13
2ej

UNIVERSIDAD NACIONAL AUTONOMA
DE MEXICO

FACULTAD DE CIENCIAS

" PAQUETE DE PROGRAMAS PARA RESOLVER PROBLEMAS DE PROGRAMACION MATEMATICA "

T E S I S

QUE PARA OBTENER EL TITULO DE:

A C T U A R I O

P R E S E N T A :

JOSE LUIS CECILIANO MEZA

MEXICO, D. F. FEBRERO 1993

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

CONTENIDO

CAPITULO 4

CAPITULO 5

CAPITULO 6

PREFACIO

Con el fin de apoyar científica y tecnológicamente a la industria eléctrica del país, se creó el Instituto de Investigaciones Eléctricas (IIE). Este organismo está estructurado por Divisiones, entre la cuales se encuentra la División de Sistemas de Control. La División de Sistemas de Control está formada por cuatro departamentos; uno de ellos es el Departamento de Análisis de Redes, en donde fue elaborado este trabajo.

Gran parte de los proyectos del Departamento están relacionados con la operación de redes eléctricas y la planeación de su operación o expansión, considerando la utilización óptima de recursos de manera que el sistema eléctrico opere con economía y seguridad. Esto, se formaliza con planteamientos matemáticos que dan origen, muchas veces, a problemas muy complejos. Los problemas que se resuelven son de naturaleza lineal, no lineal y mixtos, pudiendo ser de gran escala (muchas variables y restricciones en el problema). La única manera de resolver estos problemas es aplicando técnicas de optimización matemática, estudiando e implantando algoritmos eficientes, que simplifican o reducen la complejidad del problema.

Existen en el mercado paquetes de cómputo utilizados para resolver problemas de optimización, programas en Fortran o Algol para realizar cálculos del Simplex, así como toda una gran cantidad de códigos con los mismos propósitos. Para las necesidades de los investigadores del Departamento se requiere poder manipular, y en ocasiones modificar el contenido de dichos programas para incorporarlos como subrutinas en programas de aplicación más complejos. Esto no es siempre fácil de obtener de un paquete comercial, limitado a un número de variables y de restricciones, y del que sólo se tiene un programa ejecutable.

Es por eso que se desarrolla en el Departamento, como un proyecto de infraestructura, una biblioteca especializada de programas para los proyectos del área de optimización matemática.

INTRODUCCION

OBJETIVO DE LA TESIS

El objetivo de esta tesis es hacer funcionar un paquete de cinco programas que resuelvan problemas de optimización; cuatro de estos programas se encuentran en el libro "Fortran Code for Mathematical Programming: Linear, Quadratic and Discrete" de H. A. Land y S. Powell; están en Fortran IV y la actividad desarrollada fue implantarlos a Fortran 77 estructurado, documentarlos, hacerlos funcionar, probarlos y dejar un documento en donde se describe completamente a cada uno de ellos. Además, aprovechando la estructura de dichos programas se desarrolló el código necesario para obtener un quinto programa que resuelva problemas lineales de gran escala con la estructura bloque angular, utilizando el método de descomposición de Dantzig-Wolfe.

ALCANCES Y LIMITACIONES

Los programas tienen un procedimiento de reinversión y reglas de paro para prevenir la obtención de soluciones falsas. Además se desarrolló el código necesario para incorporar un procedimiento de escalamiento en el programa LINP (programación lineal), con el propósito de evitar problemas de estabilidad y precisión numérica.

El código se encuentra en archivos de las terminales VAX del IIE, éste es de uso interno y por lo cual no se incluye completamente en la tesis, además de que representaría un volumén notable en ésta. Los programas pueden ser dimensionados y organizados a las necesidades de los investigadores. Como petición del Departamento, se trabajó para que los programas pudiéran correr en una computadora personal, de ciertas características. Este trabajo representa un diskette, el cual contiene un ejecutable de cada programa con dimensiones definidas, para resolver problemas de mediana escala.

CONTENIDO

Es importante que el usuario de los programas tenga conocimientos previos de los temas tratados en esta tesis, no obstante, en cada capítulo se deja una sección para conceptos básicos de cada tema. En el capítulo 1 se describe al programa LINP (programación

lineal). Está compuesto por una sección de conceptos básicos de programación lineal, una descripción del algoritmo utilizado, la estructura del programa, el formato para datos de entrada, el concepto de escalamiento y el manual de usuario.

El capítulo 2 habla del programa BB (Branch and Bound), y contiene una sección de conceptos básicos de programación entera mixta, una descripción del algoritmo utilizado, la estructura del programa, el formato para datos de entrada y el manual de usuario.

En el capítulo 3 se describe al programa PLP (programación lineal paramétrica), éste contiene una sección de conceptos básicos de programación lineal paramétrica, la descripción del algoritmo utilizado, la estructura del programa, el formato para datos de entrada y el manual de usuario.

El capítulo 4 se refiere al programa QP (programación cuadrática) y contiene una sección de conceptos básicos de programación cuadrática, una descripción del algoritmo utilizado, la estructura del programa, el formato para datos de entrada y el manual de usuario.

En el capítulo 5 se describe al programa DDW (descomposición de Dantzig-Wolfe), contiene conceptos básicos de este método de descomposición, una descripción del algoritmo utilizado, la estructura del programa, el formato para datos de entrada y el manual de usuario.

Por último, se incluyen los resultados de este trabajo y unas recomendaciones, que a juicio son pertinenetes, para trabajos posteriores con los programas.

CAPITULO 1

PROGRAMA LINP (PROGRAMACION LINEAL)

1.1 INTRODUCCION

La descripción de variables, parámetros, el código fuente y problemas de prueba, se encuentran en un reporte interno que se hizo del programa, para el Departamento [10]. El método utilizado para resolver los problemas es el método simplex revisado.

Los problemas deben ser del tipo de maximizar (min w = -max -w) y no necesitan estar en forma estándar. Las restricciones pueden ser cualquier combinación de $\leq, \geq$ ó $=$.

La dimensión del problema más grande que resuelve el programa, para efectos del diskette, es de 30 restricciones y 50 variables. Si alguna variable está acotada superiormente, el programa está diseñado para que tal situación no represente una restricción más en el problema. Toda la aritmética se realiza en doble-precisión, pero por razones de espacio en los formatos de salida, los resultados se escriben en formatos de 12 cifras con cuatro decimales como máximo (F12.4), para reales y cinco cifras para enteros (I5).

Al programa LINP le fueron agregadas 7 rutinas con el fin de escalar y desescalar los problemas que se desean resolver. La idea es escalar la matriz de restricciones, el lado derecho de las restricciones y el vector de la función objetivo, para poder reducir posibles errores de precisión en la solución obtenida de problemas desescalados. Por ejemplo, problemas cuyos coeficientes de las restricciones se encuentren en un rango de $(10^{-6}, 10^{6})$, cambiarlas al rango $(-1,1)$.

El programa obtiene los datos del problema de un archivo LINP.DAT y los resultados se dan en el archivo de salida LINP.RES, como se muestra en la siguiente figura.

