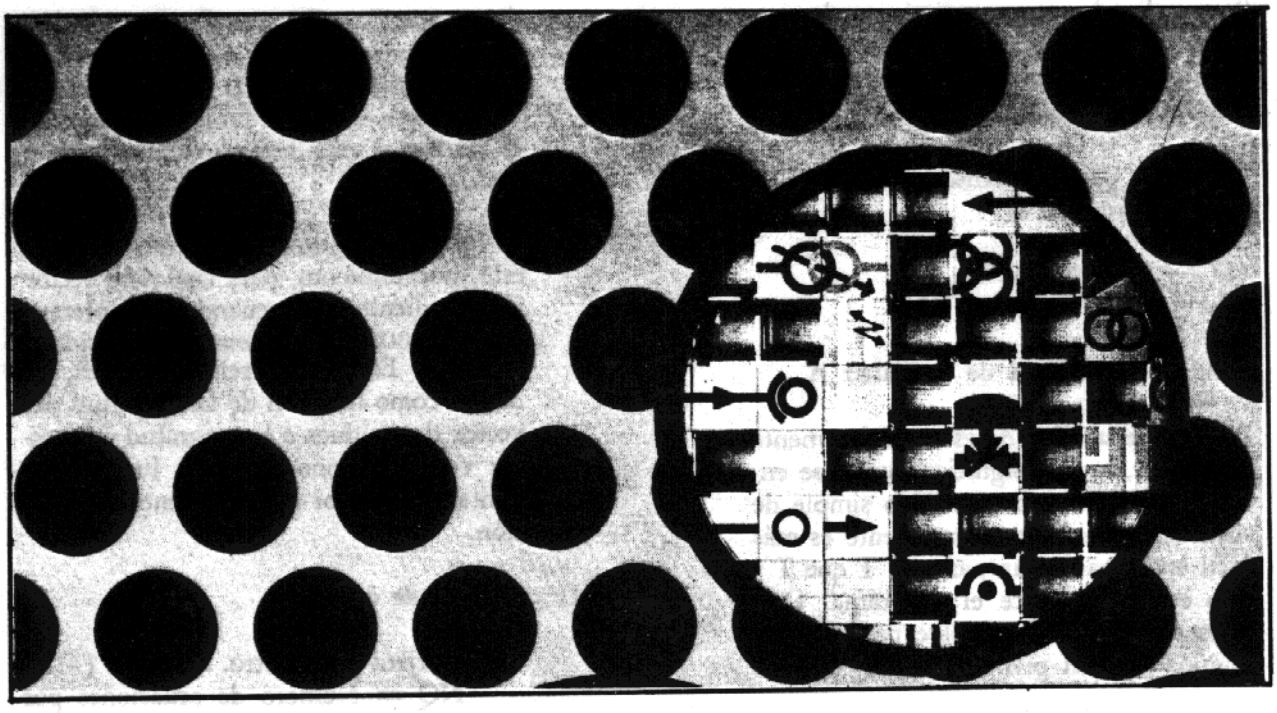


ALGORITMO PARA RESOLVER UN SISTEMA GRANDE Y DISPERSO DE ECUACIONES ALGEBRAICAS

Por el Lic. Guillermo García Gil.

I) Resumen.

Este algoritmo resuelve un sistema muy grande y disperso de ecuaciones algebráicas lineales. Lo disperso del sistema permite una pre-triangularización de las ecuaciones tal, que por medio de un método de partición pueden ser resueltas por una combinación de:

1) La solución de un pequeño sistema denso por eliminación del pivote máximo con iteración de los residuos y;
2) La solución de un grande sistema triungular por sustitución anterior.

II) Descripción detallada.

Antes que nada, un rearreglo sistemático, llamado pre-tringularización de las hileras y columnas de la matriz de coeficientes se lleva a cabo. Su finalidad es obtener una triangularización superior parcial de la esquina superior izquierda en dirección hacia la parte inferior derecha y al mismo tiempo una casi-triangularización inferior de la parte inferior izquierda en dirección a la esquina superior derecha, de tal forma que el espacio a la izquierda de estas dos diagonales resulte vacío.

