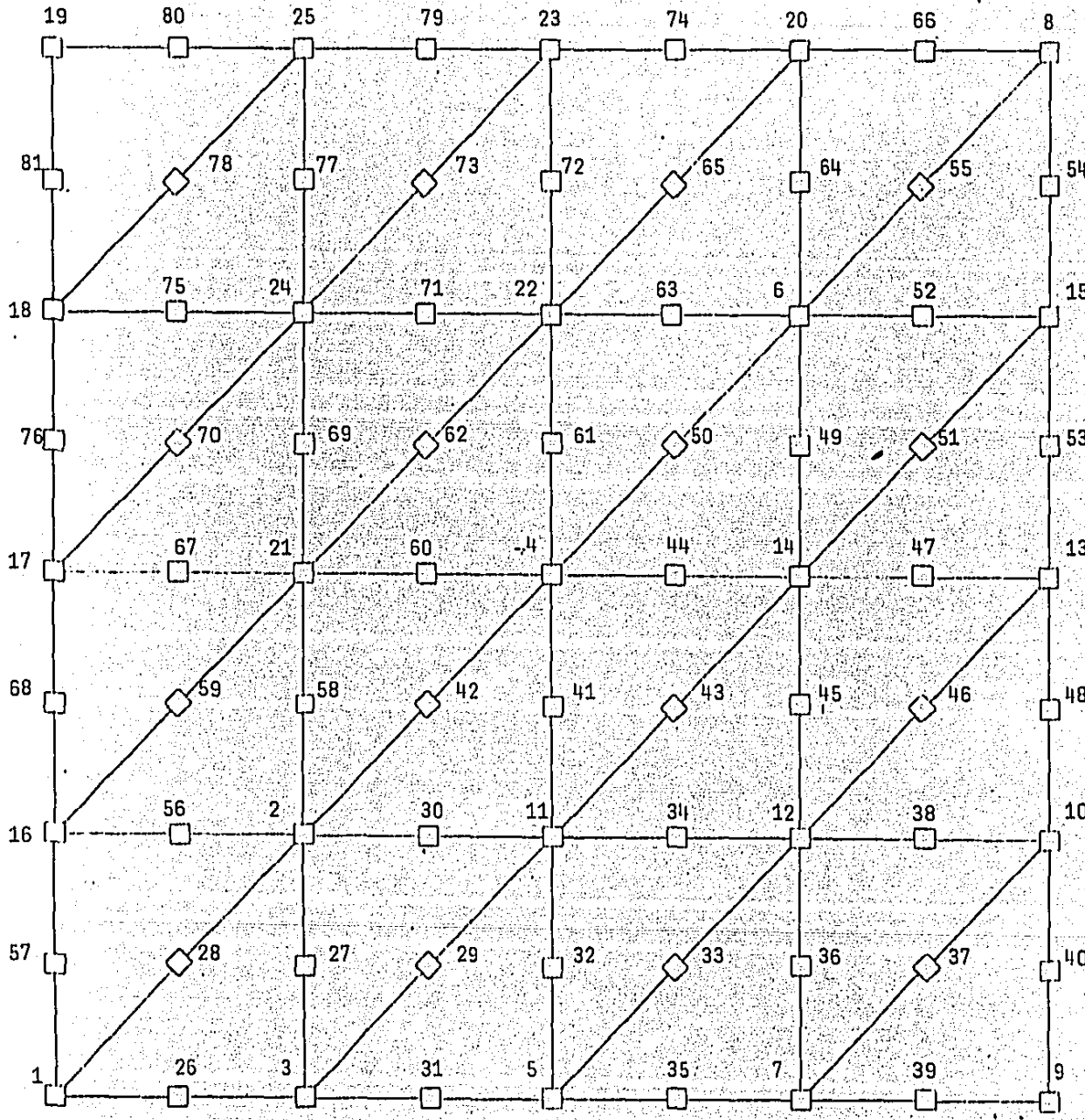

19 80 25 79 23 74 20 66 8
81 78 77 73 72 65 64 55 54
75 24 71 22 63 6 52 15
18
76 70 69 62 61 50 49 51 53
67 21 60 4 44 14 47 13
17
68 59 58 42 41 43 45 46 48
56 2 30 11 34 12 38 10
16
57 28 27 29 32 33 36 37 40
1 26 3 31 5 35 7 39 9

Figura 3.1

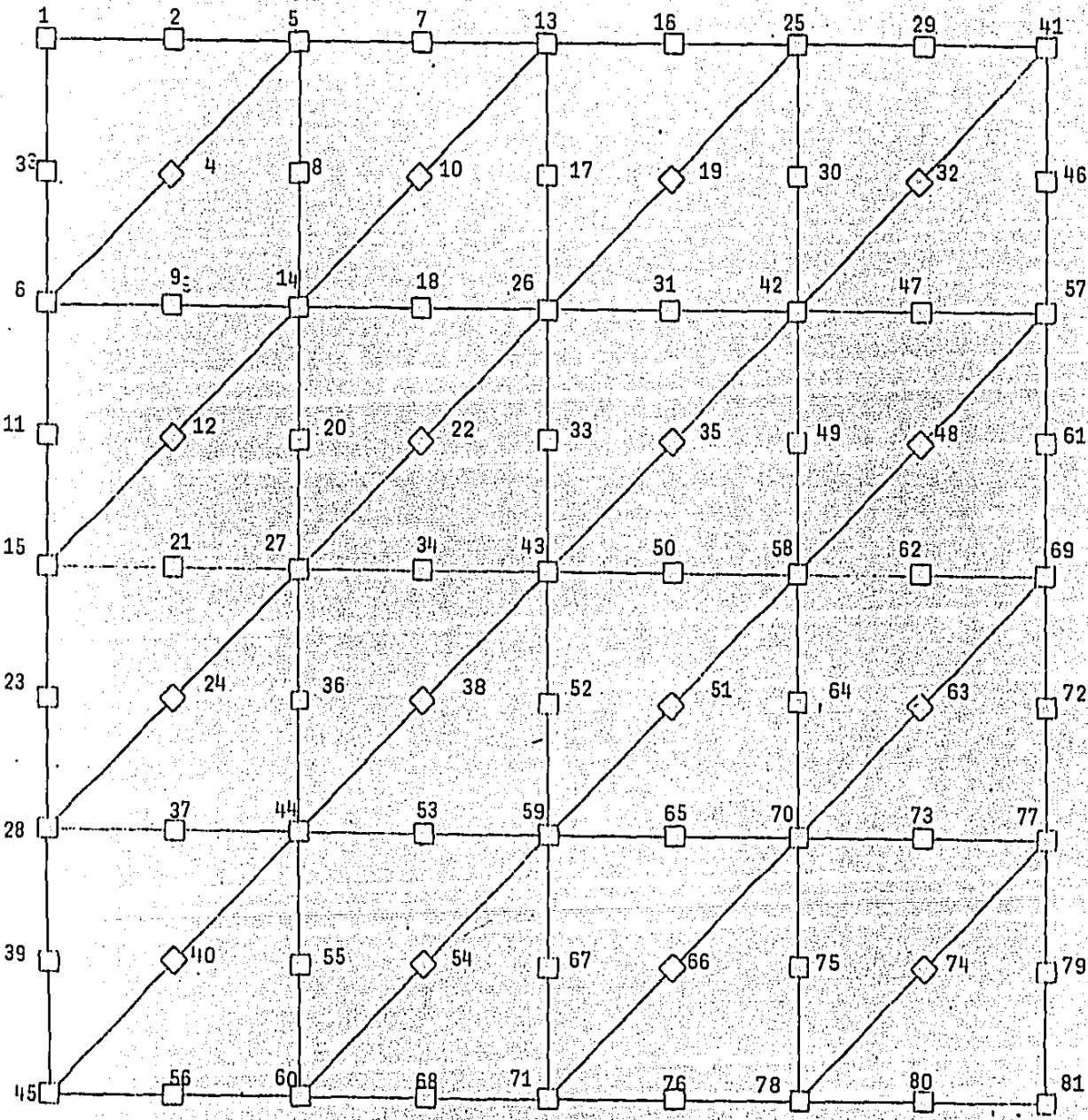

1
2
5
7
13
16
25
29
41
3
4
8
10
17
19
30
32
46
6
9
14
18
26
31
42
47
57
11
12
20
22
33
35
49
48
61
15
21
27
34
43
50
58
62
69
23
24
36
38
52
51
64
63
72
28
37
44
53
59
65
70
73
77
39
40
55
54
67
66
75
74
79
45
56
60
68
71
76
78
80
81

