

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE CIENCIAS

"METODO DE LANCZOS PARA SISTEMAS LINEALES  
GRANDES, HUECOS Y SIMETRICOS".

T E S I S

QUE PARA OBTENER EL TITULO DE:

M A T E M A T I C O

P R E S E N T A

ARMANDO SAAVEDRA ESPINOSA



## **UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso**

### **DERECHOS RESERVADOS © PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL**

Todo el material contenido en esta tesis está protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

## INDICE

0. INTRODUCCION	1
I. PLANTEAMIENTO DEL PROBLEMA E INTERPRETACION GEOMETRICA	3
1.1 PLANTEAMIENTO MATEMATICO DEL PROBLEMA	4
1.2 INTERPRETACION GEOMETRICA	6
1.3 OBJETIVOS DEL TRABAJO	11
II. METODO DE MINIMIZACION POR ITERACION	12
2.1 VIA COLUMNAS DE A	13
2.2 VIA PSEUDO-HILF	25
2.3 FACTORIZACION LU DE T	37
2.4 SOLUCION ITERATIVA DEL PROBLEMA	43
2.5 ALGORITMO MATEMATICO MIMLG	48
2.6 ALGORITMO INFORMAL MIMLG	51
III. IMPLEMENTACION DEL ALGORITMO MIMLG	54
3.1 PRESENTACION DEL ALGORITMO	54
3.2 PROGRAMACION	56
3.3 PRUEBAS	57
IV. CONCLUSIONES	62
V. APENDICE	63
VI. BIBLIOGRAFIA	74

## G. INTRODUCCION

Uno de los problemas mas antiguos en las matematicas, ha sido el resolver sistemas de ecuaciones lineales. Se cree que los antiguos egipcios en el siglo III antes de cristo ya utilizaban estos para calcular areas, debido, a que, al desbordarse el Nilo borraba las divisiones de terreno. Todo esto se desprende de los estudios realizados en el papiro del Rhinn (Ponencia presentada en el II -Seminario CORNEX Tax. Gro. México 1982).

En epocas mas recientes el gran matematico aleman Gauss, estudio este problema desarrollando un metodo para resolverlo, el cual es conocido como eliminacion gaussiana, y con algunas modificaciones es de los metodos mas comunes y utilizados hoy en dia.

En la decada de los cuarentas la llegada de la computadora, marco el inicio de una nueva era para las Matematicas, y muy en especial para el analisis Numerico. El cual ha tenido un auge impresionante, debido a su gran utilidad en la solucion de problemas que no habian sido abordados por los matematicos, a causa de su gran tamaño, ya que este es un indicador de la cantidad de operaciones que son necesarias para llegar a la solucion de estos.

En particular una de las areas de la Matematica mas favorecidas con todo esto ha sido, las Ecuaciones Diferenciales, tanto Ordinarias como Parciales, en las cuales es dificil, sino que imposible, en una gran cantidad de casos obtener una solucion analitica de ellas. Uno de los metodos para resolver este tipo de ecuaciones es el aproximar las derivadas que estan involucradas

por medio de diferencias finitas; por ejemplo:

$$y' = (y(x+h) - y(x)) / h$$

$$y'' = (y(x-h) - 2y(x) + y(x+h)) / h^2$$

etc.

Dando origen a sistemas de ecuaciones de gran tamaño con una cantidad grande de elementos iguales a cero, a los cuales se les conoce como Sparse, Huecos, Ralos, etc.

De todo esto se desprende la necesidad de resolver este tipo de sistemas, tratando de economizar tanto en tiempo de computo como en memoria.

El presente trabajo esta enfocado a la solución de sistemas Simétricos, Grandes y Huecos.

En el primer capítulo se plantea el problema de sistemas de ecuaciones lineales y su interpretación geométrica, aclarando el significado de las definiciones de Simétrico, Grande y Hueco.

En el segundo capítulo presentaremos y explicaremos un método para resolver dichos sistemas, presentando al final un algoritmo en lenguaje informal de dicho método conocido como Método de Iteraciones Minimizadas.

En el tercer y último capítulo se implanta en forma práctica el algoritmo para sistemas de ecuaciones con estructura.

# I. PLANTEAMIENTO MATEMATICO DEL PROBLEMA E INTERPRETACION GEOMETRICA

En el presente trabajo estaremos interesados en resolver mediante una maquina digital un sistema de ecuaciones lineales con elementos reales y con las características siguientes:

1).- El número de ecuaciones como de variables es el mismo y ademas es "grande".

2).- Los coeficientes de las variables  $j$ -ésima e  $i$ -ésima en las ecuaciones  $i$ -ésima y  $j$ -ésima respectivamente son iguales (Sistema Simétrico).

3).-El porcentaje de coeficientes distintos de cero es "pequeño".

De la primera se deduce que para obtener la solución se requiere de una gran cantidad de trabajo.

Las otras dos implican la utilización de poca memoria.

Por otra parte nuestro método deberá tener las siguientes propiedades:

i).-Ser iterativo.

Se busca un método iterativo en base a la idea que en  $K$ -iteraciones (con  $K$  menor que  $n$ ) tengamos ya una solución aproximada, la cual no sería mejorada sustancialmente al continuar el proceso.

ii).-No aumentar significativamente nuestras necesidades de memoria.

Si utilizamos una gran cantidad de memoria adicional, se pierde una cualidad importante de nuestro sistema (esto es, la poca densidad) y aumenta el costo para obtener la solución aproximada.

## 1.1 PLANTEAMIENTO MATEMATICO DEL PROBLEMA

Pensemos en el sistema de n-ecuaciones con n-incognitas.

$$a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1$$

$$a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n = b_2$$

.....

.....

$$a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n = b_n$$

Donde  $a_{ij} \in \mathbb{R}$  para  $i=1,2,\dots,n$ ,  $j=1,2,\dots,n$

y  $x_i \in \mathbb{D}$  para  $i=1,2,\dots,n$

con las propiedades siguientes:

1).- n "grande".

2).-  $a_{ij} = a_{ji}$  (Simetria).

3).- El porcentaje de las  $a_{ij} \neq 0$  es "grande"

Observese que las  $a_{ij}$  y  $b_i$  son dadas.

El termino n "grande" (denotado  $n \gg 1$ ) es una idea subjetiva y poco precisa. Para nuestros fines consideraremos n "grande" si  $n \geq 100$ .

De manera analoga tomaremos como porcentaje "grande" si es mayor o igual a un 80%.

Por comodidad utilizaremos la notación matricial de nuestro problema esto es:

$$\left\{ \begin{array}{l} Ax=b \\ I \\ A=A \\ n \geq 1 \\ A \text{ hueca.} \end{array} \right. \quad (1.1)$$



## 1.2 INTERPRETACION GEOMETRICA

Para dar dicha interpretacion veamos algunos conceptos y notaciones.

Sea  $A = [a_{ij}]$ , donde

$$a_j = \begin{bmatrix} a_{1j} \\ a_{2j} \\ \vdots \\ a_{nj} \end{bmatrix}$$

El subespacio generado por las columnas de la matriz  $A$  (denotado por  $S(A)$ ) se define como:

$$S(A) = \{x \in \mathbb{R}^n / x = \sum_{i=1}^n \theta_i a_i, \theta_i \in \mathbb{R}\}$$

El conjunto  $\{u_1, u_2, \dots, u_k\}$

es linealmente independiente si y solo si

$$\sum_{i=1}^k \theta_i u_i = 0 \implies \theta_i = 0 \text{ para toda } i$$

La dimension del subespacio  $S(A)$  es el numero de columnas de la matriz  $A$  que son linealmente independientes y la denotaremos como  $\text{DIM } S(A)$ .

Diremos que la matriz  $A$  es de rango maximo si  $\text{DIM } S(A) = n$ , en caso contrario diremos que es de rango deficiente.

El vector 0, es el vector:

$$0 = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

Al conjunto de vectores  $x$  tales que  $Ax=0$ , se le conoce como nucleo de  $A$  denotado( $N(A)$ ).

En un curso basico de Algebra Lineal se ve el siguiente resultado. (Alternativa de Fredholm)

i).- Si  $S(A)=\mathbb{R}^n$  entonces el sistema (I.1) tiene solucion unica, para toda  $b \in \mathbb{R}^n$  dada.

ii).- Si  $b \in S(A) \not\subseteq \mathbb{R}^n$  entonces el sistema (I.1) tiene una infinidad de soluciones.

iii).- Si  $b \notin S(A)$  entonces el sistema (I.1) no tiene solucion.

Para mayor claridad dividiremos la interpretacion en tres casos.

i).-  $\text{DIM } S(A)=n$

En este caso existe solucion unica para el problema (I.1) por el teorema anterior.

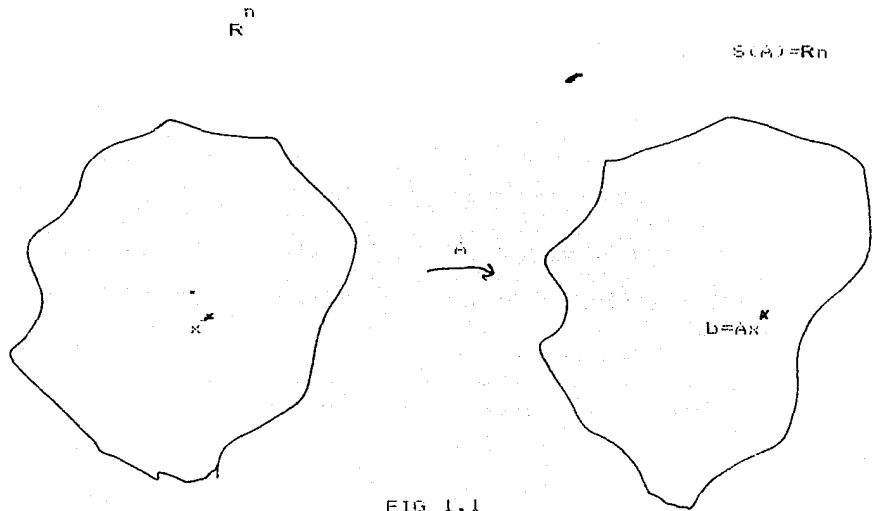


FIG 1.1

$$b = Ax^k$$

$$x^k = \sum_{i=1}^n a_{ii}$$

2).-  $\dim S(A) = k < n$  y  $b \in S(A)$

Existe solución para el problema (1.1) pero no es única en virtud del resultado anterior.

$\mathbb{R}^n$

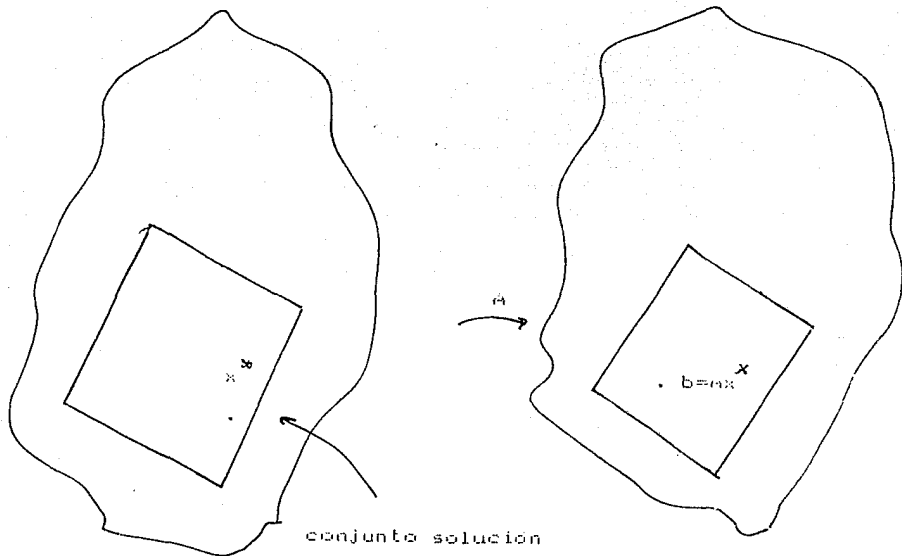


FIG 1.2

$$Ax = b, \quad x = \sum_{i=1}^k \theta_i a_i + \sum_{i=k+1}^n \delta_i v_i$$

donde  $v_i \in N(A)$  para  $i=k+1, k+2, \dots, n$

3).- DISEÑO DE UN SISTEMA DE CONTROL

En este caso unico no existe solución por el resultado mencionado pero podríamos estar interesados en encontrar  $x$  solución aproximada al problema (1.1).