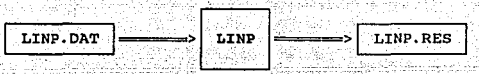


Fig 1.1. Diagrama de flujo de información

1.2 CONCEPTOS BASICOS

Un problema de programación lineal es un problema matemático en el cual la función objetivo es lineal en las incógnitas y las restricciones constan de igualdades y desigualdades lineales. La forma exacta de estas restricciones puede variar de unos problemas a otros, pero cualquier programa lineal se puede convertir a la siguiente forma estándar

$$
\begin{array}{ll}
\text{maximizar} & c_1x_1+c_2x_2+\ldots+c_nx_n \\
\text{sujeto a} & a_{11}x_1+a_{12}x_2+\ldots+a_{1n}x_n = b_1 \\
 & a_{21}x_1+a_{22}x_2+\ldots+a_{2n}x_n = b_2 \\
 & \qquad\qquad\vdots \\
 & a_{m1}x_1+a_{m2}x_2+\ldots+a_{mn}x_n = b_m \\
\text{y} & x_1, x_2, \ldots, x_n \geq 0
\end{array}
$$

donde las b_i, c_i y a_{ij} son constantes reales fijas, y las x_i son valores reales a determinar. Se supondrá que todo $b_i \geq 0$, en caso necesario, cada ecuación se multiplica por menos uno. En notación vectorial, el problema puede expresarse como

$$
\begin{array}{ll}
\text{maximizar} & \overline{c}\,\overline{x} \\
\text{sujeto a} & A\overline{x} = \overline{b} \\
\text{y} & \overline{x} \geq \overline{0}
\end{array}
$$

Aquí $\overline{x}$ es un vector columna n-dimensional, $\overline{c}$ es un vector renglón n-dimensional, A es una matriz de m x n, y $\overline{b}$ es un vector columna

m-dimensional. La desigualdad vectorial $\overline{x} \geq \overline{0}$ quiere decir que las componentes de $\overline{x}$ son no negativas.

Un vector x que satisfaga todas las restricciones, se llama vector (punto) factible. El conjunto de estos puntos constituye la región o espacio factible. El problema de programación lineal puede ser establecido como: Dentro de todos los vectores factibles encontrar el que maximice la función objetivo.

1.3 ESCALAMIENTO

El procedimiento escalamiento [17] se usa para referirse a dificultades numéricas que pueden surgir durante la solución de un problema, y consiste en transformar el problema que se quiere resolver en otro equivalente con características numéricas uniformes (todos sus coeficientes dentro de un rango pequeño).

Se puede tener un programa tan robusto y confiable como se piense, pero no falta un problema cuyas características numéricas (términos muy pequeños y muy grandes) causen dificultades a tal programa; además, no todos los paquetes comerciales incluyen un procedimiento para escalar los problemas que resuelven.

1.3.1 Escalamiento por transformación de variables

Este tipo de escalamiento convierte las variables, de unidades que reflejan la naturaleza física del problema, a otras que ofrecen ciertas propiedades numéricas deseables durante la maximización (minimización) del problema.

Existe una diferencia entre transformar variables para mejorar la conducta de un método de optimización y cambiar la categoría de un problema. Una regla básica de escalamiento es que, las variables del problema, ya escalado, deben ser de similar orden y magnitud que la unidad en la región de interés.

En las rutinas de optimización, tanto la convergencia a la solución como otros criterios, están basados o dependen de la definición de "pequeño" y "grande", de este modo las variables con gran variación en orden de magnitud pueden causar dificultades a algunos algoritmos. Si los valores típicos de todas las variables son

conocidos, un problema puede ser transformado en otro en donde todas las variables sean del mismo orden de magnitud.

Considérese un problema de un generador de calor, expresado en términos de gas\agua. La siguiente tabla muestra algunas de sus variables, su interpretación y un valor típico para cada una de ellas; las magnitudes en las variables surgen de las unidades en las cuales son expresadas.

VALORES TIPICOS DE VARIABLES NO-ESCALADAS

CARACTERISTICAS	VARIABLES	UNIDADES	VALORES TIPICOS
flujo de gas	X_1	*lbs/hr*	11,000
flujo de agua	X_2	*lbs/hr*	1,675
resistencia térmica al vapor desgaste de	X_3	$(BTU/(hr(pie^2)°F))^{-1}$	100
la estructura	X_4	$(BTU/(hr(pie^2)°F))^{-1}$	$6X10^{-4}$
radiación de gas	X_5	$(BTU/(hr(pie^2)°F))^{-1}$	$5.4X10^{-10}$

Como casi todas las variables son medidas en diferentes unidades físicas, no existe razón para suponer que deberían ser de tamaño similar; aún cuando las unidades físicas de medida son las mismas, puede existir gran diferencia en valores típicos. Normalmente sólo transformaciones lineales de las variables se usan para escalar, pero algunas veces transformaciones no lineales son posibles.
La transformación más común es de la forma

$$x = Dy$$

donde (x_j) son las variables originales, (y_j) son las variables transformadas y D es una matriz diagonal.
Para las variables dadas en la tabla, un escalamiento puede realizarse, haciendo d_j, el j-ésimo elemento en la diagonal de D, a un valor típico de la j-ésima variable. Por principio d_j *puede ser* $1.1\ X\ 10^{-4}$. Otra alternativa es conocer un rango real para los valores que una variable puede tomar. Un rango tal puede ser las restricciones de cota superior e inferior que son impuestas a las variables. Si una variable x_j, está siempre en el rango $a_j \leq x_j \leq b_j$, una nueva variable y_j puede ser definida como :

$$y_j = \frac{2x_j}{b_j - a_j} - \frac{a_j + b_j}{b_j - a_j}$$

Esta transformación se describe en la forma matricial

$$x = Dy + c$$

donde D es una matriz diagonal con j-ésimo elemento $\frac{1}{2}(b_j - a_j)$, y c es un vector con j-ésimo elemento $\frac{1}{2}(a_j + b_j)$. La transformación garantiza que $-1 \leq y_j \leq 1 \quad \forall j$, si $x_j \in [a_j, b_j]$.

Por ejemplo, para una variable que está en el rango [200.1242,200.1806], la transformación apropiada es

$$x_j = 0.0282 y_j + 200.1524,$$

lo cual permite a y_j estar en el rango [-1,1].

1.3.2 Escalamiento de la función objetivo

Escalar la función objetivo puede no ser necesario, ya que en teoría, la solución de un problema no es alterada si f(x) se multiplica por una constante positiva o si un valor constante se suma a ésta. Cuando la magnitud de f es grande en todos los puntos de interés, ésta se puede multiplicar por una constante adecuada y entonces reducir su magnitud. Por ejemplo, si f(x) es del orden de 10^5, la constante adecuada para multiplicar sería 10^{-5}.

1.3.3 Escalamiento de restricciones

Considérese un problema de dos variables, en donde el valor del producto interno $a_1 x$ representa la cantidad de oxígeno disuelto en agua, con un valor típico de una parte por millón, y el producto interno $a_2 x$ representa la velocidad de flujo del agua con un valor típico de 5 millas por hora. Supóngase que estos productos internos definen los rangos de restricciones

$$0 \leq a_1 x \leq 8 \; X \; 10^{-6}$$

$$3 \leq a_2 x \leq 10$$

y que la matriz A, de 2x2, con renglones a_1 y a_2, está dada por

$$A = \begin{pmatrix} 10^{-6} & -10^{6} \\ 1 & 1 \end{pmatrix}$$

Las restricciones pueden ser balanceadas multiplicando la primera restricción por 10^6, con este escalamiento (transformación de la variable que presenta valores "muy pequeños" o "muy grandes", en una equivalente con valores del orden de la unidad), la matriz A llega a ser

$$\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$$

la cual está bien condicionada y tiene columnas ortogonales.

Esta idea puede servir para originar un procedimiento de escalamiento. Otra idea es escalar renglones y columnas de la matriz, ignorando el escalamiento de la función objetivo. Esquemas de este último tipo son de uso frecuente en programación lineal y cuadrática, en donde la matriz de coeficientes A de mxn, se multiplica por

$$D_1 A D_2$$

siendo D_1 y D_2 matrices diagonales positivas. A continuación se describe un algoritmo posible, para calcular estas matrices. Su justificación se encuentra en Murray W., Practical Optimazation, [17].

ALGORITMO PARA ESCALAR RESTRICCIONES LINEALES

PASO 1 (Cálculo del cociente más grande entre dos elementos de la misma columna).
Calcular ρ_0, el cual esta definido como :

$$\rho_0 = \max_j \max_{r,s} |a_{r,j}/a_{s,j}| \quad j=1,\ldots,n; \quad r,s=1,\ldots,m$$

donde $a_{s,j} \neq 0$.