Figura 3.3

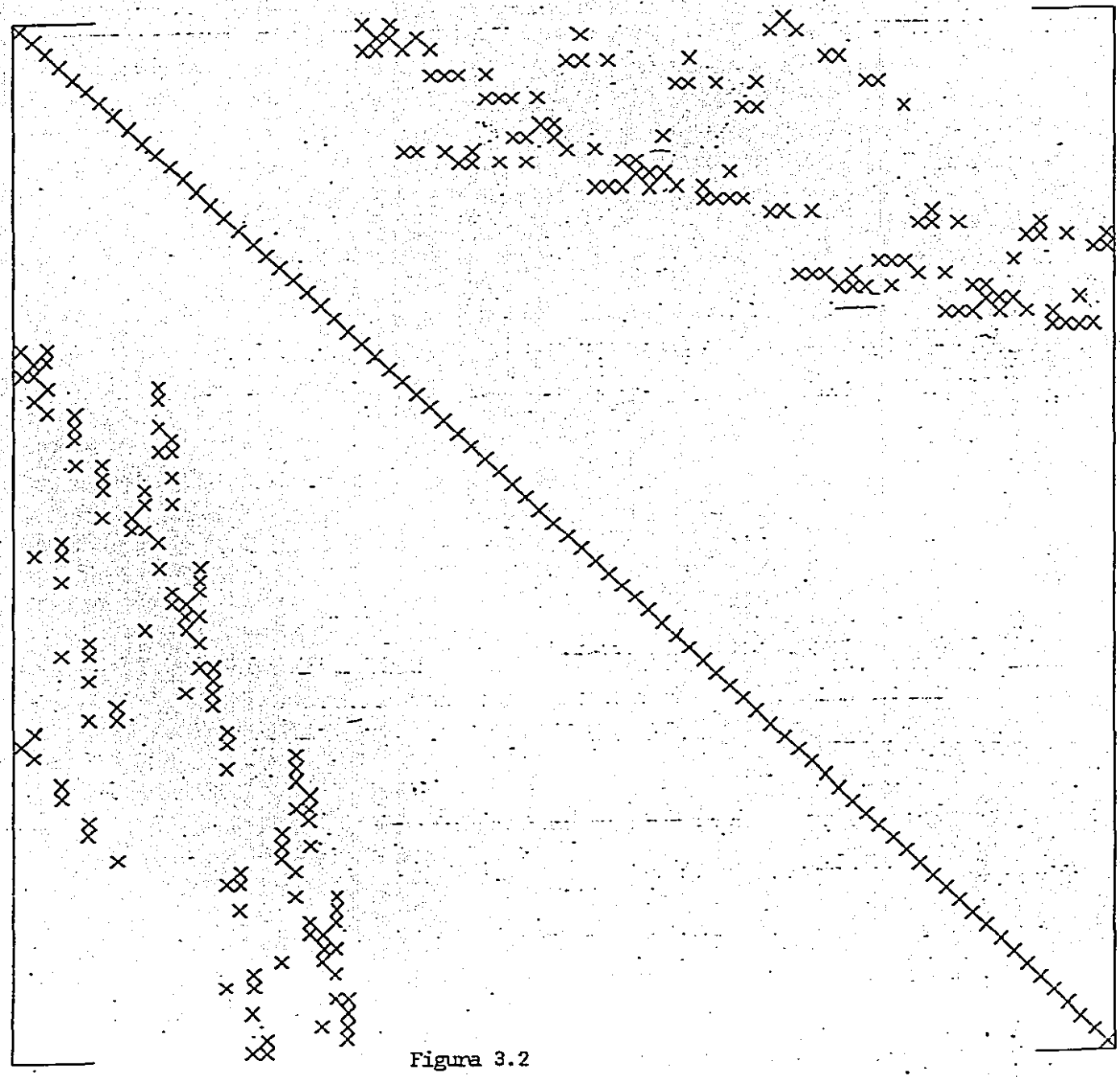


Figura 3.2

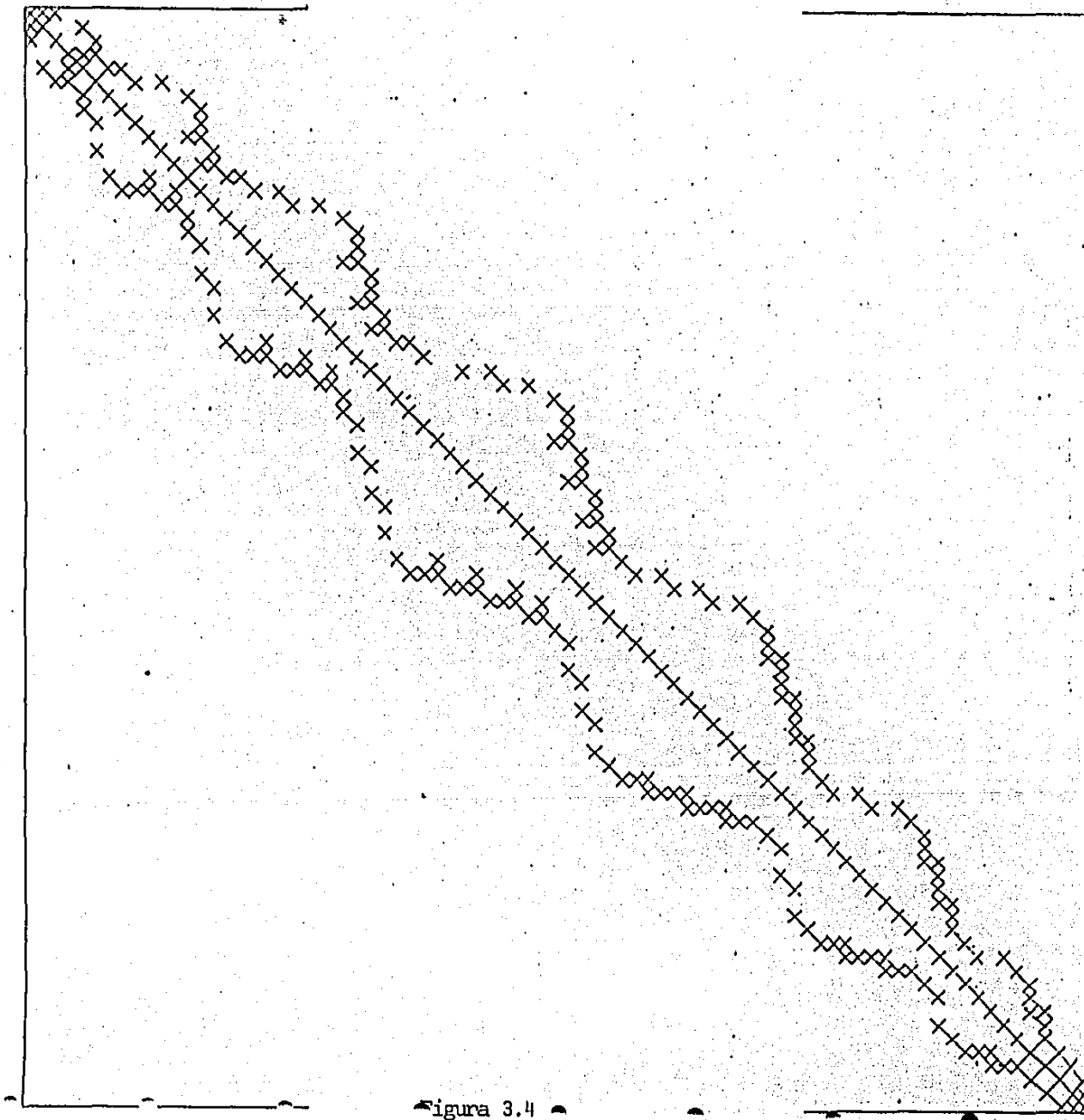


Figura 3.4

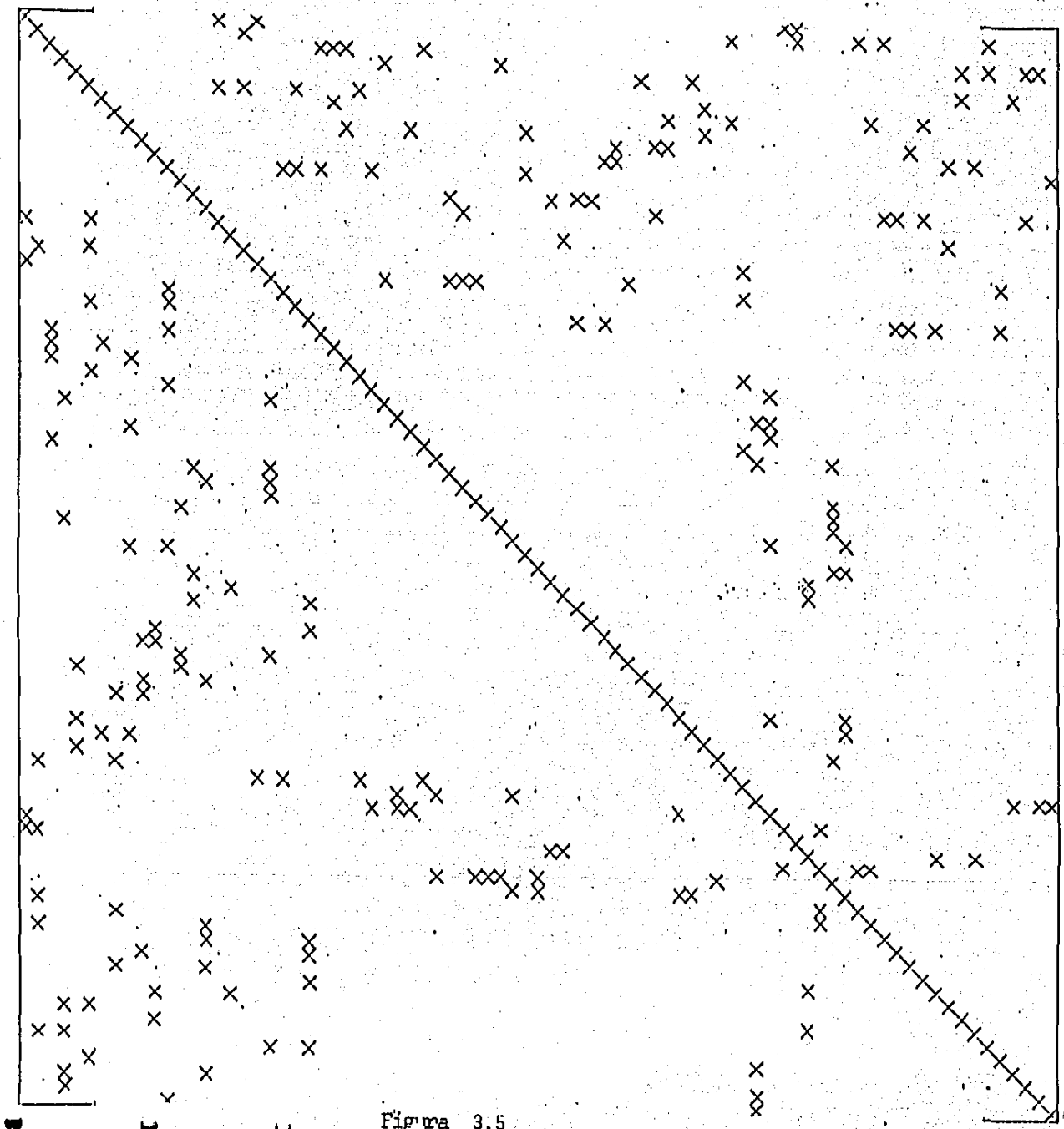


Figura 3.5

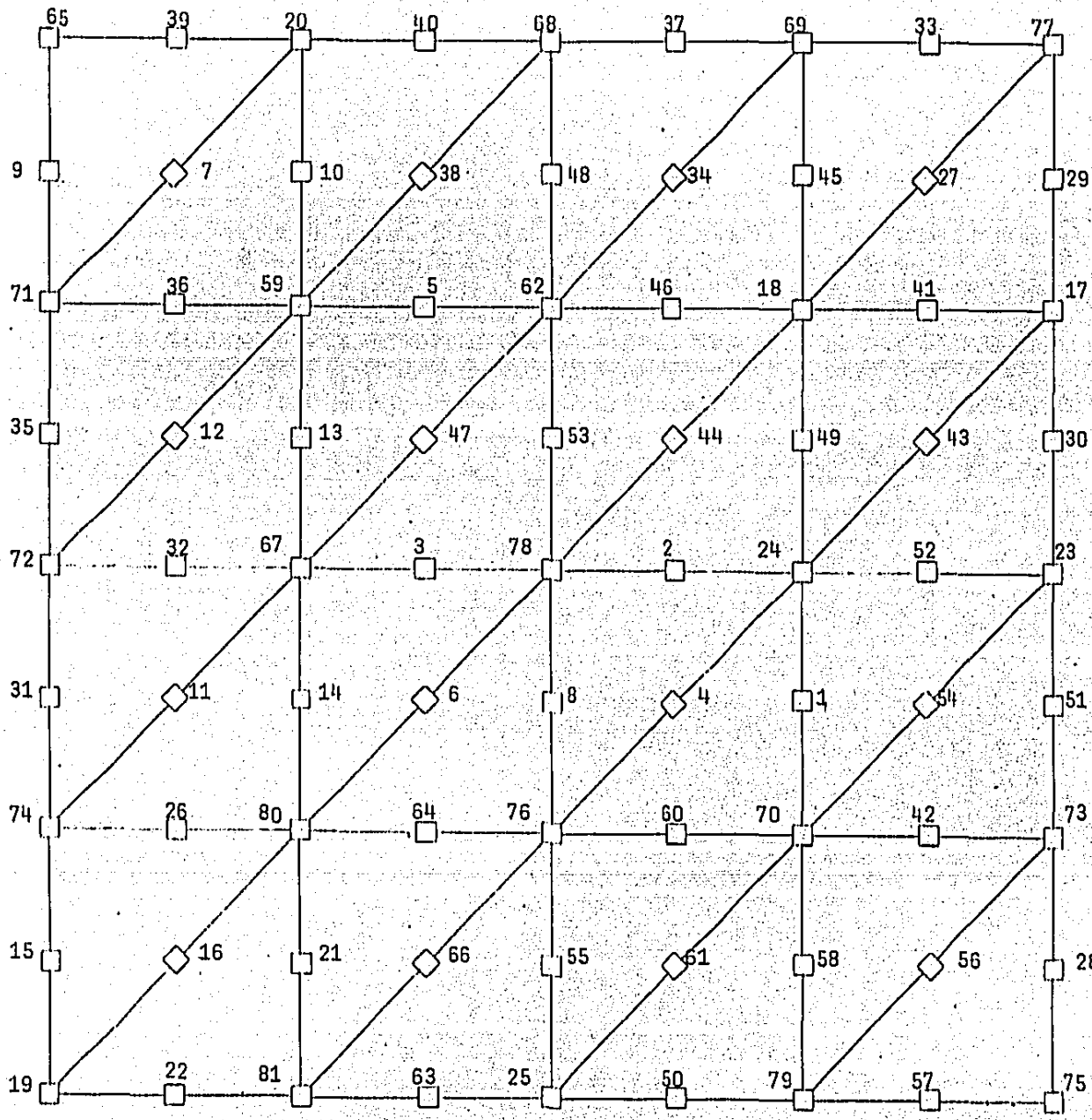

65 39 20 40 68 37 69 33 77
9 7 10 38 48 34 45 27 29
71 36 59 5 62 46 18 41 17
35 12 13 47 53 44 49 43 30
72 32 67 3 78 2 24 52 23
31 11 14 6 8 4 1 54 51
74 26 80 64 76 60 70 42 73
15 16 21 66 55 61 58 56 28
19 22 81 63 25 50 79 57 75