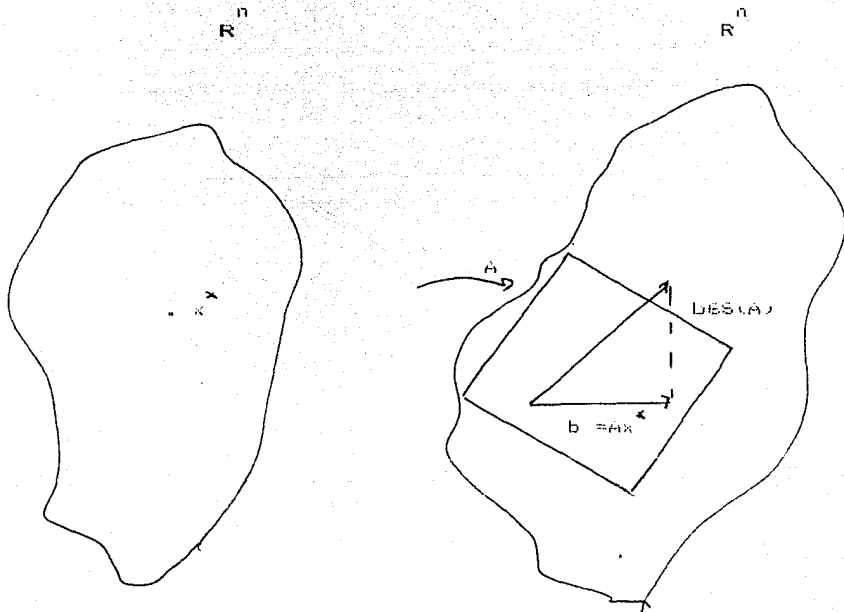


FIG 1.3

donde  $x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$

### I.3.- OBJETIVOS DEL TRABAJO

1).- Discusion de un metodo para resolver (1.1)

2).- Implementacion de este en una maquina digital

utilizaremos FORTRAN IV como lenguaje de programacion (y usaremos una maquina B7800).

3).- Representar el metodo en un algoritmo "eficaz" para resolver (encontrar una solucion aproximada) nuestro problema (1.1).

4).- Comprobacion del metodo con algunas matrices de estructura especial.

## II. METODO DE MINIMIZACION POR ITERACION

Una hipótesis importante respecto del método que desarrollaremos en este capítulo es que en el sistema (I.1) se tenga que  $bES(A)$ . (Rango Completo).

La idea general del método es la siguiente:

Tomar o construir en forma iterativa

1).- Vectores linealmente independientes en  $R^n$ , digamos

$$U = \{u_1, u_2, \dots, u_i\}$$

2).- Transformar  $U$  en  $V = \{v_1, v_2, \dots, v_i\}$

colección de vectores ortonormales ( $q_1, q_2, \dots, q_k$ ).

3).- Calcular  $\rho_i = \min \left\| b - \sum_{j=1}^n \theta_j^{(i)} q_j \right\|_2$

hasta que  $bES(U_k)$

Observese que  $bES(U_k)$  cuando  $\rho_k = 0$  y  $k \leq n$

Para luego, finalmente, usar estos nuevos vectores en la resolución del sistema (I.1). Mediante el cambio de base

$\{q_1, q_2, \dots, q_i\}$  a la Base  $\{u_1, u_2, \dots, u_i\}$ .

## II.1 VIA COLUMNAS DE A

$n \times n$

Recordemos que nuestro problema es  $AX=B$ ,  $A \in \mathbb{R}^{n \times n}$ ,

simétrica, "grande", y hueca.

Como necesitamos que  $B \in S(U)$  para alguna  $U$ , colección de vectores linealmente independientes, es natural pensar a esta como las columnas de la matriz  $A$ , ya que son vectores dados en nuestro problema (I.1).

Procedamos entonces a emplear el algoritmo de Minimización por Iteración.

Para la presentación recordemos un resultado elemental:

Sean  $\{v_1, v_2, \dots, v_r\}$  una colección de vectores ortonormales, en  $\mathbb{R}^n$  y  $u \in \mathbb{R}^n$  entonces existen  $\alpha_1^0, \alpha_2^0, \dots, \alpha_r^0$

tales que:

$$\|u - \alpha_1^0 v_1 - \alpha_2^0 v_2 - \dots - \alpha_r^0 v_r\|_2 \text{ es mínimo.}$$

Más aun  $\alpha_i^0 = \text{COMP}_i u = \langle u, v_i \rangle$  (Componente de  $u$  en la dirección de  $v_i$ )

Sea  $A = [a_1, a_2, \dots, a_k]$

tomemos  $U = A$

$$p_i = \min \|b - \alpha_1^0 a_1 - \alpha_2^0 a_2\|_2$$



Ortonormalizemos  $U_i$ , para hacer esto basta con tomar

$$q_i = a_i / \|a_i\|_2, \text{ con lo cual se ha transformado el problema}$$

de minimizar como:

$$J_i = \min_{\beta_i} \|b - \beta_i q_i\|_2^2$$

donde el mínimo se alcanza con  $\tilde{\beta}_i = \langle b, q_i \rangle$

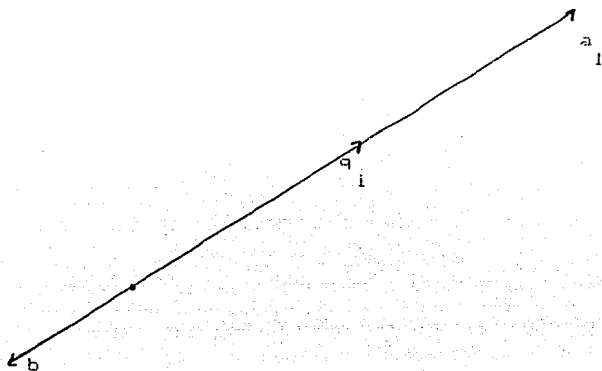
Si  $J_i = 0$  entonces el algoritmo termina, obteniendo que

$$b \in \text{span}(U_i) \text{ y } b = \tilde{\beta}_i q_i = \tilde{\beta}_i a_i / \|a_i\|_2$$

La solución al problema (1.1) sería entonces:

$$x = \begin{bmatrix} \tilde{\alpha}_1(w) \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

ver la fig siguiente.



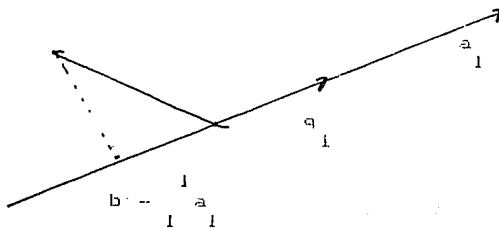
BES(U)  
i

FIG 2.1

Si  $p_i \neq 0$ , quiere decir que  $b \in \text{Span}(a_i)$  entonces tomaremos como solución aproximada a (1.1) el vector

$$x_i = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

y  $b$  tal que  $Ax=b$



be  $S(U_1)$

FIG 2.2

Procedamos al siguiente paso haciendo:

$$U_2 = [a_1, a_2]$$

y planteandonos

$$f_2 = \min_{\alpha_2} \| b - \alpha_1 a_1 - \alpha_2 a_2 \|^2$$

Resolver este problema no es tan facil como el de la iteracion anterior, es por esta causa que recurrimos a la ortonormalizacion de  $U_2$ , lo cual se efectuara mediante el conocido proceso de GRAMM-SCHMIDT [1]. El cual produce dos Matrices

$Q \in \mathbb{R}^{n \times r}$  con  $Q^T Q = I$  y  $R \in \mathbb{R}^{r \times r}$  Triangular Superior tales que  $A = QR$

Tomemos la factorización QR de  $A_2$

$$[a_{i2} \quad a_{j2}] = [q_{i2} \quad q_{j2}] \begin{bmatrix} \gamma_{11} & \gamma_{12} \\ 0 & \gamma_{22} \end{bmatrix}$$

donde  $\gamma_{11} = a_{i2}$

$$\gamma_{12} = \frac{a_{j2} \cdot q_{i2}}{a_{i2}}$$

$$\gamma_{22} = \frac{a_{j2} - \gamma_{12} q_{j2}}{q_{j2}}$$

El problema de minimización se puede expresar en forma matricial como:

$$\rho = \min_{\alpha} \left\| b - [a_{i2} \quad a_{j2}] \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} \right\|_2^2$$

observemos la expresión

$$[a_{i2} \quad a_{j2}] \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}$$

Utilizando la factorización anterior

$$\begin{aligned}
 \begin{bmatrix} 1 & a \\ 1 & a \end{bmatrix} \begin{bmatrix} \alpha \\ \alpha \end{bmatrix} &= \begin{bmatrix} 1 & a \\ 1 & a \end{bmatrix} \begin{bmatrix} \gamma & \alpha \\ 0 & \alpha \end{bmatrix} \begin{bmatrix} \alpha \\ \alpha \end{bmatrix} \\
 &= \begin{bmatrix} 1 & a \\ 1 & a \end{bmatrix} \begin{bmatrix} \gamma \alpha + \alpha \alpha \\ \alpha \alpha \end{bmatrix} \\
 &= \begin{bmatrix} 1 & a \\ 1 & a \end{bmatrix} \begin{bmatrix} \beta \\ \beta \end{bmatrix}
 \end{aligned}$$