PASO 2 (Escalamiento de renglón).
Dividir cada renglón i de A por $((\min_j|a_{i,j}|)(\max_j|a_{i,j}|))^{1/2}$, donde el mínimo es tomado sobre todos los $a_{i,j} \neq 0$. Obteniendoce D_1A.

PASO 3 (Escalamiento de columna).
Dividir cada columna j de D_1A por $((\min_i|a_{i,j}|)(\max_i|a_{i,j}|))^{1/2}$, donde el mínimo es tomado sobre todos los $a_{i,j} \neq 0$. Obteniéndose D_1AD_2.

PASO 4 (Cálculo del cociente más grande entre dos elementos de la misma columna).
Calcular ρ de D_1AD_2, el cual está definido como :

$$\rho = \max_j \max_{r,s} (|a_{r,j}/a_{s,j}|)$$

donde $a_{s,j} \neq 0$.

PASO 5 (Criterio de paro).
Si $\frac{|\rho - \rho_0|}{|\rho_0|} \geq 10^{-1}$ regresar al paso 1, de otra manera el algoritmo termina.

Este algoritmo fue tomado como base para desarrollar el código (apéndice A), que se incorporó en 7 subrutinas al programa LINP. La transformación en los problemas originales fue la siguiente.

Dado el problema de programación lineal

$$\begin{aligned} &\text{Max } z = && cx \\ &\text{sujeto a} && Ax \leq b \\ & && 0 \leq x \leq u \end{aligned}$$

y su dual

$$\begin{aligned} &\text{Min } w = && yb \\ &\text{sujeto a} && yA \geq c \\ & && y \geq 0 \end{aligned}$$

donde u es un vector columna de cotas superiores para las variables.

Se pueden realizar las operaciones siguientes, sin alterar la naturaleza de los problemas originales

$$\begin{aligned} \text{Max } z = \quad & (cD_2)\,(D_2^{-1}x) \\ \text{sujeto a} \quad & (D_1AD_2)\,(D_2^{-1}x) \le D_1b \\ & 0 \le D_2^{-1}x \le D_2^{-1}u \end{aligned}$$

$$\begin{aligned} \text{Min } w = \quad & (yD_1^{-1})\,(D_1b) \\ \text{sujeto a} \quad & (yD_1^{-1})\,(D_1AD_2) \ge cD_2 \\ & yD_1^{-1} \ge 0 \end{aligned}$$

con lo que se obtiene los problemas equivalentes

$$\begin{aligned} \text{Max } z = \quad & c'x' \\ \text{sujeto a} \quad & A'x' \le b' \\ & 0 \le x' \le u' \\ \text{Min } w = \quad & y'b' \\ \text{sujeto a} \quad & y'A' \ge c' \\ & y' \ge 0 \end{aligned}$$

con

$c' = cD_2$

$b' = D_1b$

$A' = D_1AD_2$

$x' = D_2^{-1}x$

$u' = D_2^{-1}u$

$y' = yD_1^{-1}$

Estos problemas se encuentran ya escalados, y son los que resuelve el programa LINP.

1.4 ALGORITMO

El procedimiento que utiliza el programa LINP para resolver los problemas, basado en el método simplex revisado [14], se describe en los pasos siguientes.

PASO 1 Se escala el problema a resolver.

PASO 2 Se forma la base inicial agregando una variable de holgura en la primera restricción.

PASO 3 Se actualizan los arreglos asociados con la base actual y se verifican todas las restricciones, por si no existe factibilidad en alguna.

Si todas las restricciones son factibles ir al paso 5. De lo contrario, si la no factibilidad existe en una restricción no presente en la base, aumentar la inversa por un renglón y una columna.

PASO 4 Determinar la variable que entra a la base para reducir la no factibilidad e ir al paso 6.

Si no existe tal variable, el problema es no factible. Ir al paso 9.

PASO 5 Se determina si la base actual es óptima.

Si la solución es óptima ir al paso 9; de lo contrario se busca la variable que entre a la base para mejorar el valor de la función objetivo.

PASO 6 Se calcula la columna asociada a la variable que entrará a la base.

PASO 7 Se determina la variable que saldrá de la base.

Si tal variable no existe, el problema es no acotado ir al paso 9. De lo contrario, si esta variable está asociada con una restricción no presente en la base, se agrega un renglón y una columna a la inversa.

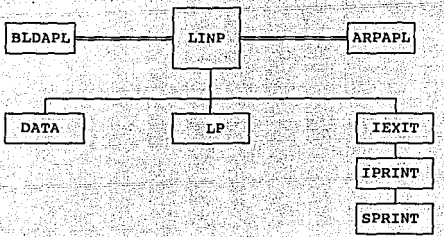
PASO 8 Se realiza el cambio de base e ir al paso 3.

PASO 9 Se transforma el problema a sus unidades originales (desescalar el problema), y se escriben resultados.

1.5 ESTRUCTURA DEL PROGRAMA

El programa está compuesto por 37 rutinas, formando 5645 líneas de código. Su estructura se representa en los siguientes diagramas, y además se describe el objetivo de cada rutina.

i)



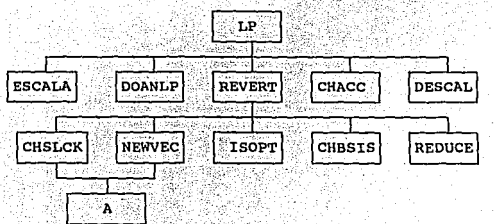
LINP Programa principal.

ARPAPL Archivo que tiene como objetivo definir el tamaño de los parámetros que utiliza el programa, éstos se pueden cambiar para resolver problemas con un mayor número de restricciones y variables.

BLDAPL Archivo en el que se define un conjunto de variables dimensionadas, comunes dentro del programa.

SUBRUTINA	OBJETIVO
DATA	Realizar la lectura de datos del problema de programación lineal (PL). Los datos son leídos del archivo LINP.DAT.
LP	Llamar a la subrutina DOANLP para resolver el problema PL. Si existe solución al problema (óptima, no acotada o no factible), la subrutina CHACC verifica su exactitud; si ésta no es correcta, se realiza una reinversión.
IEXIT	Escribir los mensajes correspondientes a la solución del problema, terminando así la ejecución del programa.
IPRINT	Escribir casi todos los valores de las variables globales (COMMON's), incluyendo las referidas a la matriz A.
SPRINT	Escribir el problema en su forma original (no en forma condensada), cuando el número de variables es menor que ocho.

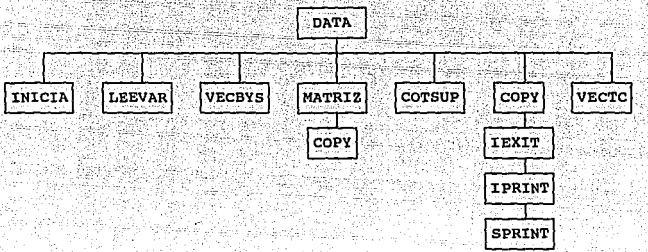
ii)



SUBRUTINA	OBJETIVO
ESCALA	Transformar el problema original en un problema equivalente, cuya característica es estar escalado.
DOANLP	Llamar a otras subrutinas para resolver el problema, obteniendo, si es que existe, una solución final.
REVERT	Invertir la base actual y reiniciar la subrutina DOANLP con una nueva solución básica.
CHACC	Comprobar la exactitud en la solución. Al realizar los cálculos de YA-C (restricciones del dual) en las variables básicas, y B-Ax-SLACK (restricciones del primal) en todas las restricciones, se desea que éstos sean cero o que no difieran por más de una tolerancia establecida; de no ser así, la solución encontrada no es exacta.
DESCAL	Volver el problema a su forma original y transformar los resultados obtenidos a su valor real.
CHSLCK	Actualizar los vectores Y (valor de las variables duales), YA-C y SLACK (valor de las variables básicas de holgura), determinando si éste último es factible o no.
NEWVEC	Calcular la columna asociada a la variable (NEWX), que entrará a la base.
ISOPT	Determinar si una solución factible básica es óptima (NEWX=0); si la solución no es óptima, la rutina busca la variable que entre a la base para mejorar el valor de la función objetivo.
CHBSIS	Realizar el cambio de base en la matriz inversa y vectores asociados con ésta. Actualizar el valor de la función objetivo.