Figura 3.6

III. 2 La ecuación de Poisson discretizada en una malla variable.

En este ejemplo consideramos una región rectangular, subdividada en rectángulos. Se hicieron variar las dimensiones de la malla para observar los cambios sobre el ancho de banda. De aquí se concluye que si la malla es de n por n, la banda de la matriz tiene un ancho de n; mientras que si las dimensiones son de $n \times m$ para $n < m$ el ancho de banda es de $n + 1$. otro resultado importante sobre este ejemplo se obtuvo al numerar los nodos por columnas y después por renglones; de estos ordenamientos, los algoritmos dieron el mismo resultado; e. d las numeraciones finales coinciden, aún cuando la manera de ordenar a los nodos inicialmente fué distinta. La tabla siguiente es una muestra de lo expuesto.

n	m	no. de nodos	Ancho de banda	Perfil
5	5	25	5	90
5	10	50	6	235
10	10	100	10	705
10	20	200	11	1795
50	50	2500	50	84525
50	100	5000	51	211975

III. 3 Una muestra de matrices generadas aleatoriamente

El objetivo de este ejemplo consiste en verificar si el ancho de banda obtenido por el algoritmo es el óptimo. Por el tamaño de matrices (or en 10) fué posible encontrarlo; pero conforme el orden aumenta, esto se vuelve más complicado. Las iniciales β O , β RCM , β M , EDC, ENV RCM corresponden a el ancho de banda original, ancho de banda dado por el RCM, ancho de banda mínimo, porcentaje de elementos distintos de cero (se incluye la diagonal) y número de elementos de la envolvente después del RCM, respectivamente. Las matrices se generaron con la función TIME(x) donde x = 0,1, ..., 12 de la Burroughs 7000, que proporciona números al azar entre 0 y 1.

BO	BRCM	BM	EDC	ENVRCM
8	2	2	20	5
7	1	1	20	5
8	1	1	16	3
8	1	1	18	4
4	2	2	18	4
8	2	2	22	6
7	2	2	20	5
3	1	1	14	2
6	2	2	20	5
3	1	1	14	2
7	1	1	20	5
7	2	2	26	8
2	1	1	12	1
3	1	1	12	1
6	1	1	22	6
8	2	2	18	4
9	2	2	26	8
8	1	1	22	6
7	4	4	46	25
9	5	4	50	25
8	5	4	48	30
9	7	5	58	35
9	4	4	40	22
9	5	5	52	28
7	4	4	44	22
9	5	4	48	24
8	6	5	54	31
9	6	6	72	35
7	7	7	78	39
9	8	7	80	40
9	7	7	78	34

III. 4 ¿Qué sucede con los empates?

Recordemos que cuando el CM procede a numerar los nodos de una gráfica se tiene que si, v_i, $v_j \in V(v_k)$, $gr(v_i) = gr(v_j)$ y ℓ es el entero por asignar, entonces v_i o v_j pueden tomar indistintamente la etiqueta ℓ. En esta observación se basa este ejemplo. Supongamos que se ha dado a la malla de la figura 3.7 la numeración que ahí se muestra, en donde el ancho de banda de la matriz correspondiente es de 3.

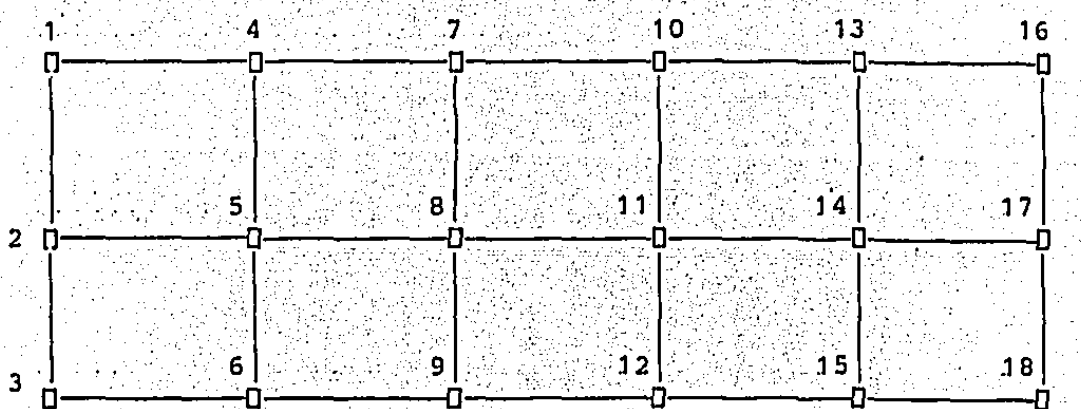


Figura 3.7

Mediante el CM la numeración queda como en la figura 3.8 con la cual el ancho de banda es de 4. Una manera de resolver esto e.d. que si el ancho de banda del ordenamiento original es menor que el proporcionado por el CM, pués que se quede con el ordenamiento original, esta solución es muy fácil pero en realidad el problema no esta ahí sino en la manera de resolver los empates. Una muestra de ello queda establecida en el ordenamiento de la figura 3.9 que produce un ancho de banda de 3 y se llegó a esta solución al asignar la etiqueta 2 donde el CM colocó la etiqueta 3 y viceversa.

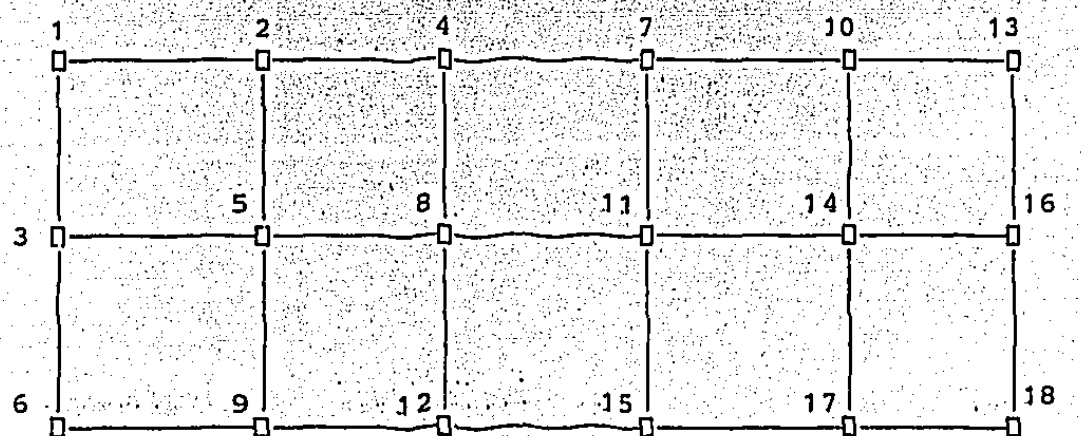


Figura 3.8

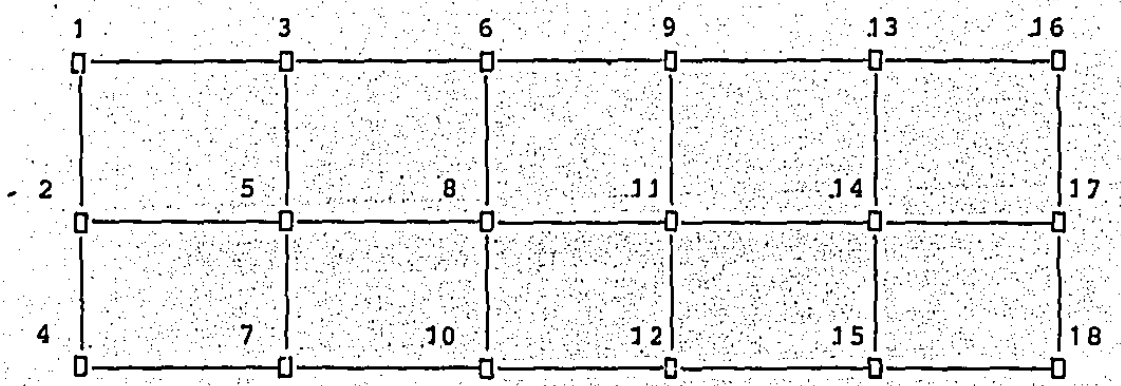


Figura 3.9

CONCLUSIONES

Se ha comprobado mediante los ejemplos expuestos aquí y en algunos otros no enunciados, que los algoritmos del capítulo II proporcionan ordenamientos subóptimos para reducir el ancho de banda, el perfil y el llenado de una matriz simétrica, definida positiva y rala. Sobre el RCM quedó demostrado que al reordenar una matriz mediante el CM, el perfil no aumenta al hacer la inversión. Con respecto al CM señalaremos que dada la manera como se resuelven los empates, e.d. si un vértice se numera enseguida u otro ocasiona que el ancho de banda pueda diferir al elegir numerar uno u otro vértice. Desde luego que el ver a futuro cual vértice conviene numerar primero y cual después trae consigo mayor tiempo en obtener el ordenamiento, así como la utilización de más arreglos para guardar la información que se requiera en ese momento o en su defecto complicar la estructura del programa. De la forma en la que se eligen los nodos iniciales para el algoritmo de Gibbs et al [17], observamos que los nodos de excentricidad grande no son considerados

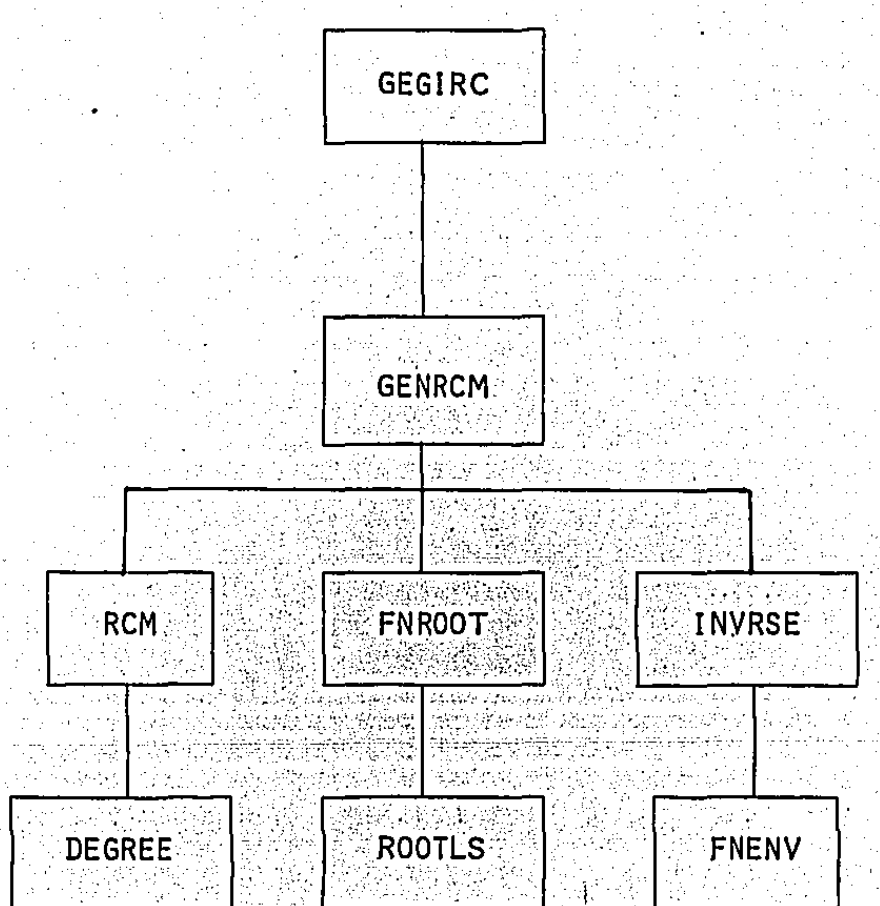
cuando se encuentran en los niveles intermedios, por esto podría resultar conveniente elegir un conjunto de nodos al y seleccionar alguno con excentricidad máxima e iniciar el algoritmo con este como nodo inicial.