Finalmente con este desarrollo el problema de minimización queda expresado como:

$$f_2 = \min_{\beta_2} \| b - \beta_2^1 a - \beta_2^2 a \|_2^2$$

$$\text{obteniendo } \hat{\beta}_2^1 = \text{b.q.}_1$$

$$\hat{\beta}_2^2 = \text{b.q.}_2$$

$$\text{Si } f_2 = 0 \text{ entonces } b = \hat{\beta}_2^1 a + \hat{\beta}_2^2 a$$

$$\text{además } \hat{\beta}_2^1 = \gamma \hat{\alpha}_2^1$$

$$\hat{\beta}_2^2 = \gamma \hat{\alpha}_2^1 + \gamma \hat{\alpha}_2^2$$

encontrando que

$$\tilde{a}_2^1 = (\tilde{\beta}_2^1 - \gamma_{12} \tilde{\beta}_2^2, \gamma_{22}, \gamma_{11})$$

$$\tilde{a}_2^2 = \tilde{\beta}_2^2, \gamma_{22}$$

lo cual expresa

$$b = \tilde{a}_2^1 a_{11} + \tilde{a}_2^2 a_{22}$$

La solución al problema (I.1) sería

$$x = \begin{bmatrix} \tilde{a}_2^1 \\ \tilde{a}_2^2 \\ 0 \\ \cdot \\ \cdot \\ 0 \end{bmatrix}$$

es conveniente ver la siguiente figura

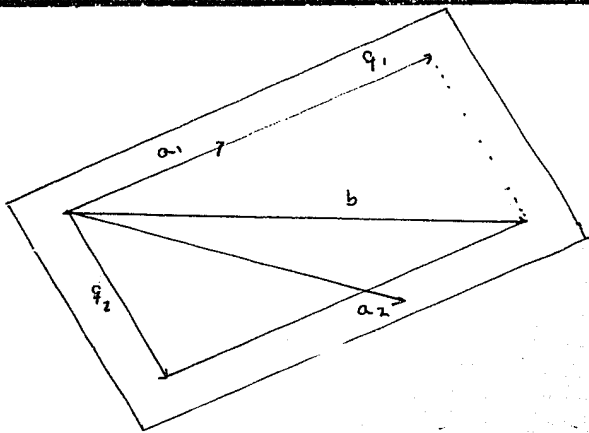
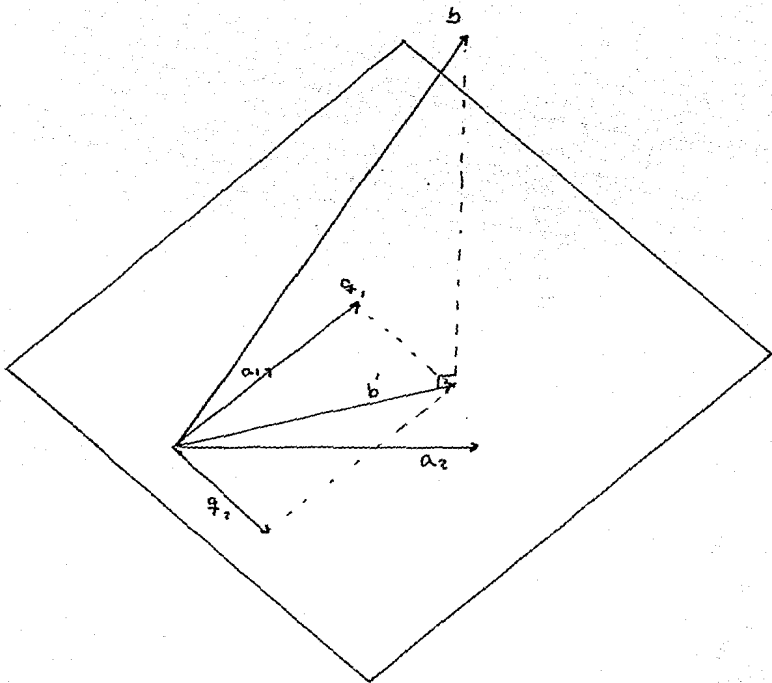


FIG 2.3

Si  $\alpha_2 = 0$  entonces  $\text{BES}(U)$ , pero tomaremos como solución aproximada

$$K^2 \begin{bmatrix} \alpha_2^{(1)} \\ \alpha_2^{(2)} \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix}$$

y  $b = Ax$  (ver fig 2.4)



$b \in S(U)$  y  $f \neq 0$

FIG 2.4

En el paso  $k$ -ésimo  $U = \{a_1, a_2, \dots, a_k\}$



$$p = \min_{d \in \mathbb{R}^k} \| b - \alpha_1 a_1 - \alpha_2 a_2 - \dots - \alpha_k a_k \|^2$$

en notación matricial, resolver

$$p = \min_{d \in \mathbb{R}^k} \| b - A d \|^2$$

para ello necesitamos la factorización QR de A

$$A = Q R$$

Utilizando el resultado de álgebra lineal:

$$S(A) = S(q_1, q_2, \dots, q_k) = S(Q)$$

Así como el hecho de que  $b - A \alpha$  es ortogonal a  $Q$ , por lo

tanto se sigue que:

$$Q^t (b - A \alpha) = 0$$

luego, sustituyendo A por su factorización QR, se tiene que

$$Q^t (b - Q R \alpha) = 0$$

distribuyendo

$$Q^t b - R \alpha = 0$$

puesto que  $Q^t Q = I$ :

de donde necesitamos unicamente resolver

$$R \alpha_k = Q_k^t b_k$$

con

$$Q_k \in R^{n \times k}, \quad R_k \in R^{k \times k} \text{ triangular superior;}$$

ya que  $\tilde{\beta}_k^i = \langle b, q_k \rangle$  y las  $\tilde{\alpha}_k^i, i=1, k,$

se obtienen por sustitucion hacia atras. Obsérvese que las

$\tilde{\beta}_k^i$  son precisamente  $Q_k^t b_k$  y las  $\tilde{\alpha}_k^i$  se obtienen a partir de ellas

El proceso puede ir aumentando columnas de la matriz A.

Notese que en a lo mas n-iteraciones el proceso termina y el sistema (I.1) tiene solucion si  $b \in S(A)$ , en caso contrario se ha obtenido una solucion aproximada.

Este es un algoritmo que nos da la seguridad de encontrar una solucion a nuestro problema planteado en (I.1) pero observese que intrinsecamente estamos efectuando la factorizacion QR de la matriz A, la cual adolece en ciertos aspectos a saber:

Q matriz "grande" y llena.

R matriz "grande" y triangular superior.

con lo cual se ha perdido una de las cacteristicas mas importantes del metodo deseado, en el sentido de no tener grandes requerimientos de memoria.

Formar  $Q$  no es caro, ya que, en iteraciones anteriores se ha calculado

$$q_1^t, q_2^t, \dots, q_{k-1}^t, b^t.$$

El problema radica en  $R$ , ya que esta matriz es llena y puede requerir  $1/2 n^2$  entradas distintas de cero, en lo que si  $n=1000$  y  $A$  es una matriz en banda con un ancho de 11 requiere a lo mas 11000 entradas distintas de cero.

En el paso  $k$ -esimo,  $R$  requiere  $1/2 k^2$ , sup  $k=200$  serian 20000 entradas que ya rebasa en 9000 a lo necesario por  $A$ .

En la iteracion  $k=500$  requeriria 125000, lo cual, ya es considerable. Si se hicieran mil iteraciones es necesario 500000. comparadas con las 11000 que requeria  $A$  no es nada al lado de lo necesario para  $R$ .

### "CLARAMENTE POR AQUI NO ES EL CAMINO"

Todo esto nos lleva a desechar el procedimiento de tomar como  $U_k$  las primeras  $k$ -columnas de la matriz  $A$ .

Esto es busquemos otra via para la minimización por iteración, ya que no podemos renunciar a un proceso iterativo por ser  $n$  "grande".

Observese tambien que no hemos aprovechado el hecho de que  $A$  es simétrica. En la siguiente seccion trataremos de aprovechar al máximo esta propiedad.



Veamos que  $AQ=QT$ , implica que

$$AQ = QT$$

$$= [q_1, q_2, \dots, q_n] \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

de donde

$$Aq_i = \alpha_i q_i + \beta_i q_2$$

multiplicando por  $q_i^t$ , por ser  $q_1$  y  $q_2$  ortogonales se tiene

$$q_i^t A q_i = \alpha_i q_i^t q_i + \beta_i q_i^t q_2$$

$$= \alpha_i$$

ademas  $\beta_i q_2 = (A - \alpha_i I) q_i$

tomando  $R = (A - \alpha_i I) q_i$

y  $\beta_i = \|R\|$

entonces  $q_2 = R / \beta_i$

de la misma forma

$$Aq = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \beta_3 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} \alpha_1 & \alpha_2 & \dots & \alpha_n \end{bmatrix} \begin{bmatrix} \beta_2 \\ \alpha_2 \\ \beta_3 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

obteniendo

$$Aq = \beta q + \alpha q + \beta q$$

multiplicando por  $q^t$

$$q^t Aq = \beta q^t q + \alpha q^t q + \beta q^t q$$

$$= \alpha$$

Por otra parte

$$\beta q = (A - \alpha I)q - \beta q$$

sea  $K = (A - \alpha I)q - \beta q$

y  $\beta = \|K\|$

por lo tanto  $q = R / \beta$

En general, para la  $j$ -ésima columna

$$Aq_j = [q_1, q_2, \dots, q_n]^T \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \beta_j \\ \alpha_j \\ \beta_{j+1} \\ \vdots \\ 0 \end{bmatrix}$$

$$Aq_j = \beta_j q_{j-1} + \alpha_j q_j + \beta_{j+1} q_{j+1}$$

Análogamente multiplicando por  $q_j^t$

$$q_j^t Aq_j = \beta_j q_j^t q_{j-1} + \alpha_j q_j^t q_j + \beta_{j+1} q_j^t q_{j+1}$$

$$= \alpha_j$$

y tomemos

$$R_{j+1} = CA - \alpha_j I_j q_j - \beta_j q_{j-1}$$

$$\|R_{j+1}\|_{j+1, 2}$$

De lo cual

$$q_{j+1} = R_{j+1} / q_j \quad j=2, \dots, n-1$$

En la última columna

$$Aq = \begin{bmatrix} q_1 & q_2 & \dots & q_n \\ \beta_1 & & & \\ & \beta_2 & & \\ & & \dots & \\ & & & \beta_n \\ & & & \alpha_n \end{bmatrix}$$

$$Aq_n = \beta_n q_{n-1} + \alpha_n q_n$$

obteniendo

$$q_n Aq_n = \beta_n q_{n-1} q_n + \alpha_n q_n q_n$$

$$= \beta_n$$

Y en este momento hemos terminado nuestra tridiagonalización.

Resumamos lo anterior en un algoritmo en lenguaje informal matemático.



ALGORITMO PARA OBTENER Q-T DE LA FACTORIZACION  $AQ=QT$  TAL

QUE  $Q^t Q = I$ ,  $q_{i+1}^t q_i = 0$ ,  $T$  TRIDIAGONAL

$$q_0 = 0, \quad q_i = b / \beta_i, \quad \beta_i = \|b\|_2, \quad i=1$$

Mientras  $\beta_{i+1} \neq 0$

$$\left\{ \begin{array}{l} \alpha_i = q_i^t A q_i \\ R_{i+1} = (A - \alpha_i I) q_i - \beta_i q_{i-1} \\ \beta_{i+1} = \|R_{i+1}\|_2 \\ q_{i+1} = R_{i+1} / \beta_{i+1} \\ i = i + 1 \end{array} \right.$$

Este es el conocido algoritmo de Lanczos para tridiagonalizar una matriz simetrica. Los vectores  $q_j$  son llamados vectores de Lanczos. [3]

Observese que:

$$q_1 = A e_1$$

$$A q_1 = A^2 e_1$$

$$= A^3 e_1$$

puesto que  $AQ=QT$

$$A^2 q_i = A Q T e_i$$

$$= A Q T e_i$$

$$= Q T^2 e_i$$

en general

$$A^{j+1} q_i = A^j A q_i$$

$$= A^j Q T e_i$$

$$= Q T^{j+1} e_i$$

de donde se obtiene

$$[q_i, A q_i, \dots, A^{n-1} q_i] = [Q e_i, Q T e_i, \dots, Q T^{n-1} e_i]$$

factorizando  $Q$

$$[q_i, A q_i, \dots, A^{n-1} q_i] = Q [e_i, T e_i, \dots, T^{n-1} e_i]$$

La sucesión  $q_i, A q_i, \dots, A^{n-1} q_i$

es llamada "sucesión de Krilov" generada por el vector  $q_i$  y la matriz  $A$ , y la denotaremos a la matriz (Matriz de Krilov) como:

$$[q_i, A q_i, \dots, A^{n-1} q_i] = K(A q_i, n)$$

Luego entonces

$$K(Aq_i, n) = BK(T, e_i, n)$$

La matriz  $K(T, e_i, n)$  es una matriz triangular superior como se muestra a continuación.

$$T e_i = \begin{bmatrix} \beta_1 \\ \alpha_2 \\ 0 \\ \cdot \\ \cdot \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

y lo denotaremos por

$$\begin{bmatrix} 1 \\ c_1 \\ 1 \\ c_2 \\ 0 \\ \cdot \\ \cdot \\ 0 \end{bmatrix}$$

$$T e_i = T T e_i$$





La matriz  $K(Aq, n)$  no tiene la propiedad que la  $IM(K(Aq, n))$  este contenida en la  $IM(A)$  a menos que  $q \in IM(A)$  o bien  $R(A) = n$ .