Esto es, si la aparición de las ecuaciones después de la pre-triangularización es:

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix}\begin{pmatrix} X_1 \\ X_2 \end{pmatrix}=\begin{pmatrix} C_1 \\ C_2 \end{pmatrix} \ldots (1)$$

A es triangular superior, B y D son rectangulares y C puede ser descrita como casi-triangular inferior. Por casi-triangularización entendemos que ningún elemento principal de cualquier hilera de la matriz esté más a la derecha que el elemento principal de la hilera precedente.

En general entre más dispersa sea la matriz original, mayor sería la extensión a la que puede llevarse a cabo la pre-triangularización, en otras palabras el orden de A será mayor.

Lic. en Matemáticas, Analista Programador de la Gerencia de Informática de Petróleos Mexicanos.

Por medio de un procedimiento de partición, la solución del problema puede ser obtenida en dos partes como se mencionó anteriormente.

El sistema pequeño denso mencionado en 1) anteriormente es:

$$(D - C^* T^* B)^* X_2 = (C_2 - C^* T^* C_1 \ldots (2)$$

donde T = inversa de la matriz A

En esta ecuación A es completamente triangular, lo que significa que T se encuentra por un procedimiento simple de sustitución anterior. Actualmente es más fácil encontrar en un paso C* T que T y esto es lo que hace el programa. Después el sistema se resuelve para X_2 por eliminación del pivote máximo.

El sistema triangular grande mencionado en 2) anteriormente es:

$$A^* X_1 = (C1 - B^* X_2) \ldots (3)$$

Ahora se conoce X_2 y después que es substituido el sistema puede ser resuelto para X_1 por sustitución anterior puesto que A es triangular. Después de haber obtenido esta primera solución, el programa empieza a iterar sobre los residuos de la siguiente manera. Regresando a la ecuación (1) y sustituyendo X_1 y X_2 antes que nada se supone que los residuos $R1 = C1 - (A^* X_1 + B^* X_2) \ldots (4)$ obtenidos de la parte superior de la ecuación (1) son cero.

Esto quiere decir se supone que no se introduce error en la solución por sustitución anterior de la ecuación (3) que es nada menos que la parte superior de la ecuación (1) reordenada.

Los residuos $R2 = C2 - (C^* X1 + D^* X2) \ldots (5)$

Se obtienen por sustitución de X_1 y X_2 en la parte inferior de la ecuación (1). La solución de la ecuación se repite exactamente excepto que R1 = (o) se substituye por C_1 y R2 por C_2. La solución así obtenida se suma a la previamente obtenida para obtener la nueva solución corregida y el procedimiento completo se puede repetir tantas veces como se desée.

El programa automáticamente termina este procedimiento, para cualquier rector independiente dado, cuando una tolerancia de 0.000001 se obtiene en los residuos relativos. El residuo relativo, DRR, se define como la razón de la magnitud del vector de residuos a la magnitud del vector cuyos componentes son los componentes del vector independiente en cuestión.

III) Entrada de datos.

(1) Tarjeta para metro. Formato (415)
NQ = Número de ecuaciones por resolver hasta 999
NE = Número de elementos no-cero de la matriz de coeficientes hasta 4000.
NR = Número de vectores independientes hasta 5
NI = Número de iteraciones.

(2) Elementos no cero de la matriz de coeficientes en cualquier orden Formato (3(215, E15.5)).
Cada tarjeta contiene tres tripletas de números como sigue: posición de la hilera, posición de la columna, valor del elemento.

(3) Número de elementos no-cero en algún vector independiente Formato (15)

(4) Elementos no-cero del vector independiente Formato (3(15, E15.5)).
Cada tarjeta contine tres pares de números como sigue: Te Posición de la hilera valor del elemento.