APENDICE

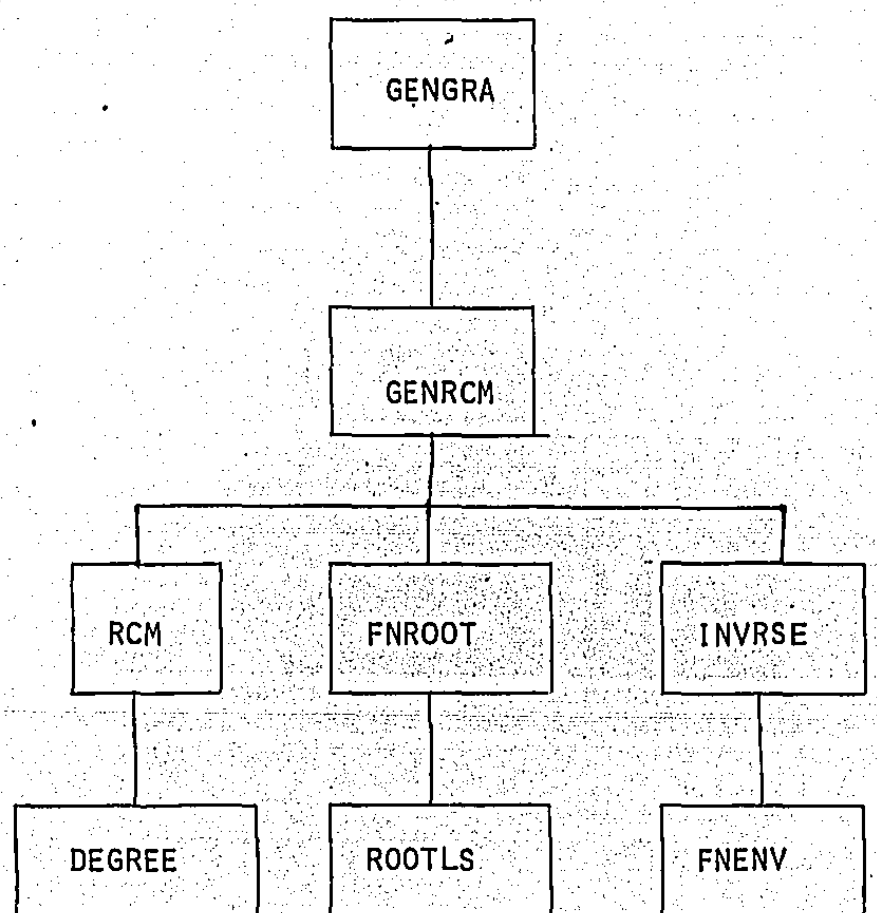
Los ordenamientos encontrados para los nodos de los ejemplos del capítulo III se obtuvieron utilizando los algoritmos que se describen en el capítulo II, secciones 2.2, 2.3 y 2.4. La instrumentación computacional se presenta a continuación; así como los programas con breves comentarios para cada una de las subrutinas. Todas ellas están compiladas en FORTRAN IV de la B-7000. Los esquemas de las señalan el orden para dicha instrumentación. Con las subrutinas ahí mencionadas se obtiene un ordenamiento subóptimo en los nodos de una gráfica G para reducir el ancho de banda y el perfil de una matriz simétrica y definida positiva cuya gráfica es G. Observemos que los esquemas A y B difieren en la subrutina que construye la estructura de adyacencia. El esquema A contine a GEGIRC, que utiliza más arreglos que GENGRA y por lo tanto más memoria, pero GEGIRC es procesada en menos tiempo que GENGRA.

El esquema C muestra un esqueleto que define el camino a seguir, para encontrar un reordenam nto en los nodos de una gráfica asociada a una matriz A, que reduce el llenado de A.

A



B



Dada una matriz simétrica de orden n, sabemos que es posible asociarle una gráfica G^A con n puntos. (Capítulo II, secc. 1). La gráfica queda completamente determinada cuando para cada vértice v_i con $i = 1,2,\ldots, n$, sabemos cuales índices $j,(v_i, v_j) \in A(G^A)$.

Las subrutinas GENGRA y GEGIRC construyen a partir de una matriz A, con las características antes mencionadas, la estructura de adyacencia de su gráfica asociada G^A. Para ello, se utiliza un esquema descrito por A. George en [ ]. Este esquema consta de dos vectores unidimensionales llamados ADJNCY y XADJ de longitud $2|A(G^A)|$ y $n + 1$ respectivamente. El vector ADJNCY guarda para cada vértice v_i el conjunto $ADY(v_i)$, el cual se encuentra después del conjunto $ADY(v_{i-1})$ y antes de $ADY(v_{i+1})$. En el vector XADJ, se establecen posiciones para cada vértice mediante la regla:

$$XADJ(v_1) = 1$$

y $$XADJ(v_i) = XADJ(v_{i-1}) + |AD(v_{i-1})|, \quad i = 2,3\ldots n$$

De lo anterior, tenemos que si

$$XADJ(v_i) \neq XADJ(v_{i+1})$$

y $\mathcal{L}$ es tal que

$$XADJ(v_i) \leq \mathcal{L} \leq XADJ(v_{i+1}) - 1$$

$$V_j \in ADy(v_i) \quad si \quad V_j = ADJNC(\mathcal{L}).$$

GENGRA

Sea A una matriz rala simétrica y definida positiva Supongamos que A ha sido almacenada por columnas, en donde únicamente figuran los elementos distintos de cero a_{ij}, con $i \geq j$.

La subrutina GENGRA construye la estructura de adyacencia de la gráfica G^A asociada a A, operando sobre los arreglos IR, AD, ADJNCY y XADJ. De entrada IR contiene para cada a_{ij} distinta de cero con $i \geq j$ el índice i, mientras que AD es un vector de apuntadores para el principio y el final de cada renglón. Los arreglos de salida ADJNCY y XADJ que contiene para cada vértice su vecindad y posición de ésta, respectivamente, son utilizados por GENGRA para localizar los vecinos j de un vértice i si j es menor que i.

```
C     *************************************************************
C
C     ........................ GENGRA  ...........................
C
C     *************************************************************
C
C     LA SUBRUTINA GENGRA  GENERA LA ESTRUCTURA DE ADYACENCIA DE UNA
C     GRAFICA ASOCIADA A UNA MATRIZ ,SIMETRICA DEFINIDA POSITIVA DE
C     ORDEN EL NUMERO DE ECUACIONES.
C
C     *************************************************************
C
C     PARAMETROS DE ENTRADA -
C        IR - ARREGLO DE UNA DIMENSION QUE CONTIENE EL INDICE DE REN-
C          GLON DE CADA ELEMENTO DISTINTO DE CERO AIJ,EN DONDE I ES
C          MENOR O IGUAL QUE J.
C       AD - ARREGLO DE UNA DIMENSION DE LONGITUD EL NUMERO DE
C          ECUACIONES MAS UNO. CONTIENE APUNTADORES AL PRINCIPIO
C          Y AL FINAL DE CADA RENGLON.
C       NEC - NUMERO DE ECUACIONES.
C
C      PARAMETROS DE SALIDA -
C       ADJNCY - ARREGLO DE UNA DIMENSION QUE CONTIENE LA VECIN-
C          DAD DE CADA VERTICE.
C        XADJ - CONTIENE APUNTADORES AL PRINCIPIO Y FINAL DE CADA
C          VECINDAD.
C
C      ************************************************************
C
      SUBROUTINE GENGRA (IR,AD,ADJNCY,XADJ,NEC)
C
C      ************************************************************
C
      INTEGER IR(1),AD(1),ADJNCY(1),XADJ(1),NEC
C
C      ************************************************************
C
C      ------
C      INICIA
C      ------
C
      L=1
      I=1
      XADJ(I)=1
      N=AD(I+1)-AD(I)
        IF(N.NE.1) GO TO 1
       XADJ(I+1)=XADJ(I)
        GO TO 2
        DO 100 J=AD(I)+1,AD(I+1)-1
         ADJNCY(L)=IR(J)
100      L=L+1
          XADJ(I+1)=XADJ(I)+N-1
2       DO 200 I=2,NEC
         NV=0
         DO 300 K=1,I-1
          K1=XADJ(K+1)-XADJ(K)
         IF(K1.EQ.0) GO TO 300
         DO 400 M=XADJ(K),XADJ(K+1)-1
          IF(ADJNCY(M).GT.I) GO TO 300
          IF(ADJNCY(M).LT.I) GO TO 400
          ADJNCY(L)=IR(AD(K))
```

```
            NV=NV+1
            L=L+1
          GO TO 300
400        CONTINUE
300       CONTINUE
         N=AD(I+1)-AD(I)
         IF(N.EQ.1) GO TO 3
         DO 500 J=AD(I)+1,AD(I+1)-1
          ADJNCY(L)=IR(J)
500       L=L+1
3        XADJ(I+1)=XADJ(I)+NV+N-1
200     CONTINUE
         RETURN
         END
```

GEGIRC

Dada una matriz A simétrica, definida positiva y rala, la estructura de adyacencia de su gráfica asociada G^A, es generada por GEGIRC. Esta subrutina trabaja sobre los arreglos IR, IC, AIR, AIC; en donde IR e IC contienen los índices de renglón y de columna, respectivamente, de cada elemento a_{ij} distinto de cero, con $i \geq j$. Los vectores AIR y AIC están relacionados con IR e IC, de tal manera que:

Para cada renglón y/o columna, es posible determinar el número de elementos distintos de cero, así como su principio y final.