Tomemos la matriz:

$$U = AK(Aq, k) = [a_1, a_2, \dots, a_k]$$

La  $IM(U) \subset IM(A)$  y suponemos  $b \in IM(A)$ , siendo no necesariamente que  $b \in IM(U)$ .

Esto nos induce a tomar como  $q = b/b$  y de esta forma asegurar que  $b \in IM(U)$ .

Por el momento veamos si esta elección de  $U$  nos reduce la cantidad de memoria y computo en el método de iteraciones con minimización.

$$f = \min_{k \in \mathbb{R}^k} \|b - U \theta\|_2^2$$

$$= \min_{k \in \mathbb{R}^k} \|b - AK(A, q, k) \theta\|_2^2$$

Utilizando la factorización QR de  $K(A, q, k)$

$$f = \min_{k \in \mathbb{R}^k} \|b - QR(K(T, e, k)) \theta\|_2^2$$

utilizando el resultado

$$QR = \begin{bmatrix} Q & 0 \\ 0 & I \end{bmatrix} + r \begin{bmatrix} e \\ 0 \end{bmatrix}$$

$$r = 0 \text{ cuando } q = 0 \quad [4]$$

se obtiene:

$$f = \min_{k \in \mathbb{R}^k} \|b - (Q \begin{bmatrix} I & r \\ 0 & I \end{bmatrix} + e) K(T, e, k) \theta\|_2^2$$

ya que  $b - (Q \begin{bmatrix} I & r \\ 0 & I \end{bmatrix} + e) K(T, e, k)$  es

ortogonal al generado por los  $q_i$  entonces

$$Q_k^t (b - Q_k^t T_k + r_k e_k) - K_k(T_k, e_k, k) \theta = 0$$

distribuyendo

$$Q_k^t b - Q_k^t Q_k^t T_k K_k(T_k, e_k, k) \theta + Q_k^t r_k e_k - K_k(T_k, e_k, k) \theta = 0$$

Utilizando el resultado mencionado y despreciando el tercer termino

$$Q_k^t b - T_k K_k(T_k, e_k, k) \theta = 0$$

por lo tanto

$$T_k K_k(T_k, e_k, k) \theta = Q_k^t b$$

y haciendo  $y = K_k(T_k, e_k, k) \theta$

$$T_k y = Q_k^t b$$

El problema a resolver es  $Ax = b$  pero  $AQ_k^t = Q_k^t T_k$  coinciden

cuando  $q_{k+i} = 0$ , es decir podemos tomar  $A = Q_k^t T_k$

Por lo tanto el sistema queda expresado como  $Q_k^t T_k x = b$

luego entonces  $T_k x = Q_k^t b$

de donde se deduce que:

$$x = Q_k^t y$$

## II.3 FACTORIZACION LG DE T

Hasta el momento se ha observado que la tridiagonalizacion de Lanczos es util para resolver el sistema

$$Ax=b$$

A simple vista parece que no hay mucha ganancia por este camino, ya que, obtener  $Q^t b$  no es mucho problema puesto que en pasos anteriores se puede obtener:

$$q_1^t b, q_2^t b, \dots, q_{k-1}^t b$$

el problema radica en  $q_k^t b$

$$y_k = Q_k^t y$$

En la presente y siguiente seccion atacaremos estos problemas.

Esto nos lleva a la necesidad de resolver

$$\Gamma_k y = Q_k^t b$$

donde  $\Gamma_k$  es una matriz tridiagonal simetrica.

Paige y Saunders presentaron en 1974, un algoritmo basado en la factorizacion ortogonal de  $\Gamma_k$  mediante rotaciones de Givens, para resolver el problema anterior [6].

A continuacion presentaremos en detalle dicho algoritmo.

La factorizacion  $\Gamma_k = L G$  puede ser obtenida mediante la multiplicacion de matrices ortonormales  $G$  donde  $G_{i,i+1}$  difiere de la identidad solo en los elementos:

$$g_{i,i} = \cos \theta, \quad g_{i,i+1} = -\sin \theta$$

$$y \quad g_{i,i+1} = \sin \theta, \quad g_{i+1,i} = \cos \theta$$





y deseamos que  $\gamma = 0$

$$\begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix} = \begin{bmatrix} c \\ \alpha \end{bmatrix}$$

conociendo que  $c_1^2 + s_1^2 = 1$

$$\begin{bmatrix} \delta_1 & 0 \\ \delta_2 & \alpha \end{bmatrix} = \begin{bmatrix} \delta_1 \\ \beta_2 \end{bmatrix} \begin{bmatrix} c_1 & -s_1 \\ s_1 & c_1 \end{bmatrix}$$

entonces  $-\delta_1 s_1 + \beta_2 c_1 = 0$

despejando  $s_1$  y  $c_1$

$$s_1 = \beta_2 c_1 / \delta_1$$

y  $c_1^2 + \beta_2^2 c_1^2 / \delta_1^2 = 1$

despejando  $c_1$

$$c_1^2 (1 + \beta_2^2 / \delta_1^2) = 1$$

$$c_1^2 (\delta_1^2 + \beta_2^2) / \delta_1^2 = 1$$

de donde

$$c_1 = (\delta_1^2 / (\delta_1^2 + \beta_2^2))^{1/2}$$

utilizando esto para encontrar

$$\delta_1 c_1 + \beta_2 s_1 = \gamma$$

sustituyendo  $s_1$  y  $c_1$

$$\gamma_{11} + \beta_{21} \bar{\delta}_1 = \gamma_{11}$$

$$(\bar{\gamma}_1 + \beta_{21} \gamma_{11}) = \gamma_{11}$$

desarrollando

$$((\bar{\delta}_1 + \beta_{21} \gamma_{11}) / (\bar{\delta}_1 + \beta_{21} \gamma_{11}))^{1/2} = c_{11}$$

introduciendo el primer factor dentro de la raíz

$$\gamma_{11} = \left( \frac{(\bar{\delta}_1 + \beta_{21} \gamma_{11})}{\bar{\delta}_1} \cdot \frac{\bar{\delta}_1}{(\bar{\delta}_1 + \beta_{21} \gamma_{11})} \right)^{1/2}$$

simplificando

$$\gamma_{11} = (\bar{\delta}_1 + \beta_{21} \gamma_{11})^{1/2}$$

sustituyendo en  $c_{11}$  y  $s_{11}$

$$c_{11} = \bar{\delta}_1 / \gamma_{11} \quad y \quad s_{11} = \beta_{21} \bar{\delta}_1 \cdot \bar{\delta}_1 / \gamma_{11} = \beta_{21} \bar{\delta}_1$$

obteniendo

$$\delta_{21} = \alpha_{21} + \beta_{21} c_{11}$$

y

$$\bar{\delta}_{21} = \alpha_{21} - \beta_{21} s_{11}$$

efectuemos la segunda iteración

$$t_3 = \begin{bmatrix} \alpha_{11} & \beta_{21} & 0 \\ \beta_{21} & \alpha_{21} & \beta_{31} \\ 0 & \beta_{31} & \alpha_{31} \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \beta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \alpha_1 & \beta_2 & 0 \\ \beta_2 & \alpha_2 & \beta_3 \\ 0 & \beta_3 & \alpha_3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \alpha_2 & \beta_3 \\ 0 & \beta_3 & \alpha_3 \end{bmatrix} = \begin{bmatrix} \gamma_1 & 0 & 0 \\ \delta_2 & \gamma_2 & 0 \\ \epsilon_3 & \delta_3 & \gamma_3 \end{bmatrix}$$

resultando primeramente

$$\begin{bmatrix} \gamma_1 & 0 & 0 \\ \delta_2 & \gamma_2 & \beta_3 \\ \beta_3 & \beta_3 & \alpha_3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \gamma_1 & 0 & 0 \\ \delta_2 & \gamma_2 & 0 \\ \delta_3 & \gamma_3 & \alpha_3 \end{bmatrix}$$

de donde

$$E = \begin{bmatrix} \beta_3 & \beta_3 \\ \beta_3 & \alpha_3 \end{bmatrix}$$

haciendo el mismo desarrollo de la iteración anterior se obtiene

$$\begin{aligned} \delta_2 &= \gamma_2 / \gamma_1 \\ \epsilon_3 &= \beta_3 / \gamma_2 \\ \delta_3 &= \alpha_3 \epsilon_3 + \beta_3 \delta_2 \\ \gamma_3 &= \alpha_3 \delta_3 - \beta_3 \epsilon_3 \end{aligned}$$

en general

$$\gamma_i = \bar{\gamma}_i + \beta_{i+1} \gamma_{i+1}$$

$$c_i = \bar{c}_i \gamma_i$$

$$s_i = \beta_{i+1} \gamma_i$$

$$d_{i+1} = \alpha_{i+1} s_i + \beta_{i+1} c_{i+1}$$

$$e_{i+1} = \beta_{i+1} s_{i+1}$$

$$\bar{\gamma}_{i+1} = \alpha_{i+1} c_i - \beta_{i+1} c_{i+1}$$

## II.4 SOLUCION ITERATIVA DEL PROBLEMA

Finalmente en esta seccion resolveremos el problema (I.1). En secciones anteriores se observo la necesidad de resolver el sistema  $\Gamma y = b$ , pero recordemos que  $\Gamma$  es una matriz ortogonal de columnas:

$$\Gamma = [q_1, q_2, \dots, q_n]$$

si tomamos  $q_1 = b/b_1$  entonces el problema queda simplificado como:

$$\Gamma y = e$$

Lo cual nos da una razon mas para tomar esta eleccion de  $q_1$ , el problema radica en la factorizacion, utilizando esta

$$L \Gamma y = e$$

sea  $C = \Gamma y$

$$L C = e$$

la cual puede ser resuelta en una forma sencilla

$$y = G$$

$$x = G G^T b \quad \dots (4.1)$$

Primero resolveremos el sistema

$$L C = e$$

y a continuación se efectuará

$$x = \begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix}$$

Resolvamos (4.1) efectuando unas iteraciones para entender mejor el algoritmo.