REDUCE	Borrar una variable de holgura de la inversa, después de un cambio de base.
A	Buscar el valor del elemento a_{ij} de la matriz A, en el arreglo AA (vector que contiene los elementos distintos de cero de la matriz A).

iii)

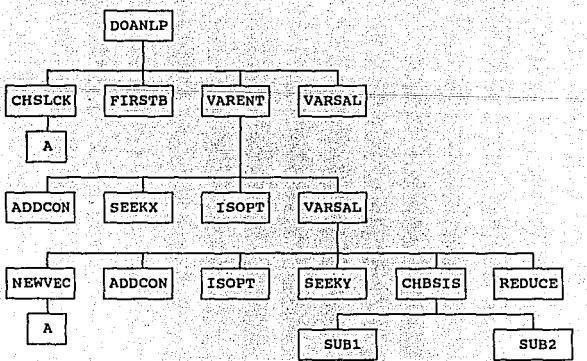


SUBRUTINA	OBJETIVO
INICIA	Inicializar las variables globales, que se utilizan en el programa.
LEEVAR	Leer del archivo LINP.DAT el número de: restricciones (M), variables (N) y variables acotadas superiormente (ISBND) en el problema; así como el número máximo de iteraciones (ITRMAX) y reinversiones (IRMAX), y la decisión de imprimir o no la inversa (MOREPR).
VECBYS	Llenar el vector B (lado derecho de las restricciones) y el vector S (signo de las restricciones), con los valores leídos del

archivo LINP.DAT.

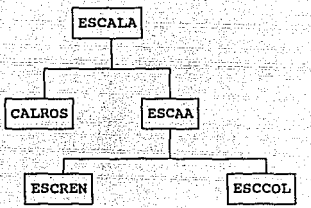
MATRIZ	Guardar la matriz de coeficientes (A), del problema que se quiere resolver, en forma compactada (sin elementos iguales a cero).
COTSUP	Llenar el vector BOUND con el valor de las cotas superiores, si ISBND (número de variables con cota superior) es mayor que cero.
COPY	Copiar los elementos distintos de cero de un renglón de la matriz A, dentro del vector AA.
VECTC	Asignar a cada elemento del vector c (coeficientes de la función objetivo) el valor leído del archivo LINP.DAT.

iv)



SUBRUTINA	OBJETIVO
FIRSTB	Formar la base inicial, agregando una variable de holgura a la primera restricción del problema.
VARENT	Determinar la variable que debe entrar a la base, si es que no existe factibilidad o no se tiene una solución óptima.
VARSAL	Llamar a otras subrutinas para determinar la variable que saldrá de la base; entonces se realiza el cambio de base y se reduce la inversa por un renglón y una columna.
ADDCON	Aumentar la matriz inversa por un renglón y una columna.
SEEKX	Determinar la variable que entrará a la base (NEWX), para obtener una solución básica factible. Si NEWX=0, entonces no existe solución básica factible para el problema (problema no factible).
SUB1	Realizar el cambio de base, actualizando la matriz inversa y el valor de las variables duales.
SUB2	Terminar el cambio de base y actualizar el valor de la función objetivo.
SEEKY	Determinar la variable que saldrá de la base; tal variable está asociada con la variable NEWY (renglón de la matriz inversa), si NEWY=0 entonces el problema es no acotado.

v)



SUBRUTINA	OBJETIVO
CALROS	Calcular el máximo cociente entre dos elementos de una misma columna de la matriz A.
ESCAA	Escalar la matriz de restricciones que está ya en su forma compactada (vector AA).
ESCREN	Realizar el escalamiento en los renglones de la matriz A.
ESCCOL	Realizar el escalamiento en las columnas de la matriz A.

vi)

SUBRUTINA	OBJETIVO
DESCAA	Desescalar la matriz de restricciones A.

1.6 FORMATO PARA DATOS DE ENTRADA (ARCHIVO LINP.DAT)

En esta sección se describe cómo crear el archivo LINP.DAT, con los datos del problema que se quiere resolver.

Registro 1

Columna		
1-10	M	Número de restricciones.
11-20	N	Número de variables.
21-30	ISBND	Número de variables acotadas.
		= -1 todas las variables tendrán cota superior de 1.0.
31-40	MOREPR	= 0 imprime los datos de entrada, pero no imprime la inversa en la rutina IPRINT.
		= 1 imprime los datos de entrada y la inversa.
		= 2 no imprime los datos de entrada pero imprime la inversa.
		= 3 no imprime los datos de entrada ni la inversa.
41-50	ITRMAX	Número máximo de iteraciones; si es cero, éste tendrá un valor heurístico de 3*(M+N+ISBND).
51-60	IRMAX	Número máximo de reinversiones.

Registro 2

Columna		
1-3	J	Número de un elemento diferente de cero de la función objetivo (Vector C).

5-10	C(J)	j-ésimo elemento de la función objetivo.
11-13	J	
15-20	C(J)	
	.	
	.	
	.	
71-73	J	
75-80	C(J)	

El registro 2 se repite hasta cubrir todos los elementos diferentes de cero de la función objetivo, ocho por registro.

Registro 3

Columna		
1-10	999999999	Indica el fin de los elementos de C.

Sí ISBND es mayor que cero; es decir, si existen variables del problema acotadas superiormente, el siguiente registro es el 4. De lo contrario el siguiente será el 6.

Registro 4

Columna		
1-3	J	Número de una variable acotada.
5-10	BOUND(J)	Valor de la cota superior de la j-ésima variable.
11-13	J	
15-20	BOUND(J)	
21-23	J	

25-30	BOUND(J)	
	.	
	.	
71-73	J	
75-80	BOUND(J)	

El registro 4 se repite hasta cubrir todos los elementos que tienen cota superior, ocho por registro.

Registro 5

Columna		
1-10	999999999	Indica el fin de los elementos con cota superior.

Registro 6

Columna		
1-3	I	Número de un elemento del vector b, lado derecho.
4	S(I)	= 0 restricción de igualdad.
		= 1 restricción de menor o igual.
		= 2 restricción de mayor o igual.
5-10	B(I)	i-ésimo elemento del vector b.
11-13	I	
14	S(I)	
15-20	B(I)	

71-73	I
74	S(I)
75-80	B(I)

El registro 6 se repite hasta cubrir todos los elementos del vector b, ocho por registro. Cualquier elemento que no sea especificado, el programa asume como restricción de menor o igual con un valor muy grande para el elemento de B.

Registro 7

Columna		
1-10	999999999	Indica el fin de los elementos del vector b.

Registro 8

Columna		
5-10	I	Número de un renglón de la matriz A.
11-13	J	Número de una columna de la matriz A.
15-20	A(IJ)	Elemento diferente de cero de la matriz A.
21-23	J	
25-30	A(IJ)	
	.	
	.	
	.	
71-73	J	
75-80	A(IJ)	

Todos los elementos del registro deben pertenecer al mismo renglón, pero no necesariamente en orden correcto en las columnas. Los renglones deben meterse en el orden correcto. El registro 8 debe ser repetido hasta cubrir con todos los elementos diferentes de cero de la matriz A, siete por registro.

Registro 9

Columna		
1-10	999999999	Indica el fin de los elementos diferentes de cero de la matriz A.