De salida, la estructura de adyacencia de G^A, se encuentra en un par de vectores unidimensionales ADJNCY y XADJ. En donde el primero contiene para cada vértice a su vecindad y el segundo señala el inicio y el final de esta.

```
C       ************************************************************
C       ..............  GEGIRC  ..................................
C       ************************************************************
C
C       ESTA SUBRUTINA CONSTRUYE LA ESTRUCTURA DE ADYACENCIA DE LA
C       GRAFICA GA ASOCIADA A LA MATRIZ A , QUE ES SIMETRICA,DEFINI-
C       DA POSITIVA Y DE ORDEN N. GEGIRC UTILIZA LOS INDICES DE REN-
C       GLON Y COLUMNA DE CADA ELEMENTO A(I,J) DISTINTO DE CERO,
C       CON I MAYOR O IGUAL A J.
C
C       ************************************************************
C
C       PARAMETROS DE ENTRADA -
C         IR E IC -ARREGLOS UNIDIMENSIONALES QUE CONTIENEN LOS INDI-
C            CES DE RENGLON Y COLUMNA RESPECTIVAMENTE, DE CADA EE-
C            MENTO DISTINTO DE CERO.
C          AIR Y AIC - ARREGLOS DE UNA DIMENSION QUE APUNTAN AL PRIN-
C            CIPIO Y FINAL DE CADA RENGLON Y CADA COLUMNA.
C
C       PARAMETROS DE SALIDA -
C         ADJNCY - VECTOR DE UNA DIMENSION QUE CONTIENE LA VECINDAD
C            DE CADA VERTICE.
C         XADJ - SENALA LA POSICION DEL INICIO Y FINAL DE CADA VE-
C            CINDAD EN ADJNCY.
C
C       ************************************************************
C
        SUBROUTINE GEGIRC  (IR,IC,AIR,AIC,ADJNCY,XADJ,NEC)
C
C       ************************************************************
C
        INTEGER IR(1),IC(1),AIR(1),AIC(1),ADJNCY(1),XADJ(1),NEC
C
C       ************************************************************
C
C       -------
C       INICIA
C       -------
         L=1
         I=1
        XADJ(I)=1
         M=1
        N=AIR(I+1)-AIR(I)
        IF(N.EQ.1)GO TO 10
         GO TO 30
 40     DO 100 I=2,NEC
        M=AIC(I)-AIC(I-1)
        IF(M.EQ.1) GO TO 20
          DO 300 K=AIC(I-1)+1,AIC(I)-1
          ADJNCY(L)=IC(K)
 300      L=L+1
 20     N=AIR(I+1)-AIR(I)
        IF(N.EQ.1) GO TO 10
 30       DO 200 J=AIR(I)+1,AIR(I+1)-1
          ADJNCY(L)=IR(J)
 200      L=L+1
 10      XADJ(I+1)=XADJ(I)+N+M-2
        IF(I.EQ.1) GO TO 40
 100    CONTINUE
        RETURN
        END
```

Sea ROOT el nombre de un nodo inicial. La estructura de nivel cimentada en ROOT es obtenida por la subrutina ROOTLS, la cual utiliza cinco arreglos de una dimensión XADJ, ADJNCY y MASK en donde los dos primeros contienen la estructura de adyacencia de la gráfica y el tercero indica cuales nodos pertenecen a la misma componente que ROOT. Los arreglos LS y XLS contienen la estructura de nivel. Los nombres de los nodos de cada nivel se encuentran en LS en donde el nivel N_i se localiza después del N_{i-1} se localiza después del N_{i-1} y antes del N_{i+1} mientras que en XLS es un vector de apuntadores para el inicio y el final de cada nivel. El entero NLVL corresponde al número de niveles en (XLS, LS).

```
C     ***************************************************************
C     ********* ROOTLS.. CONSTRUYE LA ESTRUCTURA DE NIVEL *********
C     ***************************************************************
C
C      ESTA SUBRUTINA GENERA LA ESTRUCTURA DE NIVEL CIMENTADA
C       EN EL VERTICE LLAMADO ROOT. UNICAMENTE LOS NODOS PARA
C       LOS CUALES MASK NO ES CERO SE CONSIDERAN.
C      ***************************************************************
C      PARAMETROS DE ENTRADA -
C         ROOT- EL NODO EN EL CUAL SE CIMENTA LA ESTRUCTURA DE
C          NIVEL.
C
C        (XADJ,ADJNCY) - CONTIENE LA ESTRUCTURA DE ADYACENCIA
C                        DE LAGRAFICA.
C
C        MASK - SE USA PARA ESPECIFICAR UNA SECCION DE LA GRA-
C         FICA. NODOS CON MASK(I)=0 SE IGNORAN.
C
C      PARAMETROS DE SALIDA -
C
C       NLVL - CUENTA EL NUMERO DE NIVELES.
C
```

```
C          (XLS,LS) -CONTIENE A LA ESTRUCTURA DE NIVEL CIMENTADA
C          EN ROOT. XLS ES UN VECTOR DE POSICION DE LA ESTRUCTU-
C          RA DE NIVEL Y LS ES UN ARREGLO DE UNA DIMENSION ,
C          QUE GUARDA EN ORDEN SUCESIVO LOS NOMBRES DE LOS
C          VERTICES DE CADA NIVEL.
C     ***************************************************************
C
      SUBROUTINE ROOTLS (ROOT,XADJ,ADJNCY,MASK,NLVL,XLS,LS)
C
C     ***************************************************************
C
        INTEGER ADJNCY(1),LS(1),MASK(1),XLS(1)
        INTEGER XADJ(1),I,J,JSTOP,JSTRT,LBEGIN,
     1          CCSIZE,LVLEND,LVSIZE,NBR,NLVL,
     1           NODE,ROOT
C
C     ***************************************************************
C
C         ------------------------------------------------------
C         CONSTRUYE LA ESTRUCTURA DE NIVEL CIMENTADA EN ROOT
C         ------------------------------------------------------
C         -------
C         INICIA
C         -------
          MASK(ROOT)=0
          LS(1)=ROOT
        NLVL=0
        LVLEND=0
        CCSIZE=1
C       ------------------------------------------------------------
C       LBEGIN ES UN APUNTADOR AL PRINCIPIO DEL NIVEL CORRIENTE
C       Y LVLEND APUNTA AL FINAL DE DICHO NIVEL.
C       ------------------------------------------------------------
  200   LBEGIN=LVLEND+1
        LVLEND=CCSIZE
        NLVL=NLVL+1
        XLS(NLVL)=LBEGIN
C       ------------------------------------------------------------
C       GENERA EL NIVEL SIGUIENTE ENCONTRANDO TODOS LOS VECI-
C        NOS DE LOS  NODOS MARCADOS CON MASK DEL NIVEL QUE SE
C         SE ESTA ANALIZANDO.
C       ------------------------------------------------------------
        DO 400 I=LBEGIN,LVLEND
        NODE=LS(I)
        JSTRT=XADJ(NODE)
        JSTOP=XADJ(NODE+1)-1
        IF(JSTOP.LT.JSTRT) GO TO 400
         DO 300 J= JSTRT,JSTOP
           NBR = ADJNCY(J)
        IF(MASK(NBR) .EQ. 0) GO TO 300
        CCSIZE=CCSIZE+1
        LS(CCSIZE)=NBR
        MASK(NBR)=0
  300           CONTINUE
  400           CONTINUE
```

```
C     ------------------------------------------------------------
C     CALCULA LA ANCHURA DEL NIVEL CORRIENTE. SI ESTA NO ES
C     CERO CONSTRUYE EL NIVEL SIGUIENTE.
C     ------------------------------------------------------------
      LVSIZE=CCSIZE-LVLEND
      IF(LVSIZE.GT.0) GO TO 200
C     ------------------------------------------------------------
C     REENGARZAR.ASIGNARLES UNO A LOS NODOS MARCADOS POR
C      MASK.
C     ------------------------------------------------------------
      XLS(NLVL+1)=LVLEND+1
       DO 500 I=1,CCSIZE
      NODE=LS(I)
      MASK(NODE)=1
500     CONTINUE
          RETURN
         END
```

FNROOT [7]

Esta subrutina instrumenta una versión modificada del esquema descrito por Gibbs et al [ ], para encontrar un par de nodos pseud -periféricos. Considera a ROOT como nodo inicial, los arreglos XADJ, ADJNCY, MASK, XLS, LS y al entero NLVL. Trabaja sobre la estructura de nivel construida por ROOT y elige del último nivel un nodo de grado mínimo v y construye su estructura de nivel. Si la estructura con raiz en v tiene más niveles que la cimentada en ROOT, elige un vértice con grado mínimo del último nivel de $N(v)$ y construye la estructura correspondiente. En caso de que el número de niveles en $N(v)$ sea menor que los de N(ROOT), se detiene el proceso. De otra forma se continua con el procedimiento hasta encontrar un vértice en el último nivel, que de origen a una estructura de nivel con más niveles que $N(v)$ si no existe, de salida (XLS, LS) contiene la estructura de nivel $N(v)$, ROOT = v y NLVL es el número de niveles de $N(v)$.

```
C       ***********************************************************
C       **********FNROOT ******  ENCUENTRA NODOS PSEUDOPERIFERICOS...
C       ***********************************************************
C
C       FNROOT INSTRUMENTA UNA VERSION MODIFICADA DEL ESQUEMA
C       DESCRITO POR GIBBS,POOLE Y STOCKMEYER PARA ENCONTRAR
C       NODOS PSEUDOPERIFERICOS. DETERMINA ESTOS NODOS PARA
C       LA SECCION DE LA SUBGRAFICA ESPECIFICADA POR MASK Y ROOT.
C       ***********************************************************
C
        SUBROUTINE FNROOT (ROOT,XADJ,ADJNCY,MASK,NLVL,XLS,LS)
C
C       ***********************************************************
        INTEGER ADJNCY(1),LS(1),MASK(1),XLS(1)
        INTEGER XADJ(1),CCSIZE,J,JSTRT,K,KSTOP,KSTRT,
     1    MINDEG,NABOR,NDEG,NLVL,NODE,NUNLVL,ROOT
C       ***********************************************************
C       ------------------------------------------------------------
C       DETERMINA LA ESTRUCTURA DE NIVEL CIMENTADA EN ROOT
C       ------------------------------------------------------------
        CALL ROOTLS (ROOT,XADJ,ADJNCY,MASK,NLVL,XLS,LS)
C
        CCSIZE=XLS(NLVL+1)-1
        IF(NLVL .EQ. 1 .OR. NLVL .EQ. CCSIZE) RETURN
C
C       ------------------------------------------------
C       TOMA UN NODO CON GRADO MINIMO DEL ULTIMO NIVEL
C       ------------------------------------------------
C
  100   JSTRT=XLS(NLVL)
        MINDEG=CCSIZE
        ROOT=LS(JSTRT)
        IF (CCSIZE .EQ.JSTRT) GO TO 400
           DO 300 J=JSTRT,CCSIZE
             NODE=LS(J)
             NDEG=0
             KSTRT=XADJ(NODE)
             KSTOP=XADJ(NODE+1)-1
             DO 200 K=KSTRT,KSTOP
                NABOR=ADJNCY(K)
                IF(MASK(NABOR).GT.0) NDEG=NDEG +1
  200        CONTINUE
             IF(NDEG.GE. MINDEG) GO TO 300
             ROOT=NODE
             MINDEG=NDEG
  300      CONTINUE
C
C
C         ---------------------------------
C         GENERA SU ESTRUCTURA DE NIVEL
C         ---------------------------------
C
  400   CALL ROOTLS (ROOT,XADJ,ADJNCY,MASK,NUNLVL,XLS,LS)
        IF (NUNLVL .LE.NLVL ) RETURN
        NLVL=NUNLVL
        IF ( NLVL .LT. CCSIZE) GO TO 100
           RETURN
           END
```

DEGREE [71]