En la primera iteración

$$\begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 \\ \sigma_2 & \bar{\sigma}_2 \end{bmatrix} \begin{bmatrix} \xi_1 \\ \xi_2 \end{bmatrix} = \begin{bmatrix} \beta_1 \\ 0 \end{bmatrix}$$

de donde

$$\begin{aligned} \xi_1 &= \beta_1 / \delta_1 \\ \xi_2 &= -\delta_2 \xi_1 / \bar{\sigma}_2 \end{aligned}$$

en la segunda iteración

$$\begin{bmatrix} \delta_1 & 0 & 0 \\ \delta_2 & \sigma_2 & 0 \\ \delta_3 & \delta_3 & \bar{\sigma}_3 \end{bmatrix} \begin{bmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \end{bmatrix} = \begin{bmatrix} \beta_1 \\ 0 \\ 0 \end{bmatrix}$$

obteniendo

$$\begin{aligned} \xi_1 &= \beta_1 / \delta_1 \\ \xi_2 &= -\delta_2 \xi_1 / \bar{\sigma}_2 \end{aligned}$$

multiplicando y dividiendo esta expresión por

 $\bar{\sigma}_2$ 

$$\xi_2 = \frac{-\delta_2 \xi_1}{\bar{\sigma}_2} \cdot \frac{\bar{\sigma}_2}{\bar{\sigma}_2}$$

$$= \frac{\delta_{-2} \xi_1}{\xi_2} \cdot \frac{\bar{\xi}_2}{\delta_2}$$

$$= \xi_{2,2}^c$$

$$\xi_3 = (-\varepsilon \xi_3 - \delta_2 \xi_2) \cdot \bar{\xi}_3$$

en general

$$\xi_i = \bar{\xi}_i^c$$

$$\bar{\xi}_{i+1} = (-\varepsilon \xi_{i+1} - \delta_{i+1} \xi_i) \cdot \bar{\xi}_{i+1}$$

con esto hemos resuelto el sistema

$$L \bar{\xi} = \beta_2$$

Procedemos a encontrar una solución aproximada al problema

(1.1) efectuando el producto  $Q \cdot G$

$k=2$

$$Lq, q \begin{bmatrix} c & -s \\ 1 & 1 \\ s & c \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} -s q_1 + c q_2 & -s q_2 + c q_1 \\ c q_1 + s q_2 & c q_2 + s q_1 \\ s q_1 + c q_2 & s q_2 + c q_1 \\ c q_1 + s q_2 & c q_2 + s q_1 \end{bmatrix}$$

llamando  $\bar{W} = q$

entonces  $W = c \bar{W} + s q$

$$\bar{W} = -s W + c q$$

$$Q \cdot G \begin{bmatrix} \\ \\ \\ \end{bmatrix}$$

$$[W, \bar{W}] \begin{bmatrix} \xi_1 \\ \xi_2 \end{bmatrix}$$



$$x_2^0 = W_1 \left\{ \begin{array}{l} +W_2 \\ \bar{W}_2 \end{array} \right\}$$

k=3

$$[Lq, q, q] \begin{bmatrix} c & -s & 0 \\ s & c & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} c \\ s \\ 0 \end{bmatrix}$$

$$[W, W, q] \begin{bmatrix} 1 & 0 & 0 \\ 0 & c & s \\ 0 & s & c \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$= [W_1, c \bar{W}_2 + s q, -s \bar{W}_2 + c q]$$

por lo tanto

$$W_2 = c \bar{W}_2 + s q$$

$$\bar{W}_2 = -s \bar{W}_2 + c q$$

$$G_3 = [W_1, W_2, W_3]$$

$$x_2 = W_1 \left\{ \begin{array}{l} W_2 \\ \bar{W}_2 \end{array} \right\}$$

$$x_3 = W_1 \left\{ \begin{array}{l} W_2 \\ \bar{W}_2 \\ W_3 \end{array} \right\}$$

quedando

$$x_3^0 = x_2 \left\{ \begin{array}{l} \bar{W}_2 \\ W_3 \end{array} \right\}$$

en general

$$W_i = c_i \bar{W}_i + s_i q_{i+1}$$

$$W_{i+1} = -s_{i+1} \bar{W}_{i+1} + c_{i+1} q_{i+1}$$

$$x_i = x_{i-1} + \left\{ \begin{array}{l} W_i \\ i \end{array} \right.$$

$$x_{i+1} = x_i + \left\{ \begin{array}{l} \bar{W}_{i+1} \\ i+1 \end{array} \right.$$

El algoritmo termina cuando  $q_{i+1} = 0$  lo cual nos implica

$\bar{W}_{i+1} = 0$  y  $x_{i+1} = x_i$  utilizando esto como criterio de alto.

Con esto queda resuelto el problema de resolver (I.1). En el espíritu de utilizar la solución anterior para calcular una nueva solución y de esta forma requerir menor memoria y cómputo, puesto que, utiliza únicamente algunos vectores auxiliares, y los datos de dos soluciones anteriores.

En la siguiente sección únicamente escribiremos los algoritmos que reflejan los resultados obtenidos en el capítulo.

## II.5 ALGORITMO MATEMATICO MINLG

ESTE ALGORITMO RESUELVE EL SISTEMA  $Ax=b$  CON A MATRIZ SIMETRICA, GRANDE Y HUECA.

UTILIZANDO EL METODO DE ITERACIONES MINIMIZADAS, VIA PSEUDO-KRILOV Y LA FACTORIZACION LG DE LA MATRIZ T.

ENTRADA:

A; matriz nxn

n; orden de A

b; lado derecho del sistema

INICIO

$$q = 0$$

$$\beta = \|b\|_2$$

$$q = b / \beta$$

$$\alpha = q / Aq$$

$$c = 1.0$$

$$s = 0.0$$

$$\bar{r} = \alpha$$

$$\bar{w} = q$$

$$\bar{z} = \beta / \bar{r}$$

$$\bar{z} = 0.0$$

## II ITERACIONES

1) CALCULO DE LOS VECTORES Y ESCALARES DE LA TRIDIAGONALIZACION DE LANCZOS. (ESTE PROCEDIMIENTO LO EFECTUA EN FORMA ITERATIVA UTILIZANDO UNICAMENTE DOS VECTORES ANTERIORES).

$$AQ = Q T$$

UTILIZANDO EL CRITERIO DE ALTO CUANDO  $\beta_{i+1} = 0$

$$i=1$$

$$\text{mientras } \beta_{i+1} \neq 0$$

$$R_{i+1} = Aq_i - \alpha_{i,j} q_i - \beta_{i,i-1} q_{i-1}$$

$$\beta_{i+1} = \|R_{i+1}\|_2$$

$$q_{i+1} = R_{i+1} / \beta_{i+1}$$

$$\alpha_{i+1} = q_{i+1}^T A q_i$$

2) CALCULA ESCALARES DE LA FACTORIZACION ( = L G )

$$\gamma_i = (\gamma_i^2 + \beta_{i+1}^2)^{1/2}$$

$$c_i = \gamma_i / \beta_i$$

$$s_i = \beta_i / \gamma_i$$

$$d_{i+1} = \alpha_{i+1} s_i - \beta_{i,i+1} c_i c_i$$

$$e_{i+1} = \beta_{i+1} s_i$$

$$\bar{\gamma}_{i+1} = d_{i+1} c_i - \beta_{i+1} s_i c_i$$

3) CALCULA LA SOLUCION DEL SISTEMA  $Lx = b$

$$\{ \begin{matrix} x_i \\ \vdots \\ x_i \end{matrix} = \{ \begin{matrix} c \\ \vdots \\ c \end{matrix}$$

$$\{ \begin{matrix} x_{i+1} \\ \vdots \\ x_{i+1} \end{matrix} = -E \{ \begin{matrix} x_{i-1} \\ \vdots \\ x_{i-1} \end{matrix} - d \{ \begin{matrix} x_{i+1} \\ \vdots \\ x_{i+1} \end{matrix}$$

4) CALCULA LA SOLUCION APROXIMADA DEL SISTEMA ORIGINAL COMO:

$$x = \{ \begin{matrix} 0 \\ \vdots \\ 0 \end{matrix}$$

$$W = c \{ \begin{matrix} \bar{w}_i \\ \vdots \\ \bar{w}_i \end{matrix} + s \{ \begin{matrix} q_i \\ \vdots \\ q_i \end{matrix}$$

$$x_i = x_{i-1} + \{ \begin{matrix} W \\ \vdots \\ W \end{matrix}$$

$$\bar{w}_{i+1} = -s \{ \begin{matrix} \bar{w}_i \\ \vdots \\ \bar{w}_i \end{matrix} + c \{ \begin{matrix} q_i \\ \vdots \\ q_i \end{matrix}$$

$$x_{i+1} = x_{i-1} + \{ \begin{matrix} \bar{w}_{i+1} \\ \vdots \\ \bar{w}_{i+1} \end{matrix}$$

EL ALGORITMO TERMINA CUANDO  $\epsilon = 0.0$  PUESTO QUE EL NUEVO VECTOR DE LANZOS  $q_{i+1} = 0$  Y LOS ESCALARES  $c = 1.0$ ,  $s = 0.0$

RESULTANDO  $x_{i+1} = x_{i+1}$  LA SOLUCION EXACTA DEL PROBLEMA ORIGINAL (I.I).

## II.6 ALGORITMO INFORMAL MIMLG

ESTE ALGORITMO REFLEJA EL ALGORITMO ANTERIOR CON EL OBJETO DE APROVECHAR MEMORIA.

ALMACENA:

VECTORES

$q_{i-1}$	en	el vector	Q0
$q_i$	"	"	Q1
$aq_i$	"	"	AQ
$x_i$	"	"	X
$w_{i+1}$	"	"	W
$k_{i+1}$	"	"	K

ESCALARES

$\alpha_{i+1}$	en	el escalar	ALFA
$\beta_{i+1}$	"	"	BETA
$\gamma_i$	"	"	EPSILON
$c_{i-1}$	"	"	C
$s_{i-1}$	"	"	S
$\delta_{i+1}$	"	"	DELTA
$\epsilon_{i+1}$	"	"	EPSILON
$\bar{\gamma}_{i+1}$	"	"	GAMMA
$\zeta_i$	"	"	SIGMA

{  
i+1            "    "            SIGMA

1. INICIO

LAS INSTRUCCIONES SIGUIENTES REFLEJAN LAS DEL NUMERO 1-10 DEL ALGORITMO ANTERIOR.

Q0 <--- 0

BETA <--- B

Q1 <--- B/BETA<sup>2</sup>

AQ <--- AQ1

C <--- 1.0

S <--- 0.0

GAMMA <--- ALFA

W <--- Q1

SIGMA <--- BETA/GAMMA

SIGNAL <--- 0.0

II. ITERACIONES

1.) CALCULOS DE TRIANGONALIZACION DE LANZOS (REFLEJANDO LAS INSTRUCCIONES 11-15)

AQ <--- AQ-ALFA Q1-BETA\*Q0

BETA <--- AQ

SI BETA=0.0 VETE A 32

Q0 <--- Q1

Q1 <--- AQ/BETA

AQ <--- Q1\*AQ

2) CALCULO DE LA FACTORIZACION L G = T (REFLEJANDO INSTRUCCIONES 16-21)

EPSILON <--- (GAMMA + BETA)<sup>2</sup> / 2<sup>1/2</sup>

CI <--- GAMMA/EPSILON

SI <--- BETA/EPSILON

DELTA <--- SI ALFA+ CI\*C\*BETA

EPSILON <--- S\*BETA

GAMMA <--- ALFA\*CI-BETA\*SI\*C

3) SOLUCION DEL SISTEMA L O b (REFLEJANDO INSTRUCCIONES 22-23

SIGMAI <--- CI\*SIGMA

SIGMA <--- -EPSILON\*SIGMA-DELTA\*SIGMAI

4) CALCULA SOLUCION DEL SISTEMA ORIGINAL DE ACUERDO A LAS

INSTRUCCIONES 24-28 COMO

$x=0$  G

$i=1$  i

X <--- X+SIGMA\*(-SI\*B+CI\*RI)

W <--- SI\*B-CI\*RI

S <--- SI

C >--- CI

VETE A 12

X=X+SIGMA\*B

ALTO.