1.6.1 Ejemplo de un archivo LINP.DAT

Dado el problema

$$\begin{aligned} \text{Max } z = \quad & x_1+3x_2+10x_3 \\ \text{sujeto a} \quad & 12x_1+5x_2+30x_3 \leq 120 \\ & 2x_1+10x_2+30x_3 \leq 95 \\ & x_1 \geq 0,\ x_2 \geq 0,\ 0 \leq x_3 \leq 2. \end{aligned}$$

con dos restricciones, tres variables y una de ellas acotada superiormente. Además, pensando en un número máximo de iteraciones heurístico, un número de reinversiones cero y deseando que se impriman los datos de entrada y la matriz inversa; el archivo LINP.DAT correspondiente es :

```
Columna 10        20        30        40        50        60...80

        2         3         1         1         0         0
  1     1  2      3  3      10
9999999999
  3     2
9999999999
 11   120  21     95
9999999999
        1  1      12  2      5  3      30
        2  1       2  2     10  3      30
9999999999
```

1.7 MANUAL DE USUARIO

El programa está diseñado para correr en una IBM-PC o compatible, con sistema operativo MS-DOS 3.2 o más y coprocesador matemático, con un drive de 3 1\2". Como paso inicial el usuario debe tener creado su archivo LINP.DAT, con los datos del problema que desea resolver; de existir algún error en el archivo de datos, no haber asignado un valor a un elemento del vector b, c o de la matriz A, el programa escribe un mensaje y continua su ejecución, pero en caso de haber declarado un número negativo de restricciones o variables, no declarar en orden las restricciones del problema, tratar de declarar más valores de los que son, el programa sí suspende su ejecución hasta que el error sea corregido (volver a editar el archivo LINP.DAT). Todos los mensajes y la solución del problema, se escriben en el archivo LINP.RES.

Una vez creado el archivo LINP.DAT, libre de errores, el siguiente paso es la solución del problema. Esta se lleva a cabo con la instrucción

```
a:> LINP
```

Cuando el programa termina

```
a:> _
```

con la instrucción

```
a:> type LINP.RES
```

se pueden ver los resultados del problema.

Por ejemplo, para el archivo LINP.DAT asociado al problema de la página 24, el programa da los siguientes resultados (archivo LINP.RES).

```
MOREPR =     1

M ( NO. DE RESTRICCIONES ) =     2
N ( NO. DE VARIABLES  (NO DE HOLGURA) ) =     3
NO. DE VARIABLES ACOTADAS =     1

DATOS DE ENTRADA PARA LOS ELEMENTOS DEL VECTOR C....
REGIS    2
( 1/0/   1.00000)( 2/0/   3.00000)( 3/0/  10.00000)( 0/0/   0.00000)
( 0/0/   0.00000)( 0/0/   0.00000)( 0/0/   0.00000)( 0/0/   0.00000)
REGIS    3
(99/9/999.00000)( 0/0/   0.00000)( 0/0/   0.00000)( 0/0/   0.00000)
( 0/0/   0.00000)( 0/0/   0.00000)( 0/0/   0.00000)( 0/0/   0.00000)

DATOS DE ENTRADA PARA LAS COTAS SUPERIORES....
REGIS    4
( 3/0/   2.00000)( 0/0/   0.00000)( 0/0/   0.00000)( 0/0/   0.00000)
( 0/0/   0.00000)( 0/0/   0.00000)( 0/0/   0.00000)( 0/0/   0.00000)
REGIS    5
(99/9/999.00000)( 0/0/   0.00000)( 0/0/   0.00000)( 0/0/   0.00000)
( 0/0/   0.00000)( 0/0/   0.00000)( 0/0/   0.00000)( 0/0/   0.00000)

DATOS DE ENTRADA PARA LOS ELEMENTOS DEL VECTOR B....
REGIS    6
( 1/1/120.00000)( 2/1/ 95.00000)( 0/0/   0.00000)( 0/0/   0.00000)
( 0/0/   0.00000)( 0/0/   0.00000)( 0/0/   0.00000)( 0/0/   0.00000)
REGIS    7
(99/9/999.00000)( 0/0/   0.00000)( 0/0/   0.00000)( 0/0/   0.00000)
( 0/0/   0.00000)( 0/0/   0.00000)( 0/0/   0.00000)( 0/0/   0.00000)

DATOS DE ENTRADA PARA LOS ELEMENTOS DE LA MATRIZ....
REGIS    8
( 0/0/   1.00000)( 1/0/ 12.00000)( 2/0/   5.00000)( 3/0/ 30.00000)
( 0/0/   0.00000)( 0/0/   0.00000)( 0/0/   0.00000)( 0/0/   0.00000)

REGIS    9
( 0/0/   2.00000)( 1/0/   2.00000)( 2/0/  10.00000)( 3/0/ 30.00000)
( 0/0/   0.00000)( 0/0/   0.00000)( 0/0/   0.00000)( 0/0/   0.00000)
```

```
REGIS    10
(99/9/999.00000)( 0/0/   0.00000)( 0/0/    0.00000)( 0/0/   0.00000)
( 0/0/   0.00000)( 0/0/   0.00000)( 0/0/    0.00000)( 0/0/   0.00000)


 OPTIMO
OBJETIVO              32.04545455

                 J......1         2         3

          VECTOR DE COTAS SUPERIORES....
                   -1.0000   -1.0000    2.0000

            VECTOR X.....
                    3.8636    2.7273    2.0000
I   VECTOR Y                                                  B      B-AX
                  ------------------------------------
                  (                                  )
1    0.0364   (    12.0000    5.0000   30.0000  ) LE  120.00  0.0000
                  (                                  )
2    0.2818   (     2.0000   10.0000   30.0000  ) LE   95.00  0.0000
                  (                                  )
                  ------------------------------------

            VECTOR C.....
                    1.0000    3.0000   10.0000

              YA - C.....
                    0.0000    0.0000   -0.4545

           COLUMNA          1         2
                  ------------------------
            YBASIS          2         1
                YR     0.2818    0.0364

RENGLON  XBS       XR              MATRIZ INVERSA

1     2       2.7273          0.1091        -0.0182
2     1       3.8636         -0.0455         0.0909

INBASE
  2  1  -1

    4 ITERACIONES DEL SIMPLEX.
```

MOREPR = 1 significa que la inversa y el archivo de datos serán impresos. Para las variables que no están sujetas a una cota superior, el programa les define una cota de -1.0. El vector Y

contiene las variables duales. LE representa una restricción de menor o igual e INBASE indica las variables que se encuentran en la base (un -1 significa que se trata de una variable acotada). El vector XBS contiene los índices de las variables que se encuentran en la base y XR contiene los valores de estas variables. El vector YBASIS indica el número de restricción que está presente en la inversa y YR contiene el valor de las variables asociadas con estas restricciones (variables duales). El problema tiene solución óptima y se resolvió en cuatro iteraciones.

Siempre que el número de variables sea menor o igual que ocho, el archivo de resultados tendrá esta forma; de no ser así, el archivo presentará los resultados en forma condensada. El mismo archivo LINP.RES, en forma condensada es:

```
MOREPR =      1

M ( NO. DE RESTRICCIONES ) =     2
N ( NO. DE VARIABLES  (NO DE HOLGURA) ) =     3
NO. DE VARIABLES ACOTADAS =     1


OPTIMO


ELEMENTOS DIFERENTES DE CERO DE LA MATRIZ A, ASOCIADOS CON SU
COLUMNA....
      1         2         3         4         5         6
 12.000     5.000    30.000     2.000    10.000    30.000
      1         2         3         1         2         3

LOS SIGUIENTES VECTORES MUESTRAN LOS PUNTOS INICIALES DE LOS
RENGLONES DE A, SIN ELEMENTOS IGUALES A CERO.
      1     2
      1     4

OBJETIVO     32.04545455

                   J         1         2         3
              VECTOR C       1.0       3.0      10.0
              VECTOR X    3.8636    2.7273    2.0000
                  YA-C      0.00      0.00     -0.45

EL VECTOR SIGN(I) INDICA EL SIGNO DE LA I-ESIMA RESTRICCION; 0 PARA
IGUAL; 1 PARA MENOR O IGUAL; -1 PARA MAYOR O IGUAL.

                   I         1         2
              VECTOR B     120.0      95.0
                  SIGN        1.        1.
```

```
                VECTOR Y        0.0364      0.2818
                   B-AX         0.0000      0.0000

          COLUMNA          1              2

           YBASIS          2              1
                YR         0.2818         0.0364

RENGLON   XBS        XR              MATRIZ INVERSA

1     2      2.7273          0.1091        -0.0182
2     1      3.8636         -0.0455         0.0909

INBASE
  2  1  -1

    4 ITERACIONES DEL SIMPLEX.
```

CAPITULO 2

PROGRAMA BB (BRANCH AND BOUND)

2.1 INTRODUCCION

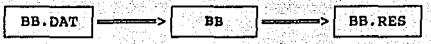
Cuando sólo algunas y no todas las variables de un problema lineal (PL), están restringidas a ser enteras, se le llama a éste "Problema Entero Mixto" (PEM). El programa BB fue elaborado para resolver PEM usando el método Branch and Bound, basado en el algoritmo de Land and Doig [14].