Los grados de los nodos de una componente conexa en donde se encuentran ROOT y los nodos i para los cuales MASK(i) ≠ 0, son calculados por la subrutina DEGREE. De entrada los arreglos XADJ y ADJNCY contienen la estructura de adyaciencia de la gráfica ROOT y MASK caracterizan a la componente conexa que contiene a ROOT. El entero positivo CCSIZE corresponde al tamaño de la componente. LS es un arreglo temporal que se usa para almacenar los nodos de la componente conexa. Estos son almacenados nivel por nivel. El vector DEG de salida contiene los grados de los nodos de dicha componente.

```
*****************************************************
 ***DEGREE. OPERA SOBRE LA COMPONENTE SENALADA CON MASK **
*****************************************************
SUBROUTINE DEGREE (ROOT,XADJ,ADJNCY,MASK,DEG,CCSIZE,LS)
*****************************************************************
INTEGER ADJNCY(1),DEG(1),LS(1),MASK(1),XADJ(1),CCSIZE
INTEGER I,IDEG,J,JSTOP,JSTRT,LBEGIN,LVLEND,LVSIZE,
1       NBR,NODE,ROOT
*****************************************************************
 ---------------------------------------------------------------
 INICIO.
 EL ARREGLO XADJ SE USA COMO UN CONTADOR TEMPORAL PARA IN-
 DICAR, CUALES NODOS CERCANOS SE HAN CONSIDERADO.
 ---------------------------------------------------------------

  LS(1)=ROOT
 XADJ(ROOT)=-XADJ(ROOT)
 LVLEND=0
 CCSIZE=1
```

```
100      LBEGIN=LVLEND+1
         LVLEND=CCSIZE
         ------------------------------------------------------------
         CALCULA LOS GRADOS DE LOS NODOS EN EL NIVEL CORRIENTE; Y
         A LA VEZ ,GENERA EL NIVEL SIGUIENTE.
         ------------------------------------------------------------
          DO 400 I=LBEGIN,LVLEND
          NODE=LS(I)
          JSTRT=-XADJ(NODE)
          JSTOP=IABS(XADJ(NODE+1))-1
           IDEG=0
           IF( JSTOP.LT. JSTRT) GO TO 300
         DO 200 J=JSTRT,JSTOP
           NBR=ADJNCY(J)
           IF(MASK(NBR) .EQ.0  ) GO TO 200
            IDEG=IDEG+1
           IF(XADJ(NBR) .LT. 0) GO TO 200
            XADJ(NBR)=-XADJ(NBR)
           CCSIZE=CCSIZE+1
           LS(CCSIZE)=NBR
200        CONTINUE
300         DEG(NODE)=IDEG
400         CONTINUE
C
C        ------------------------------------------------------------
C        CALCULA LA ANCHURA DEL NIVEL CORRIENTE,SI ES NO CERO
C        GENERA OTRO NIVEL.
C        ------------------------------------------------------------
C
            LVSIZE=CCSIZE-LVLEND
           IF(LVSIZE .GT. 0) GO TO 100
C        ------------------------------------
C        CAMBIA XADJ A SU SIGNO CORRECTO.
C        ------------------------------------
          DO 500 I=1,CCSIZE
           NODE=LS(I)
           XADJ(NODE)=-XADJ(NODE)
500        CONTINUE
           RETURN
           END
```

RCM [17]

El ordenamiento con inversión de E. Cuthill y McKee es encontrado por la subrutina RCM. Para el funcionamiento de esta es necesario tener la estructura de adyacencia de la gráfica, en un par de vectores unidimensionales XADJ y ADJNCY, elegir un nodo y todos aquellos que se encuentren en la misma componente conexa que el: al nodo se le llama ROOT y los vértices de la componente están indentificados en el arreglo MASK. El vector DEG contiene los grados de los nodos de la componente conexa que contiene a ROOT. Finalmente RCM obtiene al entero CCSIZE que corresponde al número de nodos de la componente y al vector PERM en donde se encuentra el ordenamiento para dicha componente.

```
**********************************************************************

... RCM ... ORDENAMIENTO DE E. CUTHILL-MCKEE  ..........

**********************************************************************

PROPOSITO - RCM NUMERA UNA COMPONENTE CONEXA ESPECIFICADA
   POR MASK Y ROOT , USANDO EL ALGORITMO RCM.
   LA NUMERACION SE INICIA EN EL NODO ROOT.

PARAMETROS DE ENTRADA -
   ROOT - ES EL NODO QUE DEFINE A LA COMPONENTE CONEXA
          Y ES USADO COMO NODO INICIAL PARA EL ORDENA-
          MIENTO RCM.

   (XADJ,ADJNCY) - ESTRUCTURA DE ADYACENCIA PARA LA
                   GRAFICA.

PARAMETROS DE ACTUALIZACION -
   MASK - LOS NODOS QUE TIENEN VALOR DE ENTRADA EN MASK
          DISTINTO DE CERO SON CONSIDERADOS POR LA SUB-
          RUTINA. LOS NODOS NUMERADOS POR EL RCM TENDRAN
          VALOR DE CERO EN EL ARREGLO MASK.

PARAMETROS  DE SALIDA -
   PERM - CONTIENE EL ORDENAMIENTO OBTENIDO POR EL RCM.
   CCSIZE - ES EL TAMANO DE LA COMPONENTE CONEXA NUMERA-
            DA POR EL RCM.

PARAMETROS DE TRABAJO -
   DEG - ES UN VECTOR TEMPORAL PARA CALCULAR LOS GRADOS
         DE LOS NODOS N LA SUBGRAFICA ESPECIFICADA POR
         MASK Y ROOT.

SUBRUTINAS DE PROGRAMA -
   DEGREE.
**********************************************************************

SUBROUTINE RCM (ROOT,XADJ,ADJNCY,MASK,PERM,CCSIZE,DEG)

**********************************************************************

INTEGER ADJNCY(1),DEG(1),MASK(1),PERM(1),XADJ(1),CCSIZE
INTEGER FNBR,I,J,JSTOP,JSTRT,K,L,LBEGIN,LNBR,LPERM
INTEGER LVLEND,NDR,NODE,ROOT
**********************************************************************

----------------------------------------------------------------
ENCUENTRA LOS GRADOS DE LOS NODOS EN LA COMPONENTE ES-
PECIFICADA POR EL ARREGLO MASK Y EL NODO ROOT.
----------------------------------------------------------------


CALL DEGREE (ROOT,XADJ,ADJNCY,MASK,DEG,CCSIZE,PERM)
  MASK(ROOT)=0
 IF(CCSIZE.LE. 1) RETURN
 LVLEND=0
 LNBR=1
 ---------------------------------------------------
 LBEGIN Y LVLEND APUNTAN AL PRINCIPIO Y AL FINAL
 DEL NIVEL EN CUESTION RESPECTIVAMENTE.
 ---------------------------------------------------
```

```
C
 100      LBEGIN=LVLEND+1
         LVLEND=LNBR
         DO 600 I=LBEGIN,LVLEND
C        ------------------------------------------------
C        PARA CADA NODO DEL NIVEL EN CUESTION ...
C        ------------------------------------------------
         NODE=PERM(I)
         JSTRT=XADJ(NODE)
         JSTOP=XADJ(NODE+1)-1
C        ------------------------------------------------
C        ENCUENTRA LOS VECINOS SIN NUMERAR DEL NODO.
C        FNBR Y LNBR APUNTAN AL PRIMER Y ULTIMO VE-
C        CINOS SIN NUMERAR DEL NODO CORRIENTE EN
C        PERM.
C        ------------------------------------------------
C
C
         FNBR=LNBR+1
         DO 200 J=JSTRT,JSTOP
         NBR=ADJNCY(J)
          IF (MASK(NBR) .EQ. 0) GO TO 200
          LNBR=LNBR+1
          MASK(NBR)=0
         PERM(LNBR)=NBR
 200       CONTINUE
          IF(FNBR .GE. LNBR ) GO TO 600
C        ------------------------------------------------
C        SE USA INSERCION LINEAL PARA ACOMODAR LOS VECINOS
C        EN ORDEN CRECIENTE DE LOS GRADOS.
C        ------------------------------------------------
         K=FNBR
 300       L=K
          K=K+1
           NBR=PERM(K)
 400      IF(L .LT. FNBR) GO TO 500
           LPERM=PERM(L)
          IF(DEG(LPERM) .LE. DEG(NBR) ) GO TO 500
          PERM(L+1)=LPERM
          L=L-1
          GO TO 400
 500         PERM(L+1)=NBR
          IF(K.LT.LNBR) GO TO 300
 600          CONTINUE
          IF(LNBR .GT. LVLEND) GO TO 100
C        ------------------------------------------------
C        AHORA SE TIENE EL ORDENAMIENTO CUTHILL-MCKEE
C        EL RCM SE OBTIENE ABAJO.
C        ------------------------------------------------
            K=CCSIZE/2
           L=CCSIZE
          DO 700 I=1,K
          LPERM=PERM(L)
         PERM(L)=PERM(I)
         PERM(I)=LPERM
         L=L-1
 700       CONTINUE
          RETURN
             END
```

GENRCM [/7]

Dada una gráfica G con una o más componentes, la subrutina GENRCM obtiene un ordenamiento para los nodos de G. La numeración se obtiene eligiendo en cada componente un nodo inicial al cual se le denomina ROOT. Un par de nodos pseudo-periféricos son obtenidos por FNROOT de donde se eligirá la nueva raiz. El ordenamiento para dicha componente lo encuentra la subrutina RCM.

XADJ y ADJNCY contienen la estructura de adyacencia de la gráfica. MASK es un arreglo de trabajo que se utiliza para señalar cuales nodos han sido numerados. XLS contiene a los apuntadores para la estructura de nivel que es almacenada en los espacios que no han sido usados por el vector PERM que contiene la permutación. NEQNS es el número de ecuaciones.

```
**************************************************************

.... GENRCM ... CUTHILL MCKEE CON INVERSION  ..........

**************************************************************

PROPOSITO - GENRCM ENCUENTRA  UN ORDENAMIETO PARA LOS
    NODOS DE   UNA GRAFICA EN GENERAL. PARA CADA COMPO-
    NENTE CONEXA DE LA GRAFICA GENRCM ENCUENTRA EL ORDE-
    NAMIENTO RESPECTIVO LLAMANDO AL RCM.

PARAMETROS DE ENTRADA -
   NEQNS - NUMERO DE ECUACIONES.

  (XADJ,ADJNCY) -XADJ CONTIENE LAS POSICIONES DE LOS
  VERTICES EN LA ESTRUCTURA DE ADYACENCIA Y ADJNCY
  CONTIENE LOS NOMBRES DE LOS VERTICES ADYACENTES A
  CADA UNO.
```

```
C
C       PARAMETROS DE SALIDA -
C          PERM - VECTOR QUE CONTIENE EL ORDENAMIENTO RCM.
C
C       PARAMETROS DE TRABAJO -
C          MASK - SE USA PARA ETIQUETAR A LAS VARIABLES QUE HAN
C                 SIDO NUMERADAS DURANTE EL PROCESO EN EL ORDE-
C                 NAMIENTO. SE INICIALIZA EN 1 Y CUANDO UN NODO
C                 HA SIDO NUMERADO SU VALOR EN MASK ES CERO.
C
C          XLS -  VECTOR DE INDICES PARA LA ESTRUCTURA DE NIVEL.
C                 LA ESTRUCTURA DE NIVEL ES ALMACENADA EN LOS
C                  ESPACIOS QUE NO HAN SIDO UTILIZADOS EN PERM.
C
C       SUBRUTINAS DE PROGRAMA -
C       FNROOT,RCM.
C       ***************************************************************
C
        SUBROUTINE GENRCM (NEQNS,XADJ,ADJNCY,PERM,MASK,XLS )
C
C       ***************************************************************
C
        INTEGER ADJNCY(1),MASK(1),PERM(1),XLS(1),XADJ(1),CCSIZE
        INTEGER I,NEQNS,NLVL,NUM,ROOT
C
C       ***************************************************************
C
        DO 100 I=1,NEQNS
        MASK(I)=1
  100      CONTINUE
        NUM=1
        DO 200 I=1,NEQNS
C       -------------------------------------------------
C       PARA CADA COMPONENTE MARCADA CON MASK ...
        IF(MASK(I) .EQ. 0 ) GO TO 200
        ROOT=I
C       -------------------------------------------------
C       PRIMERO ENCUENTRA UN NODO PSEUDO PERIFERI-
C       CO ROOT. NOTESE QUE LA ESTRUCTURA DE NI-
C       VEL ENCONTRADA POR FNROOT ES ALMACENADA
C       EMPEZANDO EN PERM(NUM). ENTONCES EL RCM
C       ES LLAMADO PARA ORDENAR LA COMPONENTE
C       CONEXA USANDO A ROOT COMO NODO INICIAL.
C       -------------------------------------------------
C
        CALL FNROOT (ROOT,XADJ,ADJNCY,MASK,NLVL,XLS,PERM(NUM) )
        CALL RCM ( ROOT,XADJ,ADJNCY,MASK,PERM(NUM),CCSIZE,XLS)
        NUM=NUM+CCSIZE
        IF(NUM .GT. NEQNS) RETURN
  200      CONTINUE
        RETURN
        END
```

INVRSE

La subrutina INVRSE construye el vector unidimensional INVP que de salida contiene la permutación para reordenar la solución $\bar{x}$ del sistema $\bar{A}\bar{x} = b$ y encontrar la solución del sistema original. INVRSE trabaj con el entero N que corresponde al número de ecuaciones y con el vector PERM dado por el RCM para G^A.