### III IMPLEMENTACION DEL ALGORITMO MIMLG

En el capitulo anterior presentamos un algoritmo (MIMLG) para resolver el sistema  $Ax=b$

donde A matriz simetrica , grande y hueca

En el presente y ultimo capitulo, llevaremos a la practica este algoritmo, cumpliendo de esta forma con los objetivos planteados en el trabajo:

1. Presentar un algoritmo "EFICAZ" para resolver nuestro problema (I.1).

En teoria el algoritmo MIMLG converge en a lo mas n- iteraciones, ya que, la sucesion de Krilov  $k(b, A, j)$ , genera vectores linealmente independientes a saber  $b, Ab, \dots, A^{j-1}b$  con  $j \leq n$ , en el siguiente paso los vectores son finalmente dependientes, con lo cual, se puede obtener una combinacion lineal de b en terminos de la sucesion de pseudo krilov es decir:

$$b = \alpha_1 Ab + \alpha_2 A^2 b + \dots + \alpha_j A^j b \quad [3]$$

Por esta causa en el algoritmo se obtiene  $q_j = 0$  el cual es un criterio de haber obtenido la solucion: La sucesion de soluciones

iteradas  $x_1^c, x_2^c, x_3^c, \dots, x_j^c$  cuando

$$x_j^c = x_j$$

Por lo tanto trabajando en aritmetica real en el paso j encuentra la solucion exacta de  $Ax=b$ .

Mas sin embargo, esto no ocurre en la practica, puesto que, nos enfrentamos a problemas de dimension grande lo cual nos hace utilizar una maquina para resolver (I.1), haciendo esto imposible

trabajar con aritmetica real. esto nos da una solucion aproximada. debido a efectos de error por redondeo.

De lo anterior se desprenden tres criterios de alto:

- 1.-  $\|q\| \leq \text{tolerancia} \rightarrow \text{alto}$
- 2.-  $\max_i |x_i - x_{i-1}| / |x_i| \leq \text{tolerancia} \rightarrow \text{alto}$
- 3.-  $j = n \rightarrow \text{alto}$ .

Otra de las cualidades que debemos pedir al algoritmo es la de hacerlo competitivo con otro metodo conocido: El metodo de Eliminacion Gaussiana, es uno de los mas utilizados hoy en dia. para, resolver  $Ax=b$ . dicho metodo puede destruir la estructura de la matriz. lo cual, nos llevaria a tomar como  $n^3/3$  el numero de flops necesarios en el problema de resolver (1.1); El algoritmo MINLQ requiere a lo mas  $n^2 m$  flops, donde  $m$  es el numero maximo de elementos distintos de cero en nuestra matriz.

Observemos lo siguiente:

$n^3/3 - n^2 m = 100\ 000$   
para  $n=100$ , da  $m \leq 23$ .

Es decir. tomando estas dos dimensiones existe un ahorro de 100 000 flops si ocurriese la destruccion de la estructura.

Por esta razon tomaremos matrices grandes y huecas con  $n \geq 100$  y menos de un 20% de elementos distintos de cero.

Nosotros consideraremos un algoritmo eficaz si cumple con todo lo anteriormente expuesto.

## II. PROGRAMACION

Utilizaremos una computadora BURROUGHS-7800 y FORTRAN IV como lenguaje de programacion.

Presentaremos el algoritmo en forma de subrutina tomando las ideas presentadas en el algoritmo en lenguaje informal, a su vez esta utiliza dos subrutinas del tipo function que son parte de la subrutina principal:

1.-PQIAQ (N,QI,AQ)

Esta subrutina efectua el producto escalar dados vectores a saber QI por AQ.

2.-ERREL (N,X,Y)

Esta subrutina calcula el maximo error relativo por componentes de dos soluciones subsecuentes; El cual sera utilizado como criterio de alto, para no efectuar el trabajo que no mejore en forma sustancial una solucion aproximada.

Deseamos ademas que la subrutina principal no dependa del tipo de estructura de la matriz A. es por esta causa que requiere de una subrutina auxiliar que multiplique la matriz A por un vector; Dicha subrutina debe ser proporcionada por el usuario, conservando el mismo nombre y parametros: `MATSON(Q,ABQ)` donde:

n: es la dimension del sistema.

Q: vector por el cual se multiplica la matriz A.

ABQ: resultado de la multiplicacion.

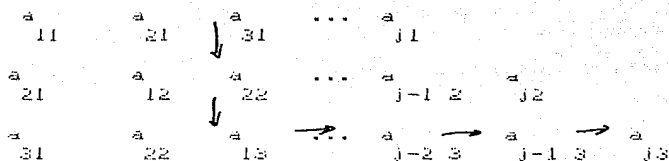
La matriz A es introducida en esta subrutina mediante la instruccion COMMON o algo equivalente.

a continuacion presentamos la subrutina principal y sus dos funciones auxiliares. (ver apendice)



Presentamos una subrutina que multiplica una matriz en Banda por un vector, ya que la matriz A debe ser simétrica la forma de multiplicarla es tomar la matriz en forma escalonada para ahorrar memoria.

es decir:



en el caso del tercer renglón como lo muestran las flechas.

A continuación presentaremos el programa principal y la subrutina MAQS para el ejemplo particular; Cabe mencionar que la subrutina MAQS funciona para cualquier matriz simétrica en Banda. (ver apéndice)

Se efectuaron pruebas para diferentes valores de n y m, con un lado derecho de tal forma que la solución x(i) sea igual a 1 obteniendo los siguientes resultados.

N=100,	M=10,	TOL=1.0E-07	SOL. EXACTA EN 34 ITERACIONES
N=150,	M=16,	"	" 46 "
N=200,	M=21,	"	" 66 "
N=250,	M=26,	"	" 81 "
N=300,	M=31,	"	" 102 "



TOL=1.0E-07

n=nx\*ny

nx=15,	ny=10,	n=100	sol. exacta en 16 iteraciones		
nx=15,	ny=15,	n=225	"	29	"
nx=20,	ny=20,	n=400	"	39	"
nx=25,	ny=25,	n=625	"	51	"
nx=30,	ny=30,	n=900	"	60	"
nx=20,	ny=50,	n=1000	"	80	"
nx=50,	ny=20,	n=1000	"	80	"

Finalmente tomamos una matriz grande, hueca, simetrica en general es decir:

```

      *   *       *   *   *
      *       *       *       *
      .....
A=    .....
      .....
      *   *       *   *   *

```

Donde los elementos marcados son distintos de cero.

Como en los ejemplos anteriores presentamos el programa principal y subrutina MA05 para este ejemplo, aclarando que la subrutina MA05 funciona en general. (ver apendice)

Probando con un sistema donde sea tal que  $a_{ij} = 0$  o 1 al azar de tal forma que no existan mas del 20% de elementos distintos de cero, con solucion conocida  $x(i)=1$ ; no se obtuvo solucion exacta en 100 iteraciones (100 dimension de A), pero su valor fluctua entre 0.97 y 1.02.



#### IV. CONCLUSIONES

Este metodo es recomendable de usarse en el caso de tener una matriz A con posibilidades de que al efectuar Eliminacion Gaussiana pierda la estructura, lo que, llevaria a obtener matrices L y U triangulares llenas con un gran requerimiento de memoria.

Si bien tiene la desventaja de estar sujeto a un unico lado derecho, es decir no puede utilizarse para varios lados derechos al mismo tiempo.

Esta desventaja es todavia un problema en estudio en base a [6].

Otra desventaja de nuestro metodo es el ser unicamente para matrices simetricas.

Esto ultimo puede ser resuelto tomando:

$$\begin{bmatrix} I & A \\ A^t & 0 \end{bmatrix} \begin{bmatrix} r \\ y \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}$$

El utilizar la tridiagonalizacion en forma directa sin utilizar alguna estrategia de este tipo tambien es un problema de estudio como se menciona en [5].

# APENDICE

SUBROUTINE MIMLG(N,B,QO,Q1,AQ,X,Y,W,TOL,J)

\*\*\*\*\*

ESTA SUBROUTINA ES LA IMPLEMENTACION DEL ALGORITMO MIMLG  
EL CUAL RESUELVE EL SISTEMA  $AX=B$  :

A: MATRIZ SIMETRICA , GRANDE , HUECA.

PARAMETROS:

ENTRADA:

N: TAMANO DEL SISTEMA.

B: LADO DERECHO DEL SISTEMA.

TOL: CRITERIO DE CONVERGENCIA PARA LA SUCECION  
DE VECTORES QUE TIENDEN A LA SOLUCION, ES-  
TO ES: EL MAXIMO ERROR RELATIVO POR COMPO-  
NENTES DE LOS DOS ULTIMOS VECTORES EN LA  
SUCECION SEA MENOR O IGUAL QUE TOL.

SALIDA:

B: VECTOR RESIDUAL.

Y: VECTOR SOLUCION.

SUBPROGRAMAS:

MARS(N,Q,AQ); ESTA SUBROUTINA MULTIPLICA LA  
MATRIZ A POR EL VECTOR Q, ASIG-  
NANDO EL RESULTADO EN AQ.

\*\*\*\*\*

ESTA SUBROUTINA ES PROPORCIONADA  
POR EL USUARIO, DEPENDIENDO DEL  
TIPO DE ESTRUCTURA DEL SISTEMA.

\*\*\*\*\*

ERREL(N,X,Y); FUNCION RE QUE CALCULA EL MAX-  
IMO ERROR RELATIVO POR COMPONEN-  
TES DE LAS DOS ULTIMAS SOLUCIO -  
NES ITERATIVAS.

PQ1AQ(N,Q1,AQ); FUNCION QUE CALCULA EL PRODUCTO  
ESCALAR DE LOS VECTORES Q1 POR  
AQ.