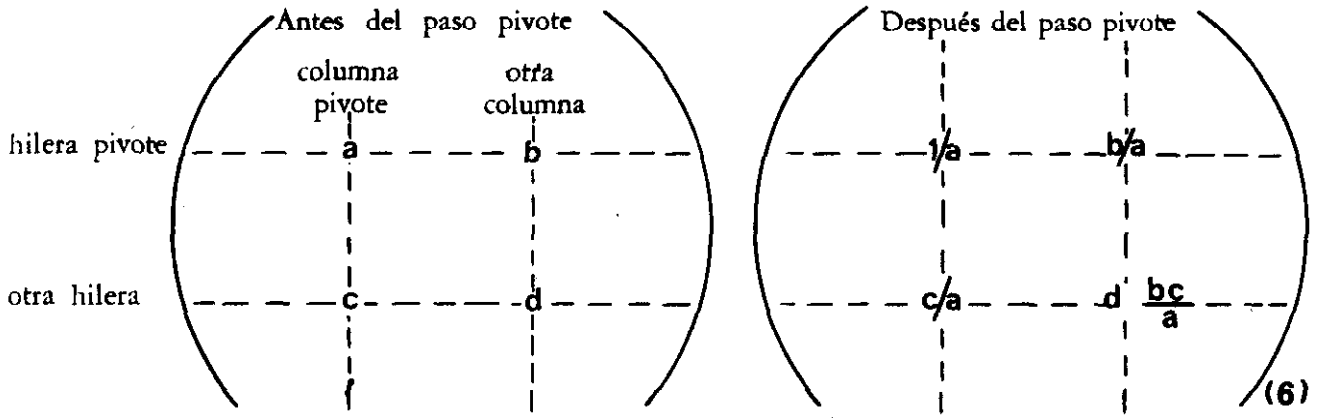
(5) Se repiten los pasos (3) y (4) para el número de vectores independientes deseado.

(6) Si otro conjunto de ecuaciones se

quiere resolver se repiten los pasos (1) a (5).

IV) Descripción del método por eliminación del pivote máximo.

Un paso pivote es la siguiente transformación de la matriz.



Se supone que el "elemento pivote" no es cero. Por ejemplo un paso pivote aplicado a la siguiente matriz con el elemento 10 en hilera 2 y columna 1 usándolo como pivote resulta en lo siguiente:

$$A = \begin{pmatrix} 50 & 0 & 0 & 0 \\ 10 & 0 & 20 & 0 \\ 0 & 0 & 0 & 0 \\ 30 & 0 & 60 & 5 \end{pmatrix} \qquad B = \begin{pmatrix} 5 & 0 & 100 & 0 \\ .1 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 5 \end{pmatrix} \qquad (7)$$

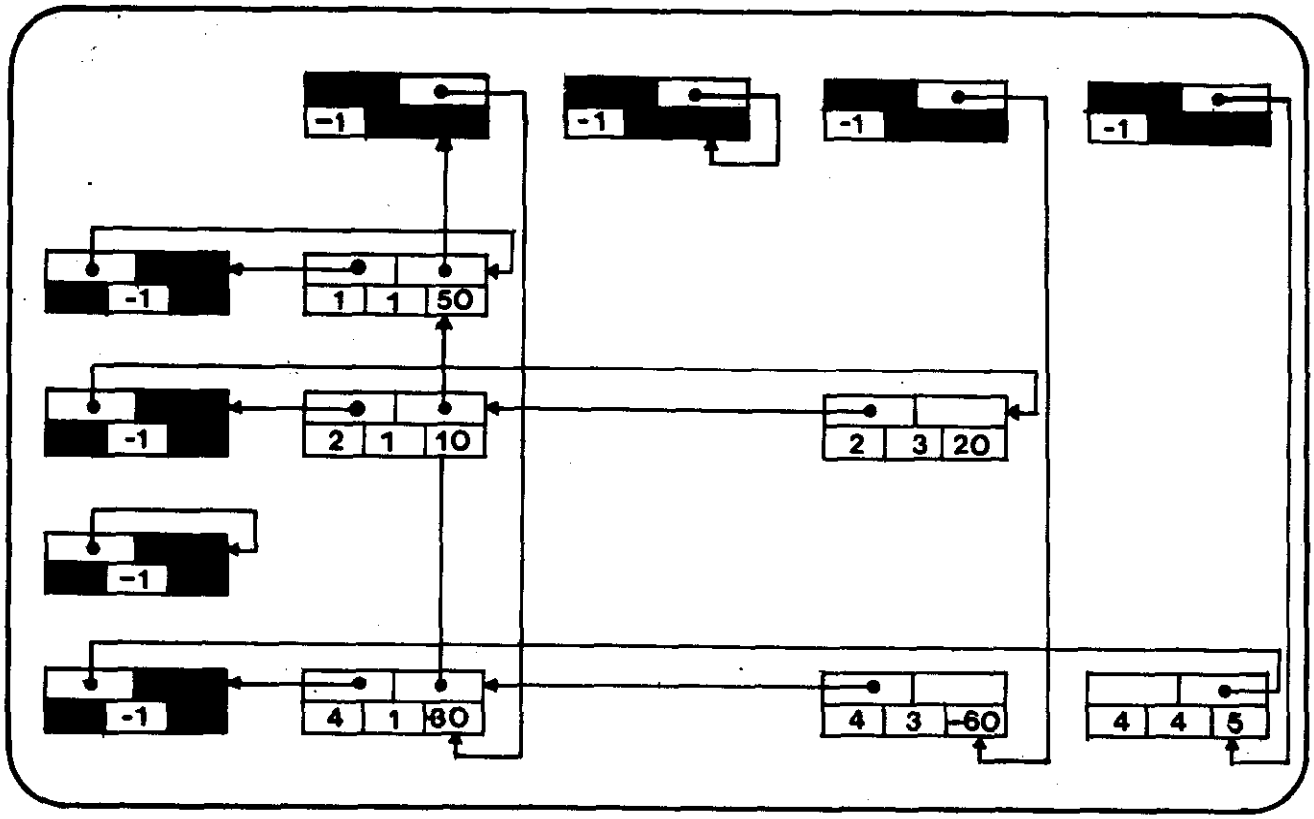
Nuestro propósito es diseñar un algoritmo que realice esta operación pivote en matrices dispersas que están representadas en la Fig. 1