El código del programa, la descripción de variables y algunos problemas de prueba, se encuentran en Guillén I.,Ceciliano J.L., Programa BB, [11]. Con el fin de tener un ejecutable en PC, la dimensión del problema más grande que resuelve el programa es de 30 restricciones y 50 variables, de las cuales pueden ser todas enteras. Sólo problemas a maximizar se resuelven (min w = -max -w); no es necesario que éstos se encuentren en forma estándar, es decir, las restricciones pueden ser cualquier combinación de $\leq, \geq$ ó =.

El programa se puede usar directamente en problemas cuyas variables tengan un tamaño de paso (razón de incremento) más grande que uno; es decir, un tamaño de paso de 1000 significa que las variables pueden tomar valores de 0,1000,2000,3000,...etc.

Como el método Branch and Bound se basa en la resolución de problemas lineales (PL), BB utiliza al programa LINP, de tal forma que éste último se convierte en una subrutina de BB; así que todo lo relacionado con LINP es válido para este programa, excepto el procedimiento de escalamiento que no fue implantado en BB.

El programa está compuesto por 37 rutinas, incluyendo las de LINP, que constituyen 7906 líneas de código. Los datos del problema a resolver se leen de un archivo BB.DAT y los resultados que se obtienen se encuentran en un archivo BB.RES, tal como se muestra en la siguiente figura:



2.1 Diagrama de flujo de información.

2.2 CONCEPTOS BASICOS

Un problema entero mixto (PEM), es de la forma

$$\begin{aligned} &\text{Maximizar } z = && cx + dy \\ &\text{sujeto a} && Ax + Dy \leq b \\ & && x,\ y \geq 0 \\ &\text{y} && (x_j) \text{ enteras}, \quad j \in J \end{aligned}$$

El algoritmo Branch and Bound es útil para resolver el PEM. La idea de tal procedimiento es que la región convexa de soluciones factibles, del programa lineal, es particionada en subconjuntos convexos y una cota superior de la función objetivo es obtenida para cada subconjunto de la partición. Si se encuentra una solución que cumpla las restricciones enteras y cuyo valor en la función no sea menor que la cota superior en todos los subconjuntos de la partición, entonces ésta es la solución óptima.

Puede ser más fácil entender este método, si se hace uso de una gráfica con nodos y ramas (árbol); en donde un nodo corresponde a un punto x^l (vector **x** con l de sus componentes fijas a valores enteros positivos) y una rama une al nodo x^l *con* x^{l+1}. El nodo x^{l+1} se define del anterior, fijando una de sus variables libres a un entero positivo. Por ejemplo, en la fig. 2.2 se crean los nodos 8, 9, y 10, fijando x_1 (variable libre en el nodo 5), a 3, 4 y 2 respectivamente. El número de nivel se refiere al número de variables fijas a valores enteros; **z(l,k,t)** denota la solución óptima para el problema lineal en x^{l+1}, fijando una de sus variables libres x_k al entero positivo t, es decir, es el programa lineal $(PL)^l$, con la restricción adicional $x_k = t$.

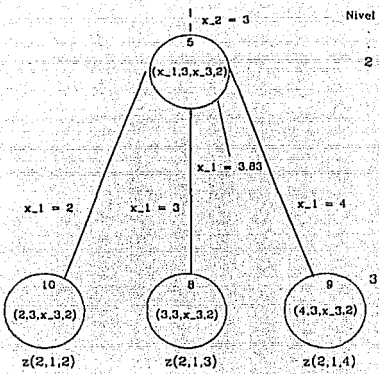


Fig. 2.2. Arbol Branch and Bound

En la figura 2.2. x_1 tiene un valor de 3.83, en la solución óptima del subproblema del nodo 5. Se fija x_1 a los dos enteros más cercanos, formándose los nodos 8 y 9. Se resuelve cada subproblema obteniéndose z(2,1,3) y z(2,1,4); el mayor de éstos, por decir z(2,1,3), es una cota superior para alguna solución del PEM que se encuentre a partir del nodo 5. Fijando x_1 en 2 se crea el nodo 10 y se determina z(2,1,2); si la actual mejor solución del PEM, z^*, es mayor o igual que z(2,1,3), entonces todos los nodos formados a partir del nodo 5 no son candidatos para mejorar el valor de la función objetivo del PEM, pero si $z^* <$ z(2,1,3), sólo el nodo 8 tendrá que ser examinado del nivel 3.

Así, sólo los nodos cuyo valor objetivo exceda de z^* son seleccionados para crear nuevos nodos. Cuando un problema de programación lineal en un nodo es no factible, todos los subproblemas en los nodos, ya sea izquierdo o derecho, son también

no factibles.

2.3 ALGORITMO

El procedimiento para encontrar la solución del problema entero mixto (PEM) [18], es resolver el problema como si fuese uno lineal (PL). Se examinan los valores de las variables que deben ser enteras, si todos son enteros factibles, la solución del PL es también la solución del PEM; de no ser así, se selecciona una variable x_k que no esté en un valor entero.

Con la solución del PL se obtienen cotas superiores en la función para todos los valores de x_k. Los valores enteros más próximos a su valor óptimo, tendrán las cotas más grandes en la función; supóngase que $x_k = \gamma$ da la cota más grande. La región factible puede ser particionada en los conjuntos

A. $x_k \leq \gamma-1$
B. $\gamma-1 < x_k < \gamma$
C. $x_k = \gamma$
D. $\gamma < x_k < \gamma+1$
E. $x_k \geq \gamma+1$.

Como no es posible que x_k tome un valor entero en los conjuntos B y D, éstos se descartan; así que sólo tres conjuntos se consideran: $x_k \leq \gamma-1$ (rama izquierda), $x_k = \gamma$ (rama principal) y $x_k \geq \gamma+1$ (rama derecha).

Se inicia la búsqueda de solución por la rama principal y la representación gráfica (árbol) se encuentra en un nivel 1. x_k tiene un valor de $\gamma-1$ en la rama izquierda y $\gamma+1$ en la rama derecha, con su respectivo valor de la función objetivo. Entonces se agrega la restricción $x_k = \gamma$ al problema, lo que da origen a un nuevo PL y la subrutina LP es llamada para resolverlo. La solución que se obtiene puede tener un valor de la función objetivo menor que el obtenido inicialmente, así que se actualizan las ramas izquierda y derecha. La situación es como al inicio, solo que ahora una variable está fija a un valor entero. Si la solución no es factible entera (todas las variables que deben ser enteras, están en un valor entero), se selecciona otra variable para fijarla a un valor entero (ramificar) y el nivel del árbol es incrementado.

Dentro del algoritmo se emplea el concepto de cola del árbol. Una cola puede ser:

1. Una solución entera.
2. Una solución del PL con valor objetivo mayor o

igual que el mejor objetivo encontrado hasta ese momento.