\*\*\*\*\*

DIMENSION B(N),QO(N),Q1(N),AQ(N),X(N),Y(N),W(N)  
INTEGER N,J  
REAL TOL,ALFA,BETA,C,C1,S,S1,GAMMA,EPSI,SIGMA,SIGMA1,Z  
REAL SIGMAO  
DOUBLE PRECISION RAIZ

\*\*\*\*\*

```

6100 C*
6200 C*      INICIO
6300 C*
6400 C*      ****
6500 C*
6600      Z=PQ1AQ(N,B,B)
6700      RAIZ=Z
6800      RAIZ=DSQRT(RAIZ)
6900      BETA=RAIZ
7000      DO 10 I=1,N
7100      Q1(I)=B(I)/BETA
7200      W(I)=Q1(I)
7300      10  QO(I)=0.0
7400      CALL MARS(N,Q1,AQ)
7500      ALFA=PQ1AQ(N,Q1,AQ)
7600      C=1.0
7700      S=0.0
7800      GAMMA=ALFA
7900      IF (GAMMA.EQ.0.0)GO TO 32
8000      SIGMA=BETA/GAMMA
8100      SIGMA1=0.0
8200 C*
8300 C*      ****
8400 C*
8500 C*      ITERACIONES
8600 C*
8700 C*      ****
8800 C*
8900 C*      ****
9000 C*
9100 C*      1) CALCULA VECTORES DE LANCZOS.
9200 C*
9300 C*      ****
9400 C*
9500      J=1
9600      12  DO 20 I=1,N
9700      20  AQ(I)=AQ(I)-ALFA*Q1(I)-BETA*QO(I)
9800      Z=PQ1AQ(N,AQ,AQ)
9900      RAIZ=Z
10000     RAIZ=DSQRT(RAIZ)
10100     BETA=RAIZ
10200     IF (BETA.LE.TOL)GO TO 100
10300     DO 30 I=1,N
10400     QO(I)=Q1(I)
10500     30  Q1(I)=AQ(I)/BETA
10600     CALL MARS(N,Q1,AQ)
10700     ALFA=PQ1AQ(N,Q1,AQ)
10800 C*
10900 C*      ****
11000 C*
11100 C*      2) REALIZA LA FACTORIZACION LG DE T.
11200 C*
11300 C*      ****
11400 C*
11500     Z=GAMMA*GAMMA+BETA*BETA
11600     RAIZ=Z
11700     RAIZ=DSQRT(RAIZ)
11800     EPSI=RAIZ
11900     IF (EPSI.EQ.0.0)GO TO 100
12000     C1=GAMMA/EPSI

```

```

12100      S1=BETA/EPFI
12200      DELTA=ALFA*S1+BETA*C1*C
12300      EPFI=BETA*S
12400      GAMMA=C1*ALFA-S1*C*BETA
12500 C*
12600 C*      *****
12700 C*
12800 C*      3) RESUELVE EL SISTEMA LZ=BETA1*E1.
12900 C*
13000 C*      *****
13100 C*
13200      SIGMA0=SIGMA1
13300      SIGMA1=SIGMA*C1
13400      SIGMA=(-SIGMA0*EPFI-DELTA*SIGMA1)/GAMMA
13500 C*
13600 C*      *****
13700 C*
13800 C*      4) ENCUENTRA LA SOLUCION ITERATIVA DEL PASO J.
13900 C*
14000 C*      *****
14100 C*
14200      DO 40 I=1,N
14300      X(I)=X(I)+SIGMA1*(C1*W(I)+S1*Q1(I))
14400      W(I)=-S1*W(I)+C1*Q1(I)
14500      S=S1
14600      C=C1
14700      ERROR=ERREL(N,X,Y)
14800      DO 50 I=1,N
14900      Y(I)=X(I)+SIGMA*W(I)
15000      J=J+1
15100      IF(ERROR.LE.TOL)GO TO 100
15200      IF (J.GT.N)GO TO 100
15300      GO TO 12
15400      DO 60 I=1,N
15500      Y(I)=0.0
15600      CALL MARS(N,Y,AQ)
15700      DO 70 I=1,N
15800      B(I)=B(I)-AQ(I)
15900      B(I)=-B(I)
16000      RETURN
16100      END
16200 C*
16300 C*      *****
16400 C*
16500 C*
16600      REAL FUNCTION FQ1AQ(N,Q,AQ)
16700      DIMENSION Q(N),AQ(N)
16800      INTEGER N
16900      DOUBLE PRECISION S1,S2,SUMA
17000 C*
17100 C*      *****
17200 C*
17300 C*      ESTE SUBROUTINA REALIZA EL PRODUCTO ESCALAR DE DOS VECTORES
17400 C*
17500 C*
17600 C*      ENTRADA:
17700 C*
17800 C*      N# DIMENSION DE LOS VECTORES.
17900 C*
18000 C*      Q# 1- VECTOR A MULTIPLICAR

```

18100 C\*  
18200 C\*  
18300 C\*  
18400 C\*  
18500 C\*  
18600 C\*  
18700 C\*  
18800 C\*  
18900 C\*  
19000 C\*  
19100  
19200  
19300  
19400  
19500 10  
19600  
19700  
19800  
19900 C\*  
20000 C\*  
20100 C\*  
20200 C\*  
20300  
20400  
20500  
20600 C\*  
20700 C\*  
20800 C\*  
20900 C\*  
21000 C\*  
21100 C\*  
21200 C\*  
21300 C\*  
21400 C\*  
21500 C\*  
21600 C\*  
21700 C\*  
21800 C\*  
21900 C\*  
22000 C\*  
22100 C\*  
22200 C\*  
22300 C\*  
22400 C\*  
22500 C\*  
22600 C\*  
22700  
22800  
22900  
23000 C\*  
23100 C\*  
23200 C\*  
23300  
23400  
23500 10  
23600  
23700  
23800 C\*  
#

AR; 2- VECTOR A MULTIPLICAR

SALIDA:

PQ1AQ; RESULTADO DEL PRODUCTO.

\*\*\*\*\*

```
SUMA=0.0  
DO 10 I=1,N  
S1=R(I)  
S2=AQ(I)  
SUMA=SUMA+S1*S2  
PQ1AQ=SUMA  
RETURN  
END
```

REAL FUNCTION ERREL(N,X,Y)  
DIMENSION X(N),Y(N)  
INTEGER N

\*\*\*\*\*

ESTA SUBROUTINA ENCUENTRA EL MAXIMO ERROR RELATIVO POR  
COMPONENTES DE DOS SOLUCIONES SUBSECUENTES

ENTRADA:

N; DIMENSION DEL SISTEMA  
X; VECTOR SOLUCION ANTERIOR  
Y; VECTOR SOLUCION ACTUAL

SALIDA:

ERREL; ERROR RELATIVO.

\*\*\*\*\*

```
ERREL=0.0  
DO 10 I=1,N  
DIF=ABS(Y(I)-X(I))  
SI X(I)=0.0 ENTONCES TOMA EL ERROR ABSOLUTO  
IF(X(I).EQ.0.0)GO TO 10  
DIF=DIF/ABS(X(I))  
IF(DIF.GT.ERREL)ERREL=DIF  
RETURN  
END
```

```

5100 C*
5200 C*          LLAMA PROGRAMA PRINCIPAL
5300 C*
5400 C*          ****
5500 C*
5600 C*          7  CALL MIHLG(N,B,Q0,Q1,AQ,X,Y,W,TOL,J)
5700 C*
5800 C*          ****
5900 C*
6000 C*          ESCRIBE RESULTADOS
6100 C*
6200 C*          ****
6300 C*
6400 C*          WRITE (6,3)
6500 C*          3  FORMAT(3X,"SOLUCION DEL SISTEMA Y RESIDUAL")
6600 C*          DO 30 I=1,N
6700 C*          30  WRITE(6,4) Y(I),B(I)
6800 C*          4  FORMAT(/,8X,2(F8.4,5X))
6900 C*          WRITE(6,5)
7000 C*          5  FORMAT(3X,"NO. DE ITERACIONES")
7100 C*          WRITE(6,*) J
7200 C*          CALL EXIT
7300 C*          END
7400 C*
7500 C*          ****
7600 C*
7700 C*          SUBROUTINE MAGS(N,Q,AQ)
7800 C*          COMMON N
7900 C*          INTEGER N,K,NM,N1
8000 C*          DIMENSION Q(N),AQ(N)
8100 C*          DOUBLE PRECISION S1,S2,SUMA
8200 C*
8300 C*          ****
8400 C*
8500 C*          ESTA SUBROUTINA MULTIPLICA LA MATRIZ A (BANDA DE LONGITUD
8600 C*          2M-1) POR UN VECTOR Q.
8700 C*
8800 C*          ENTRADA:  A:  MATRIZ NXN DE BANDA DE ANCHO 2M-1
8900 C*
9000 C*                   B:  VECTOR NX1
9100 C*
9200 C*                   M:  DIMENSION DE LA BANDA
9300 C*
9400 C*                   N:  NO. DE RENGLONES DE LA MATRIZ A
9500 C*
9600 C*          SALIDA:  AQ:  VECTOR PRODUCTO DE A POR Q
9700 C*
9800 C*          SUBROUTINA AUX:  SUBROUTINA QUE MULTIPLICA LOS RENGLONES
9900 C*                   DE LA MATRIZ A POR EL VECTOR Q.
10000 C*
#

```

```

5100 C*
5200 C*
5300 C*
5400 C*
5500 C*
5600 C*
7 CALL MIMLG(N,B,QO,QI,AQ,X,Y,W,TOL,J)
5700 C*
5800 C*
5900 C*
6000 C*
6100 C*
6200 C*
6300 C*
6400 C*
6500 C*
6600 C*
6700 C*
6800 C*
6900 C*
7000 C*
7100 C*
7200 C*
7300 C*
7400 C*
7500 C*
7600 C*
7700 C*
7800 C*
7900 C*
8000 C*
8100 C*
8200 C*
8300 C*
8400 C*
8500 C*
8600 C*
8700 C*
8800 C*
8900 C*
9000 C*
9100 C*
9200 C*
9300 C*
9400 C*
9500 C*
9600 C*
9700 C*
9800 C*
9900 C*
10000 C*
#

LLAMA PROGRAMA PRINCIPAL

*****

WRITE (6,3)
FORMAT(3X,"SOLUCION DEL SISTEMA Y RESIDUAL")
DO 30 I=1,N
WRITE(6,4) Y(I),B(I)
FORMAT(/,8X,2(F8.4,5X))
WRITE(6,5)
FORMAT(3X,"NO. DE ITERACIONES")
WRITE(6,* /) J
CALL EXIT
END

*****

SUBROUTINE MARS(N,Q,AQ)
COMMON M
INTEGER N,K,NM,N1
DIMENSION Q(N),AQ(N)
DOUBLE PRECISION S1,S2,SUMA

*****

ESTA SUBROUTINA MULTIPLICA LA MATRIZ A (BANDA DE LONGITUD
2M-1) POR UN VECTOR Q.

ENTRADA: A# MATRIZ NXN DE BANDA DE ANCHO 2M-1
B# VECTOR NX1
N# DIMENSION DE LA BANDA
N# NO. DE RENGLONES DE LA MATRIZ A

SALIDA: AQ# VECTOR PRODUCTO DE A POR Q

SUBROUTINA AUX# SUBROUTINA QUE MULTIPLICA LOS RENGLONES
DE LA MATRIZ A POR EL VECTOR Q.