—Representación de la matriz A: los nudos se muestran en el formato

fig. 1

LEFT		UP
KOW	COI	VAL

Es claro que la transformación pivote afecta solo aquellas hileras de la matriz en las cuales exista un elemento no-cero en la columna pivote, y solo afecta aquellas columnas para las cuales exista un elemento principal no-cero en el renglón pivote. Por lo tanto cuando una matriz dispersa grande es considerada, estamos logrando no solamente una reducción en espacio por la representación encadenada de los elementos no-cero sino también un aumento en la velocidad de pivotear.



El algoritmo pivote es en muchas maneras una aplicación directa de técnicas de encadenamiento; en particular se asemeja al algoritmo para suma de polinomios. Existen dos cosas sin embargo que hacen el problema un poco truculento: Si en (6) tenemos que b = 0 y c = 0 pero d = 0, la representación de la matriz dispersa no tiene entrada para d y debemos introducir una nueva entrada; y si b = 0, c = 0, d = 0 pero $d = \frac{bc}{a}$ = 0, debemos eliminar la entrada que estaba allí. Estas operaciones de insertar y eliminar en un arreglo bidimensional son más interesantes que el caso unidimensional; para efectuarlas debemos saber qué uniones se afectan. Nuestro algoritmo procesa las hileras de la matriz en forma secuencial de abajo hacia arriba. La habilidad eficiente para introducir y eliminar involucra la introducción de un conjunto de variables apuntadores PTR (j), una para cada columna considerada; estas variables atraviesan las columnas hacia arriba proporcionando la disponibilidad de actualizar los vínculos apropiados en ambas dimensiones.

V) Algoritmo S (paso pivote en una matriz dispersa). Dada una matriz como la de la Fig. 1, nosotros realizamos la operación pivote (6) suponga que PIVOT es una variable de enlace que apunta hacia el elemento pivote. El algoritmo hace uso de una tabla auxiliar de variables de enlace PTR (j) una para cada columna de la matriz S1. [Inicialización] IO ← Row (PIVOT), Jo ← Col (PIVOT), ALPHA ← 1.o/Val (PIVOT ← 1.0, P0 ← Loc (BASEROW (10)), QO ← LOC (BASE COL (JO)).

(NOTA: La variable ALPHA y el campo VAL de cada nodo se supone que son cantidades de punto flotante o racionales, mientras que el resto tiene valores enteros en este algoritmo).
S2. (Proceso de hilera pivote). Po ← LEFT (Po), J ← Col (Po), si J ← O siga el paso S3 (la hilera pivote ha sido atravesada).