3. Un PL no factible.

Cuando se encuentra una cola, las ramas izquierda y derecha son examinadas, empezando por el nivel en el que se encontró ésta. Una solución no factible o con valor de la función objetivo menor que el de una solución factible entera encontrada, se borra del árbol. Pero si en un nivel, una de sus ramas tiene la posibilidad de mejorar el valor de la función objetivo, tal rama se convierte ahora en una rama principal y el nivel del árbol se incrementa.

El algoritmo se resume en tres pasos:

PASO 1 Se resuelve el PEM como uno de programación lineal

PASO 2 Se determina si la solución actual es una cola del árbol

Si la solución no es una cola, se elige una variable para fijarla a un valor entero, se incrementa el nivel del árbol y se va al paso 1.

PASO 3 Se revisan las ramas izquierda y derecha, a través de los niveles del árbol, hasta encontrar la solución con valor de la función objetivo mejor que el de la solución más favorable hasta ese momento, y ésta es ahora la rama principal. Ir al paso 1.

Si no existe tal solución entera, que permita mejorar el valor de la función objetivo, la ya encontrada es la óptima.

2.4 ESTRUCTURA DEL PROGRAMA

La estructura del programa BB se ilustra en la figura 2.3; cada cuadro representa una rutina, y se especifica la función y objetivo de éstas.

La descripción de las rutinas LP, IPRINT, IEXIT, DATA y la función A, se encuentra en el capítulo 1, sección 5.

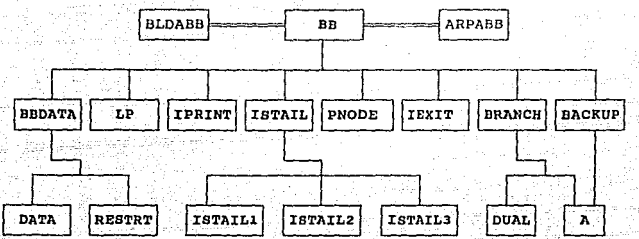


Fig. 2.3 Diagrama de estructura.

BB	Programa principal.
ARPABB	Archivo en donde se definen los tamaños de los parámetros que utiliza el programa; éstos pueden ser cambiados para resolver problemas más grandes.
BLDABB	Archivo en el que se definen las dimensiones de los arreglos que son comunes en el programa.

RUTINA	**OBJETIVO**
BBDATA	Leer los datos del problema del archivo BB.DAT e inicializar las constantes que se utilizan en el problema.
ISTAIL	Examinar la solución actual y verificar si ésta es una cola del árbol.

PNODE	Guardar la parte del árbol, rama y cota, que no fue explorada y dar una representación del nodo para el cual los cálculos se realizan.
BRANCH	Desarrollar una partición de la región factible (ramificación), fijando una variable a un valor discreto.
BACKUP	Regresar por las ramas izquierda y derecha hasta que se identifica la rama que es cola tipo 2 (ver pag. 33), del árbol.
RESTRT	Recuperar la parte del árbol, rama y cota, que es explorada.
ISTAIL1	Verificar si las variables básicas satisfacen las restricciones discretas.
ISTAIL2	Verificar si la solución encontrada es una cola del árbol, del tipo 2.
ISTAIL3	Indicar si la solución actual es no factible. Si se trata de la primera solución (programa lineal), el programa entero es no factible.
DUAL	Encontrar las variables a entrar a la base, incrementando o disminuyendo el valor de la j-ésima variable, mientras se mantiene la optimalidad.

2.5 FORMATO PARA DATOS DE ENTRADA (ARCHIVO BB.DAT)

En esta sección se describe la forma de proporcionar los datos del problema que se quiera resolver, al programa. Los registros son los siguientes

REGISTRO 1

Columna		
1-10	M	Número de restricciones.
11-20	N	Número total de variables.
21-30	NUMD	Número de variables discretas, si NUMD = -1, todas las variables tendrán un paso en JDISC(J) de 1. Si NUMD = 0 ó -1, los registros 2 y 3 no serán necesarios.
31-40	INTOBJ	= 1, el valor de la función objetivo, en la solución entera óptima, será entero. = 0, será real.
41-50	IPRBB	=1, la rutina IPRINT es llamada al inicio de la primer solución óptima del PL. En cada cola del árbol se imprime un mensaje con el número de iteraciones y las variables con sus valores correspondientes.
51-60	ITRBBM	Número máximo de iteraciones para BB. Si ITRBBM = 0, el número máximo de iteraciones será igual a N* ITRMAX (número máximo de iteraciones del PL).
61-70	IRBBM	Número máximo de reinversiones para BB.

71-80	IREST	= 0 la corrida actual no es un rearranque. = 1 los cálculos serán restaurados.

REGISTRO 2

Columna		
1-3	J	Número de una variable discreta.
5-10	JDISC(J)	Tamaño de paso de la j-ésima variable discreta.
11-13	J	
15-20	JDISC(J)	
	.	
	.	
	.	
71-73	J	
75-80	JDISC(J)	

El registro 2 se repite hasta cubrir todos los tamaños de paso para las NUMD variables discretas, ocho por cada registro.

REGISTRO 3

Columna		
1-10	999999999	Indica el fin de los tamaños de paso.

Los registros 4,5,...,12 son los mismos registros (1-9), del problema lineal descrito en el capítulo 1, sección 1.6.

Por ejemplo, el problema

$$\begin{aligned} \text{Max} \quad & z = 2x_1 + 3x_2 + 10x_3 \\ \text{sujeto a} \quad & 24x_1 + 5x_2 + 30x_3 \leq 1200 \\ & 4x_1 + 10x_2 + 30x_3 \leq 950 \\ & x_1 \geq 0,\ x_2 \geq 0,\ 0 \leq x_3 \leq 20 \\ & x_1, x_2, x_3 \in Z^+ \end{aligned}$$

donde la variable x_1 se incrementa en 5 unidades y las variables x_2, x_3 se incrementan en 10, sin cotas superiores en las variables y con valor real en la función objetivo. Debe ser expresado como

```
Columna 10        20        30        40        50        60..80

          2         3         3
  1       5  2     10  3     10
9999999999
          2         3         1         3
  1       2  2      3  3     10
9999999999
  3      20
9999999999
 11    1200  21   950
9999999999
          1  1     24  2      5  3     30
          2  1      4  2     10  3     30
9999999999
```

2.6 MANUAL DE USUARIO

El programa está diseñado para correr en una IBM-PC o compatible, con sistema operativo MS-DOS 3.2 o más y coprocesador matemático, y con un drive de 3 1\2". El usuario debe crear, por medio de algún editor, su archivo BB.DAT con los datos del problema que desea resolver. Sólo problemas a maximizar se resuelven (min w = -max -w); de existir algún error en el archivo de datos (no haber asignado un valor a un elemento del vector b, c o de la matriz A), el

programa escribe un mensaje y continua su ejecución, pero en ciertos casos (haber declarado un número negativo de restricciones o variables, no declarar en orden las restricciones del problema, tratar de declarar más valores de los que son, etc.) el programa sí suspende su ejecución, por lo que se debe corregir el archivo BB.DAT. Los mensajes finales y la solución del problema, se escriben en el archivo BB.RES.

Cuando el archivo BB.DAT no tiene errores, el siguiente paso es resolver el problema, lo cual se lleva a cabo con la instrucción

```
a:> BB
```

Una vez que el programa termina su ejecución

```
a:> _
```

la instrucción

```
a:> type BB.RES
```

muestra los resultados del problema.