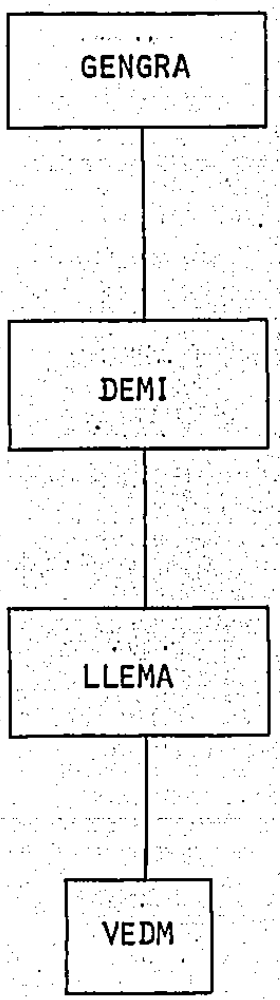
```
C     ************************************************************
C      ... INVRSE .. CONSTRUYE LA PERMUTACION PARA OBTENER LA
C     LA SOLUCION DEL SISTEMA ORIGINAL.
C     ************************************************************
C
C      PARAMETROS DE ENTRADA -
C        PERM - EL VECTOR QUE CONTIENE LA PERMUTACION OBTENIDA
C             POR EL RCM.
C          N - NUMERO DE ECUACIONES.
C
C      PARAMETROS DE SALIDA -
C        INVP - CONTIENE LA PERMUTACION PARA REORDENAR X.
C     ************************************************************
C
        SUBROUTINE INVRSE (N,PERM,INVP)
C
C     ************************************************************
C
        INTEGER  PERM(1),INVP(1),N
C
          DO 100 I=1,N
          INVP(PERM(I))=I
  100     CONTINUE
          RETURN
          END
```

FNENV [/7]

Esta subrutina necesita como entradas a los arreglos XADJ y ADJNCY que contienen la estructura de adyacencia de la gráfica, PERM e INV guardan la información sobre las permutaciones para reordenar $\bar{A}$ y $\bar{x}$ respectivamente y el entero NEQNS que corresponde al número de ecuaciones. Construye el vector XENV que consiste en un conjunto de apuntadores para la estructura de nivel que ha de usarse para almacenar la envolvente superior o inferior de la matriz reordenada. Los enteros positivos ENVSZE y BANDW corresponden respectivamente al perfil y el ancho de banda de $\bar{A}$.

```
C     ***********************************************************
C     ********** FNENV ... ENCUENTRA LA ENVOLVENTE **************
C     ***********************************************************
C
C
C     PROPOSITO - ENCONTRAR LA ENVOLVENTE DE UNA MATRIZ
C                 PERMUTADA.
C
C
C     PARAMETROS DE ENTRADA -
C       NEQNS - NUMERO DE ECUACIONES
C       (XADJ,ADJNCY) - ESTE ARREGLO CONTIENE LA ESTRUCTURA DE
C         LA GRAFICA ASOCIADA A LA MATRIZ.
C       PERM,INV - ARREGLOS QUE CONTIENEN LOS DATOS DE LA PER-
C        MUTACION DE LA MATRIZ REORDENADA.
C
C     PARAMETROS DE SALIDA -
C      XENENV - VECTOR DE INDICES PARA LA ESTRUCTURA DE NIVEL
C       QUE SE USA PARA ALMACENAR LA ENVOLVENTE INFERIOR O
C       SUPERIOR DE LA MATRIZ REORDENADA.
C      ENVSZE - ES IGUAL A XENV(NEQNS+1)-1.
C      BANDW -ANCHO DE BANDA DE LA MATRIZ REORDENADA.
C
C     ***********************************************************
C
      SUBROUTINE FNENV (NEQNS,XADJ,ADJNCY,PERM,INVP,
     1                  XENV,ENVSZE,BANDW)
C
C     ***********************************************************
C
      INTEGER ADJNCY(1),INVP(1),PERM(1),XADJ(1),XENV(1)
      INTEGER BANDW , I ,IBAND,IFIRST,IPERM,J,JSTOP
      INTEGER JSTRT, ENVSZE,NABOR,NEQNS
C
C     ***********************************************************
C
        BANDW = 0
        ENVSZE = 1
        DO 200 I=1,NEQNS
           XENV(I)=ENVSZE
           IPERM=PERM(I)
           JSTRT=XADJ(IPERM)
           JSTOP=XADJ(IPERM+1) -1
           IF(JSTOP.LT.JSTRT) GO TO 200
C
C          ------------------------------------------------
C          ENCUENTRA EL PRIMER ELEMENTO DISTINTO DE CE-
C          RO EN EL RENGLON I.
C          ------------------------------------------------
           IFIRST=I
           DO 100 J=JSTRT,JSTOP
              NABOR=ADJNCY(J)
              NABOR=INVP(NABOR)
              IF(NABOR .LT.IFIRST) IFIRST=NABOR
 100       CONTINUE
           IBAND=I-IFIRST
           ENVSZE = ENVSZE + IBAND
           IF(BANDW.LT.IBAND) BANDW=IBAND
 200       CONTINUE
           XENV(NEQNS+1) =ENVSZE
           ENVSZE=ENVSZE-1
           RETURN
        END
```

C



DEMI

Una gráfica $G = (V,A)$ es *completa*, si para cada

$$u,v \in V(G) \text{ con } u \neq v,\ (u,v) \in A(G)$$

Sea $G' = (V', A')$ con

$$V' = v_i \cup ADY(v_i)$$

y $A' = \{(w,z): w,z \in V' \text{ y } w \neq z\}$

DEMI es una subrutina que opera sobre la estructura de adyacencia (ADJNCY, XADJ), obtenida previamente por las subrutinas GEGRAF o GEGIRC, para calcular el vector DV de dimensión n y el entero DM. El arreglo DV contiene para cada vértice v_i con $i = 1, 2, \ldots, n$; su deficiencia y DM guarda el índice de un vértice con deficiencia mínima. La construcción del vector DV se basa en la idea siguiente:

Sea G' como se definió arriba. La deficiencia de un vértice v_i es el número de aristas que le faltan a G' para que sea una gráfica completa.

```
C      ************************************************************
C      ..................   DEMI.....................................
C      ************************************************************
C      LA SUBRUTINA DEMI CALCULA LA DEFICIENCIA DE LOS VER-
C      TICES Y ELIGE ALGUNO CON MINIMA DEFICIENCIA.
C      ************************************************************
C
C      PARAMETROS DE ENTRADA -
C         (XADJ,ADJNCY) - CONTIENE LA ESTRUCTURA DE ADYACENCIA.
C
C     PARAMETROS DE SALIDA -
C         DV - CONTIENE LA DEFICIENCIA DE LOS VERTICES EN LA
C              GRAFICA ORIGINAL.
C         DM - GUARDA EL INDICE DEL VERTICE CON DEFICIENCIA
C              MINIMA.
C         NEC - NUMERO DE ECUACIONES.
C
C     PARAMETROS AUXILIARES -
C         PI E FI , TIENEN LAS POSICIONES DE EL PRIMER Y ULTI-
C            MO VECINOS DEL VERTICE I RESPECTIVAMENTE.
C         KA Y KB SON VECINOS DEL VERTICE I QUE RECORREN LA VE-
C            CINDAD DE I ; KA ES ANTERIOR A KB.
C
C      ************************************
C
       SUBROUTINE DEMI (XADJ,ADJNCY,DV,DM,NEC)
C      ************************************
C
C
       INTEGER XADJ(1),ADJNCY(1),DV(1),PI,FI,DM
C
C
C      ------------------------------------
C      EL ARREGLO DV DE INICIALIZA EN -1.
C      ------------------------------------
       DO 300 I=1,NEC
        DV(I)=-1
300    CONTINUE
C
C      -------
C      INICIA
C      -------
       DO 100 1=1,NEC
         IF(DV(I).GE.0) GO TO 100
       N=XADJ(I+1)-XADJ(I)
        IF(N.LE.1) GO TO 10
          IF (I.EQ.1) GO TO 80
88          DV(I)=0
```

```
      K=0
      PI=XADJ(I)
      FI=XADJ(I+1)-1
40    L=0
      KA=ADJNCY(PI+K)
30    L=L+1
      M=PI+K+L
      IF(M.GT.FI) GO TO 50
      KB=ADJNCY(M)
      NJ=XADJ(KB+1)-XADJ(KB)
      IF(NJ.LE.1) GO TO 70
      KJ=0
      DO 200 J=XADJ(KB),XADJ(KB+1)-1
      IF(ADJNCY(J).EQ.KA) GO TO 30
      KJ=KJ+1
      IF(KJ.LT.NJ) GO TO 200
      GO TO 20
200   CONTINUE
      GO TO 30
50    K=K+1
      IF(PI+K.GT.FI) GO TO 15
      GO TO 40
20    DV(I)=DV(I)+1
      GO TO 30
70    DV(KB)=0
      DV(I)=DV(I)+1
      IF(DV(KB).LT.DV(DM)) DM=KB
      GO TO 30
10    DV(I)=0
      IF(I.EQ.1) GO TO 80
C     ------------------------------------------------------------
C     ENCUENTRA EL INDICE DE UN VERTICE CON DEFICIENCIA MINIMA
C     ------------------------------------------------------------
      IF(DV(I).GT.DV(DM)) GO TO 100
80    DM=I
      IF(I.EQ.1)GO TO 88
      GO TO 100
15    IF(I.EQ.1) GO TO 100
      IF(DV(I).LT.DV(DM)) DM=I
100   CONTINUE
      RETURN
      END
```

LLEMA

La sucesión de gráficas $\{G_i\}_{i=0}^{n}$, la gráfica G^F y la matriz del llenado F descritas en el capítulo II, secc. 4, son obtenidas por la subrutina LLEMA, que trabaja con el vector de la deficiencia de los vértices DV y el vértice con deficiencia mínima DM, calculados ambos previamente por la subrutina DEMI.