```

```

10100 C*      *****
10200 C*
10300      N1=M-1
10400      CALL AUX(N,Q,AQ,1,N1)
10500      N1=N-M+1
10600      CALL AUX(N,Q,AQ,M,N1)
10700      N1=N1+1
10800      CALL AUX(N,Q,AQ,N1,N)
10900      RETURN
11000      END
11100 C*
11200 C*      *****
11300 C*
11400      SUBROUTINE AUX(N,Q,AQ,K,NM)
11500 C*      *****
11600 C*
11700 C*      ESTA SUBROUTINA EFECTUA EL PRODUCTO DEL RENGLON K HASTA
11800 C*      EL RENGLON NM, DE LA MATRIZ A POR EL VECTOR AQ.
11900 C*
12000 C*      *****
12100 C*      COMMON A(300,31)
12200      COMMON M
12300      INTEGER N,K,NM,I,NJ,NK,M1
12400      DIMENSION Q(N),AQ(N)
12500      DOUBLE PRECISION S1,S2,SUMA
12600      DO 10 I=K,NM
12700      NJ=I
12800      SUMA=0.0
12900      DO 20 J=1,M
13000      S1=A(I,J)
13100      S2=Q(NJ)
13200      SUMA=SUMA+S1*S2
13300      NJ=NJ+1
13400      IF(NJ.GT.N)GO TO 40
13500      CONTINUE
13600      20  NJ=I-1
13700      40  DO 30 J=2,M
13800      IF(NJ.LT.1)GO TO 50
13900      S1=A(NJ,J)
14000      S2=Q(NJ)
14100      SUMA=SUMA+S1*S2
14200      NJ=NJ-1
14300      30  CONTINUE
14400      50  AQ(I)=SUMA
14500      10  CONTINUE
14600      RETURN
14700      END
14800

```



```

100 C*
200 C*
300 C*
400 C*
500 C*
600 C*
700 C*
800 C*
900 C*
1000 C*
1100 C*
1200 C*
1300 C*
1400 C*
1500 C*
1600
1700
1800
1900
2000
2100
2200 C*
2300 C*
2400 C*
2500 C*
2600 C*
2700 C*
2800 C*
2900
3000
3100
3200
3300
3400
3500
3600
3700
3800
3900 20
4000 C*
4100 C*
4200 C*
4300 C*
4400 C*
4500 C*
4600 C*
4700
4800 C*
4900 C*
5000 C*
5100 C*
5200 C*
5300 C*
5400 C*
5500
5600
5700
5800 30

```

\*\*\*\*\*  
ESTE PROGRAMA RESUELVE EL SISTEMA AX=B POR EL METODO  
NIMLG DONDE A ES UNA MATRIZ TRIDIAGONAL POR BLOQUES  
TAL COMO SE PRESENTO EN EL EJEMPLO 2 .  
\*\*\*\*\*  
\*\*\*\*\*  
DECLARACIONES  
\*\*\*\*\*  
DIMENSION B(1000),Y(1000),X(1000),Q0(1000),Q1(1000)  
DIMENSION W(1000),AQ(1000)  
COMMON NX,NY  
INTEGER N,I,J  
READ(5,/) NX,NY,TOL  
N=NX\*NY  
\*\*\*\*\*  
GENERA EL SISTEMA Y ESCRIBE DATOS  
\*\*\*\*\*  
DO 10 I=1,N  
NR=MOD(I,NX)  
IF(I.LE.NX.OR.I.GT.(N-NX))B(I)=-1  
IF(NR.EQ.1.OR.NR.EQ.0)B(I)=-1  
B(I)=-2  
B(N)=-2  
B(NX)=-2  
B(N-NX+1)=-2  
WRITE(6,/) N,NX,NY,TOL  
DO 20 I=1,N  
WRITE(6,/) B(I)  
\*\*\*\*\*  
LLAMA SUBROUTINA PRINCIPAL  
\*\*\*\*\*  
CALL NIMLG(N,B,Q0,Q1,AQ,X,Y,W,TOL,J)  
\*\*\*\*\*  
ESCRIBE RESULTADOS  
\*\*\*\*\*  
WRITE(6,3)  
FORMAT(//,5X,"SOLUCION Y RESIDUAL")  
DO 30 I=1,N  
WRITE(6,1) Y(I),B(I)

```

5900 1 FORMAT(8X,2(F10.7,5X))
6000 WRITE(6,2) J
6100 2 FORMAT(//,3X,"NO. DE ITERACIONES=",I4)
6200 CALL EXIT
6300 END
6400 C*
6500 C*
6600 C* *****
6700 C*
6800 C*
6900 C* SUBROUTINE MAQS(N,Q,AQ)
7000 C*
7100 C* *****
7200 C*
7300 C* ESTA SUBROUTINA CALCULA EL PRODUCTO DE LA MATRIZ A
7400 C* TRIANGULAR POR BLOQUES COMO EN EL EJEMPLO 2
7500 C* EN FORMA DIRECTA.
7600 C*
7700 C* ENTRADA:
7800 C* Q: VECTOR POR EL CUAL SE MULTIPLICA LA MATRIZ
7900 C* A.
8000 C*
8100 C* N: DIMENSION DEL SISTEMA
8200 C*
8300 C* SALIDA:
8400 C*
8500 C* AQ: VECTOR RESULTADO DEL PRODUCTO
8600 C*
8700 C* COMMON:
8800 C*
8900 C* NX,NY: NUMERO DE PUNTOS EN LA MALLA DE DISCRETI-
9000 C* ZACION.
9100 C*
9200 C* *****
9300 C*
9400 C* DIMENSION Q(N),AQ(N)
9500 C* COMMON NX,NY
9600 C* DOUBLE PRECISION S1,S2,S3,S4,S5
9700 C* DO 10 I=1,N
9800 C* NR=MOD(I,NX)
9900 C* S1=0.0
10000 C* S2=0.0
10100 C* S3=0.0
10200 C* S4=0.0
10300 C* S5=0.0
10400 C* IF(I.LE.NX)GO TO 15
10500 C* S1=Q(I-NX)
10600 15 IF(NR.EQ.1)GO TO 20
10700 C* S2=Q(I-1)
10800 20 S3=Q(I)
10900 C* IF(NR.EQ.0)GO TO 25
11000 C* S4=Q(I+1)
11100 25 IF(I.GT.(N-NX))GO TO 30
11200 C* S5=Q(I+NX)
11300 30 S1=S1+S2-4*S3+S4+S5
11400 10 AQ(I)=S1
11500 C* RETURN
11600 C* END
#

```

```

100 C*
200 C*
300 C*
400 C*
500 C*
600 C*
700 C*
800 C*
900 C*
1000 C*
1100 C*
1200 C*
1300 C*
1400 C*
1500 C*
1600 C*
1700 C*
1800 C*
1900 C*
2000 C*
2100 C*
2200
2300
2400
2500
2600
2700 C*
2800 C*
2900 C*
3000 C*
3100 C*
3200 C*
3300 C*
3400
3500
3600
3700
3800
3900
4000
4100
4200
4300
4400
4500
4600
4700
4800 C*
4900 C*
5000 C*
5100 C*
5200 C*
5300 C*
5400 C*
5500
5600
5700
5800
#

```

ESTE PROGRAMA RESUELVE EL SISTEMA  $AX=B$  POR EL METODO DE MINLG.

DONDE A ES UNA MATRIZ GRANDE , HUECA Y SIMETRICA EN GENERAL .

PROGRAMA PRINCIPAL

DECLARACIONES

```

DIMENSION B(100),Y(100),X(100),Q0(100),Q1(100)
DIMENSION W(100),AQ(100)
COMMON A(2000),IC(2000),K(100)
INTEGER N,M,I,J
REAL TOL

```

LECTURA Y ESCRITURA DE DATOS

```

READ(5,/) N,TOL
DO 10 I=1,N
  B(I)=K(I)
DO 15 I=1,N
  NR=NR+K(I)
  WRITE(6,*) NR
READ(5,/) IC(I),I=1,NR)
DO 20 I=1,NR
  A(I)=1
DO 40 I=1,NR
  WRITE(6,/) A(I),IC(I)
DO 60 I=1,N
  WRITE(6,/) B(I),K(I)

```

LLAMA SUBROUTINA Y ESCRIBE SOLUCIONES

```

CALL MINLG(N,B,Q0,Q1,AQ,X,Y,W,TOL,J)
WRITE (6,3)
FORMAT(3X,"SOLUCION DEL SISTEMA Y RESIDUAL")
DO 70 I=1,N

```

```

5900 70 WRITE(6,4) Y(I),R(I)
6000 4 FORMAT(8X,2(F8.4,5X))
6100 WRITE(6,5)
6200 5 FORMAT(3X,"NO. DE ITERACIONES")
6300 WRITE(6,*/).J
6400 CALL EXIT
6500 END
6600 C* *****
6700 C*
6800 SUBROUTINE MAQS(N,Q,AQ)
6900 DIMENSION Q(N),AQ(N)
7000 COMMON A(2000),IC(2000),K(100)
7100 INTEGER N,I,J,IJ,IK,JI
7200 DOUBLE PRECISION S1,S2,SUMA
7300 C*
7400 C* *****
7500 C*
7600 C* ESTA SUBRUTINA MULTIPLICA LA MATRIZ A POR UN VECTOR Q
7700 C* ALMACENANDO EL RESULTADO EN EL VECTOR AQ;
7800 C*
7900 C* ENTRADA:
8000 C*
8100 C* N; DIMENSION DEL SISTEMA
8200 C*
8300 C* Q; VECTOR POR EL CUAL SE MULTIPLICA LA MATRIZ
8400 C*
8500 C* COMMON:
8600 C*
8700 C* A; MATRIZ SPARSE EN GENERAL DADA COMO VECTOR
8800 C* ALMACENADA POR RENGLONES INDICANDO EL VAL-
8900 C* OR DEL ELEMENTO DISTINTO DE 0.
9000 C*
9100 C* IK; INDICE DE LA COLUMNA DE LOS ELEMENTOS DIS-
9200 C* TINTOS EN LA MATRIZ A.
9300 C*
9400 C* KI; ELEMENTOS DISTINTOS DE CERO EN EL RENGLON I
9500 C*
9600 C* SALIDA:
9700 C*
9800 C* AQ; VECTOR RESULTADO DEL PRODUCTO.
9900 C*
10000 C* *****
10100 C*
10200 IJ=1
10300 IK=0
10400 DO 10 I=1,N
10500 IK=IK+K(I)
10600 SUMA=0.0
10700 DO 20 JI=IJ,IK
10800 S1=A(JI)
10900 J=IC(JI)
11000 S2=Q(J)
11100 20 SUMA=SUMA+S1*S2
11200 AQ(I)=SUMA
11300 10 IJ=IK+1
11400 RETURN
11500 END
#

```

## VI. BIBLIOGRAFIA

- 1.- Serge Lang; Algebra Lineal; Addison-Wesley (1968).
- 2.- G.H. Golub, C.F. Van Loan; Matrix Computations; The Johns Hopkins University Press; cap 3. (1983).
- 3.- J. Stoer, R. Bulirsch; Introduction to Numerical Analysis; Springer-Verlag; cap 6; (1975).
- 4.- G. H. Golub, C.F. Van Loan; Matrix Computations; The Johns Hopkins University Press; cap 9; (1983).
- 5.- C.C. Paige, M.A. Saunders; A Bidiagonalization algorithm for Sparse Linear Equations and Least-Squares Problems; Technical Report Sol; 78-19 Oct 1978.
- 6.- C.C. Paige, M.A. Saunders; Solution of Sparse Indefinite Systems of Linear Equations; Siam, J. Number. Anal. 12, (1975), 617-629.
- 7.- G.D. Smith; Numerical Solution of Partial Differential Equations, Finite Difference Methods; Oxford, cap 5. (1978).
- 8.- C.C. Paige, M.A. Saunders; LSQR: Sparse Linear Equations and Least Squares Problems; ACM, vol 8, #2, June 1982, 195-209.
- 9.- W. Morven Gentelman; Least Squares Computations by Givens Transformations Without Squares Roots; J. Inst. Maths. Applies, (1974) 12, 239-336.
- 10.- C.C. Paige; Error Analysis of the Lanczos Algorithm for Tridiagonalizing a Symmetric Matrix; J. Inst. Maths. Applies (1976) 18, 341-349.
- 11.- C. Lanczos; an Iteration Method for the Solution of the Eigenvalue Problem of Linear Differential and Integral Operators; Journal of Research of the National Bureau of Standards; vol 45, #4, Oct. 1950, 255-282.

- 12.- C.C. Paige: Computational Variants of the Lanczos Method for Eigenproblem: J. Inst. Maths. Applies. (1972) 10,373-381.
- 13.- B.N. Parlett: A new look at the Lanczos Algorithm for solving Symmetric Systems of Linear Equations: Linear algebra and its Applications 22, (1980), 323-346.
- 14.- C. Lanczos: Solution of Systems of Linear Equations by Minimized Iterations: Journal of Research of the National Bureau of Standards, vol.49, #1, July 1952, 33-53.