Si no PTR (J)← Loc (BASECOL (J) y VAL (Po)← ALPHA X VAL (Po) y repita el paso S2.

S3. (Encontrar un nuevo renglón) QO← UP (QO). (El resto del algoritmo trata sucesivamente con cada hilera, de abajo hacia arriba, para la cual existe una entrada en la columna pivote).

I← Row (QO). Si I < O el algoritmo termina. Si I = IO, se repite el paso S3 (hemos realizado el renglón pivote). Si no P← Loc (BASEROW (I), PI← LEFT (P). (Los apuntadores P y P1 proceden a través de la hilera I de derecha a izquierda, a medida que Po va en sincronización a través del renglón IO; en este punto PO = Loc (BASEROW (IO))

S4. (Encontrar una nueva columna) PO← LEFT (PO), J← Col. (PO). Si J← O VAL (Qo)← ALPHA X VAL (Qo) y regresa el paso S3.

Si J = JO repite el paso S4. (Así procesamos la entrada de la columna pivote en el renglón I después que todas las entradas de la columna han sido procesadas; la razón es que VAL (Qo) se necesita en el paso S7.

S5. (Encuentre el elemento I. J). Si Col (PI)← J, P← Pl, Pl← LEFT (P) y repita el paso S5. Si Col (Pl) = J siga al paso. S7. Si no vaya al paso S6 (necesitamos introducir un nuevo elemento en la columna J de la hilera I).

S6. (Introducción del elemento I, J). Si Row (UP (PTR (J))) > I, entonces PTR (J) ← UP (PTR (J)) y repite el paso S6.

(Si no tendremos Row (UP (PTR (J))) < I ; el nuevo elemento va a ser introducido justamente arriba de NODE (PTR (J)) En la dimensión vertical, y justamente a laizquierda de NODE (P) En la dimensión horizontal).

De otra forma X← AVAIL, VAL (X)← O, Row (X)← I, Col (X) ← - J, LEFT (X)← Pl, UP (X)← UP (PTR (J)), LEFT (P)← X, UP (PTR (J))← X, Pl← X.

S7. (Pivote). VAL (Pl)← VAL (Pl) VAL (Qo) X VAL (Po). Si ahora VAL (Pl) = O, vaya a S8. (Nota: Cuando se usa aritmética de punto flotante, esta prueba "VAL (Pl) = O" deberá ser reemplazada por " | VAL (Pl) | <

EPSILON" o mejor aún por la condición "la mayor parte de las cifras significativas de VAL (Pl) se perdieron en la resta"). De otra manera PTR (J) Pl, P ← Pl, Pl ← LEFT (P) y regresa al paso S4.

S8. (Elimina el elemento I. J). Si UP (PTR (J)) ≠ Pl (o lo que es lo mismo, si Row (UP (PTR (J))) > I), PTR (J) ← UP (PTR (J)) y repite el paso S8; Si no UP (PTR (J)) ← UP (Pl), LEFT (P)← LEFT (Pl), AVAIL← Pl, Pl ← LEFT (P). Regresa a S4.

La computación con matrices dispersas no es nueva. Técnicas iterativas para estas matrices especialmente aquellas relacionadas con la solución de ecuaciones diferenciales parciales han sido extensamente desarrolladas. Métodos de matrices dispersas para resolver ecuaciones lineales por métodos directos han sido usados por mucho tiempo. En programación lineal hay una gran cantidad de literatura, experiencia computacional, programas y habilidad que ha sido desarrollado en esta área recientemente, en un número de otras áreas las aplicaciones, interés creciente ha surgido en métodos de matrices dispersas, a saber, circuitos eléctricos, ingeniería estructural y sistemas de distribución de potencia.

La razón de este interés se debe al intento para resolver problemas muy grandes los cuales en turno parecen inspirarse por

la disponibilidad de computadoras rápidas y grandes.

Referencias:

1.—The art of computer programming Vol. 1 Fundemental Algorithms by Donald Knuth Addisson Wesley.

2.—"Some Resultson Spars Metrices" by R. K. Brayton, Fred G. Gustavson y R. A. Willoughby Mathematics of Computation. 1971.