Si la deficiencia del vértice DM es cero, LLEMA modifica únicamente al vector DV, para aquellos vértices que son adyacentes a DM. De otra forma, además de cambiar la estructura del vector de deficiencias DV, llena la matriz. En cualesquiera de los dos casos, se obtiene un nuevo vértice con deficiencia mínima llamando a la subrutina VEDM.

El número LL corresponde al número de elementos a_{ij} de A, que originalmente eran cero y que en esas posiciones en la matriz F son distintas de cero.

```
C     ***********************************************
C     ..............      LLENA     ..................
C     ***********************************************
C       LLENA, EFECTUA EL LLENADO DE LA MATRIZ.
C     ***********************************************
C
C     PARAMETROS DE ENTRADA -
C        A(I,J) - LA MATRIZ ORIGINAL.
C
C        DV - EL VECTOR QUE CONTIENE LAS DEFICIENCIAS
C        DE LOS VERTICES DE LA GRAFICA ORIGINAL.
C        DM- EL INDICE DEL VERTICE CON DEFICIENCIA
C        MINIMA.
C          NEC - NUMERO DE ECUACIONES.
C
C     PARAMETROS DE SALIDA -
C       F=L+LT - LA MATRIZ DEL LLENADO DE A(I,J).
C       LL - EL NUMERO DE ELEMENTOS DISTINTOS DE CERO
C          QUE SE ORIGINARON DURANTE LA ELIMINACION.
C
C      PARAMETROS AUXILIARES -
C         NVE - CUENTA EL NUMERO DE VERTICES ELIMINADOS.
C
C        ***************************************************
C
         SUBROUTINE LLENA (A,DV,DM,LL,NEC)
C
C        ***************************************************
C
         INTEGER A(NEC,NEC),DV(1),DM,VDM
C
C        ***************************************************
C
C
C
C       ---------------------------------------------------------------
C
C       MODIFICA EL VECTOR DE DEFICIENCIAS CUANDO LA DEFICIENCIA
C       DEL VERTICE DM ES CERO.
C       ---------------------------------------------------------------
C
        LL=0
        IF(DV(DM).GT.0) GO TO 90
         DV(DM)=-1
   80    DO 400 IDM=1,NEC
        IF(IDM.EQ.DM.OR.DV(IDM).EQ.-1.OR.A(DM,IDM).EQ.0) GO TO 400
          DO 500 IV=1,NEC
        IF(IV.EQ.IDM.OR.DV(IV).EQ.-1.OR.A(IDM,IV).EQ.0) GO TO 500
           IF(IV.EQ.DM) GO TO 500
          IF(A(DM,IV).EQ.1) GO TO 500
          DV(IDM)=DV(IDM)-1
  500     CONTINUE
  400     CONTINUE
          NVE=NVE+1
           IF(NVE.EQ.NEC-2 ) GO TO 70
        CALL VEDM(DV,DM,NEC)
         IF(DV(DM).EQ.0) GO TO 80
```

```
C     ------------------------------------------------------------------
C     LLENA LA MATRIZ Y MODIFICA LA DEFICIENCIA DE LOS VERTICES
C     QUE SON EXTREMO DE LAS NUEVAS ARISTAS.
C     ------------------------------------------------------------------
C
 90    L=0
       DV(DM)=-1
C     ----------------
C     LLENA LA MATRIZ
C     ----------------
C
       IL=1
 60     DO 100 I=IL,NEC
       IF(I.EQ.DM.OR.DV(I).EQ.-1.OR.A(DM,I).EQ.0) GO TO 100
       L=L+1
       IF(L.EQ.1) GO TO 10
       IF(A(VDM,I).EQ.1) GO TO 100
         A(VDM,I)=1
         A(I,VDM)=1
         LL=LL+2
C     ------------------------------------------------------------------
C     MODIFICA LA DEFICIENCIA DEL VERTICE VDM Y LA DE LOS VECT-
C     NOS COMUNES A VDM E I.
C     ------------------------------------------------------------------
C
       DO 200 J=1,NEC
       IF(VDM.EQ.J.OR.DV(J).EQ.-1.OR.A(VDM,J).EQ.0) GO TO 200
       IF(DM.EQ.J.OR.A(DM,J).EQ.1) GO TO 200
       DV(VDM)=DV(VDM)-1
       IF(A(VDM,J).EQ.A(I,J)) GO TO 50
        DV(VDM)=DV(VDM)+1
        GO TO 200
 10      VDM=I
         GO TO 100
 50    DV(J)=DV(J)-1
         GO TO 200
 200   CONTINUE
C     ------------------------------------------
C     MODIFICA LA DEFICIENCIA DEL VERTICE I
C     ------------------------------------------
       DO 300 K=1,NEC
      IF(I.EQ.K.OR.DV(K).EQ.-1.OR.A(I,K).EQ.0) GO TO 300
         IF(A(VDM,K).EQ.1) GO TO 25
       DV(I)=DV(I)+1
 25      IF(A(DM,K).EQ.1) GO TO 300
       DV(I)=DV(I)-1
          GO TO 300
 300     CONTINUE
       MB=MB+1
        IF(MB.EQ.1) MM=I
       GO TO 100
 100    CONTINUE
        IF(MB.EQ.0) GO TO 15
        IL=MM
         L=0
         MB=0
          GO TO 60
 15        NVE=NVE+1
         IF(NVE.EQ.NEC-2) GO TO 70
       CALL VEDM (DV,DM,NEC)
        IF(DV(DM).EQ.0) GO TO 80
         GO TO 90
 70      RETURN
         END
```

VEDM

Esta subrutina modifica el vector DV, al asignar el valor de -1 en DV(I), si I es una vértice que se ha suprimido. Después encontrará un índice j que corresponda a un vértice con deficiencia mínima y DM toma el valor de j.

```
C     ************************************************************
C     ..........................  VEDM  ..........................
C     ************************************************************
C     LA SUBRUTINA VEDM ENCUENTRA  UN VERTICE CON DEFICIENCI MINI-
C     MA DM , PARA CONSTRUIR LA GRAFICA DE ELIMINACION RESPECTIVA.
C     ************************************************************
C
C     PARAMETROS DE ENTRADA -
C       DV - EL VECTOR QUE CONTIENE LA DEFICIENCIA DE LOS VERTICES.
C       NEC - NUMERO DE ECUACIONES.
C
C     PARAMETROS DE SALIDA -
C       DM - INDICE DEL VERTICE CON DEFICIENCIA MINIMA.
C
C      ************************
C      SUBROUTINE VEDM (DV,DM,NEC)
C      ************************
C
      INTEGER DV(1),DM
       MD=0
         DV(DM)=-1
        DO 100 I=1,NEC
          IF(DV(I).EQ.0) GO TO 30
          IF(DV(I).EQ.-1) GO TO 100
           MD=MD+1
          IF(MD.EQ.1) GO TO 10
          IF(DV(I).LT.DV(DM)) GO TO 10
            GO TO 100
.100    CONTINUE
         GO TO 20
 10       DM=I
         GO TO 100
 30       DM=I
 20       RETURN
          END
```

BIBLIOGRAFIA

[1] Aho, A. V., Hopcraft, J. E., ULLMAN, J. D. *The Design and Analisis of Computer Algoritms*. Addison Wesley, Reading, Mass 1974.

[2] BELLMAN, R., Coke, K. L., Lockett. J. A. *Algorithms, Graphs and Computers*. Editor Richard Bellman. Academic Press 1970.

[3] BERGE. C., *The Theory of Graphs and its Applications* Methuen Wiley. Great Britain 1966.

[4] BRAMELLER, A., ALLAN, R. N., HAMAN, Y. M., *Sparsity* Pitman Publishing 1976.

[5] BUSACKER, R., and T. L. Saaty *Finite Graphs and Networks* :*An Introduction with applications* Mc. Graw Hill New York 1965

[6] CUTHILL, E. "Several Strategies for Reducing the Bandwith of Matrices". De Rose y Willoughby páginas 157-166

[7] CHENG, K. Y., "Minimizing the Bandwith of Sparse Symetric Matrices" *Computing* 2 1973. 103-110

[8] CHENG, K. Y., "Note on Minimizing the Bandwith of Sparse Symetric Matrices" *Computing* 2 1973 27-30

[9] CHINN, P. Z., CHVATALOVA, J., DWDNEY, A. K., GIBBS, N. E., "The Bandwith Problem for Graphs and Matrices A Survey". *J. Graph Theory Vol.* 6 No. 3 Otoño 1982.

[10] DUFF, I.S., *A Survey of Sparse Matrix Research.* Computer Science and Systems Divisions, AERE Harwell. Febrero 1976.

[11] DUFF, I. S., STEWART, G. W., *Sparse Matrices Procedings* SIAM Philadelphia 1978.

[12] DUFF, I. S., *Sparse Matrices and Their Uses* Academic Press, New York. 1981.

[13] Forsythe George E., Malcom Michael A., Moler Cleve. B., *Computer Methods for Mathematical Computations.* Pretince Hall 1977.

[14] GEORGE, J. A., "Direct Solutions of Sparse Positive Definitive Systems: some basic ideas and open problems" *De Duff* 283-306.

[15] GEORGE, J. A., *Computer Implementation of the Finite Element Method*, Ph. D. Thesis, University Microfilms International Ann Arbor, Michigan Londres 1971.

[16] GEORGE A. J., LIU, J. W. H., "A minimal Storage Implementation of the Minimun Degree Algorithm" *SIAM. J. Number Anal Vol. 17 No. 2.* 282-299. Abril 1980.

[17] GEORGE. J. A. LIU, J. W. H., *Computer Solutions of Large Sparse Positive Definite Systems.* Pretince Hall, Englewood Cliffs, N. J. 1981.

[18] GIBBS, N. E., POOLE, JR. N. G., STOCKEMEYER, P. K., "An Algorithm for Reducing the Bandwith and the Profile of Sparse Matrix". *SIAM. J. Number An* 13. 236-250. 1976.

[19] Harary, F., *Graph Theory.* Addison Wesley, Reading M. A. Segunda Edición 1971.

[20] HERNANDEZ, D. B., *Análisis Numérico: Discretización y Algoritmica.* Notas del 3º Coloquio del Departamento de Matemáticas del CIEA, IPN. La Trinidad Tlaxcala, México Cd. de México 1983.

[22] JENNINGS. A., *Matrix Computation for Engineers and Scientists.* John Wiley and Sons 1980.

[23] Knuth. D. E., *The art of Computing Programming.* Addison Wesley P. C. 1968.

[25] LIU. J. W. H., "On Reducing the Profile of Sparse Symetric Matrices". Rept. C. S. 76-07 Departament of Computer Science, Univ of Waterloo, Febrero 1976.

[26] LEWIS, J. G., "Implementation of Gibbs-Poole Stockmeyer and Gibbs - King Algorithms" A.C.M. *Transactions on Mathematical Soltware* Vol. 8. No. 2 180-189. Junio 1982.

[27] PAPADIMITRIOU, C. H., "The NP-Completeness of the Bandwith Minimization Problem", Computing 16. 263-270 1976.

[28] RICE, J. R. *Numerical Methods, Software and Analysis.* International Student Edition 1983.

[30] ROSE. J. D., WILLOUGHBY. R. A., *Sparse Matrices and Their Applications*. Plenum Press. New York. London. 1972.

[31] ROSE. J. D. "Algorithmic Aspects on Vertex Elimination on Graphs". *SIAM J. Computing Vol. 5 No. 2.* 226-283 Junio 1976.

[32] Schwarz H. R., Rutishauser H., Stiefel. E., *Numerical Analysis of Symmetric Matrices* Pretince Hall, INC. 1973.

[33] TEWARSON, R. P., *SPARSE MATRICES*. Mathematics in Science and Engineering Vol. 99 Academic Press 1973.