El problema de la sección 2.5 pag. 39, con su respectivo archivo BB.DAT, puede servir de ejemplo para mostrar cómo se dan los resultados (archivo BB.RES) de un problema que es resuelto por BB. El archivo BB.RES es el siguiente,

```
M  (NO. DE RESTRICCIONES) =      2

N  (NO. DE VARIABLES) =      3
     3 DE LAS CUALES DEBEN SER ENTERAS.

MOREPR =      3

M ( NO. DE RESTRICCIONES ) =     2
N ( NO. DE VARIABLES  (NO DE HOLGURA) ) =     3
NO. DE VARIABLES ACOTADAS =     1


EL VALOR OPTIMO DEL PL ES        320.45454555 ALCANZADO EN CUATRO
ITERACIONES
```

```
LOS VALORES DE LA SOLUCION SON...
    19.3181818      27.2727273      20.0000000


LA SOLUCION NO.    1 SATISFACE LAS RESTRICCIONES DISCRETAS, CON
VALOR EN LA FUNCION DE      310.00000 EN     7 ITERACIONES
LOS VALORES DE LAS VARIABLES SON....
 0.1000000E+02  0.3000000E+02  0.2000000E+02


TODAS LAS RAMAS DEL ARBOL HAN SIDO EXPLORADAS EN   10 ITERACIONES
Y LA SOLUCION OPTIMA DISCRETA ES LA NO.  1
```

MOREPR=3 significa que ni la inversa, ni el archivo de datos son impresos. Para las variables que no están sujetas a una cota superior, el programa les define una cota de -1.0. En este ejemplo la solución se encontró hasta la iteración 10, lo cual significa que el árbol se encontraba en un nivel 3 y dos variables tuvieron que ser fijadas a un valor entero (x_1 y x_2).

CAPITULO 3

PROGRAMA PLP
(PROGRAMACION LINEAL PARAMETRICA)

3.1 INTRODUCCION

Cuando un problema de programación lineal (PL) se formula y resuelve por un proceso numérico, tal como el método Simplex, los valores de la matriz A, el vector B y C deben ser especificados. La condición de que tales coeficientes pueden tomar sólo un valor, podría representar de una manera no muy real la situación de la cual es modelo el PL, ya que en muchas ocasiones estos valores son estimados o se encuentran dentro de un rango de valores.

El programa PLP está diseñado para mostrar los cambios en la solución óptima de un problema lineal, que resultan por variar un elemento b_i del vector B (lado derecho de las restricciones) o por variar un elemento c_j del vector C (coeficientes de costo reducido), dentro de un rango de valores. El desarrollo matemático de este procedimiento se encuentra en Land A.,Powell S., Fortran Codes for Mathematical Programming, [14]. La descripción de variables y parámetros que se utilizan en el programa, así como algunos problemas de prueba y el código fuente, se encuentran en Ceciliano J.L.,Guillén I., Programa PLP, [3].

Como en PLP se debe resolver un problema lineal, el programa LINP se convierte en una subrutina de PLP. Por lo que todo lo referente al capítulo 1 (descripción de LINP), se incluye en este capítulo, excepto el procedimiento de escalamiento, el cual no fue incorporado en este programa. La dimensión del problema más grande en el que se realizan las variaciones es de 30 restricciones y 50 variables (sólo para efectos del diskette).

PLP obtiene los datos del problema a resolver, así como las variaciones paramétricas que se realizarán en éste, de un archivo PLP.DAT (sección 3.5), y los resultados obtenidos se escriben en el archivo PLP.RES, tal como se muestra en la figura 3.1.

Figura 3.1. Diagrama de flujo de información.

3.2 CONCEPTOS BASICOS

Muchos problemas de programación lineal (PL), están relacionados con los niveles de actividad en el futuro cuando todavía no es conocida la cantidad de recursos disponibles. Frecuentemente esta cantidad de recursos se encuentra dentro de un rango de valores, o el beneficio esperado de un nuevo producto se puede encontrar en un rango tal.

Es posible, con programación paramétrica, evaluar el efecto en la solución ya obtenida, al cambiar los valores de los elementos de B o C. Esto puede involucrar una serie de cambios de base, conforme el valor del parámetro se incrementa o decrementa.

En PLP, un análisis de sensibilidad en un elemento b_i, evalúa el rango de valores de b_i, dentro del cual la base óptima del PL permanece factible. El caso de un elemento c_j es análogo, sólo que en este caso se pretende que la base actual permanezca óptima.

Existen tres opciones en las cuales el valor del parámetro se varía dentro de unos límites específicos, éstas son:

1. El valor de un elemento de B (C) se puede variar entre límites específicos. Se evalúa el rango de valores dentro del cual, la base inicial del problema lineal es factible y óptima, para incrementar el valor del parámetro. Cuando el límite superior es alcanzado, el valor del parámetro se decrementa al límite inferior.

2. Un análisis de sensibilidad se puede realizar para un parámetro específico. En este caso el programa evalúa el rango de valores de b_i (c_j) en el cual la base inicial del PL es factible y óptima.

3. Un análisis de rango se puede realizar para todos los

elementos de B (C). En este caso se evalúa el rango de valores de cada elemento de B (C), dentro del cual la base inicial del PL es factible y óptima. El rango de cada elemento se evalúa independientemente.

3.3 ALGORITMO

En el algoritmo sólo es posible variar el valor de un parámetro, b_i ó c_j, pero variaciones sucesivas se pueden realizar.

PASO 1 Se resuelve el problema lineal (PL).

PASO 2 Si se realiza un análisis de rango en todos los elementos del vector B ir al paso 3. Si el análisis de rango es en los elementos del vector C ir al paso 4. De lo contrario, ir al paso 5.

PASO 3 Se determina el rango para cada elemento de B, en el cual la solución actual del PL es factible. Si no se realiza un análisis de rango en los elementos del vector C, ir al paso 5.

PASO 4 Se determina el rango para cada elemento de C, en el cual la solución actual del PL es óptima.

PASO 5 Si no existen más variaciones paramétricas ir al paso 8. Si existe una variación en un elemento de B ir al paso 6. De lo contrario, si la variación es un elemento del C, ir al paso 7.

PASO 6 Dependiendo de los límites declarados para b_i (sección 3.5), se realiza un análisis de sensibilidad o un análisis de rango en este elemento. Si la solución actual es óptima ir al paso 5, de lo contrario ir al paso 8.

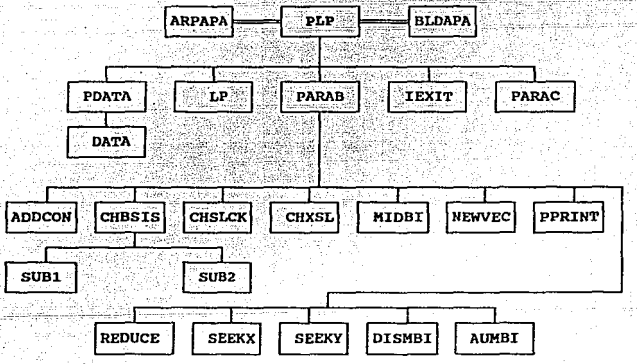
PASO 7 Dependiendo de los límites declarados para c_j (sección 3.5), se realiza un análisis de sensibilidad o un análisis de rango en este elemento. Si la solución actual es óptima ir al paso 5, de lo contrario ir al paso 8.

PASO 8 Fin.

3.4 ESTRUCTURA DEL PROGRAMA

El programa PLP tiene una estructura en su ejecución, la cual se muestra con las siguientes figuras, además se da una descripción de la función y objetivo de éstas. Cada cuadro representa una rutina en el programa.

i)



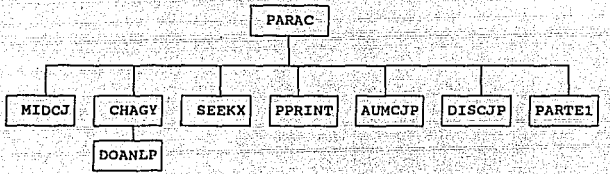
PLP Programa principal.

ARPAPA Archivo que contiene el tamaño de los parámetros que utiliza el programa, éstos se pueden cambiar para resolver problemas con un mayor o menor número de restricciones y variables.

BLDAPA Contiene la declaración de variables dimensionadas que son comunes en el programa.

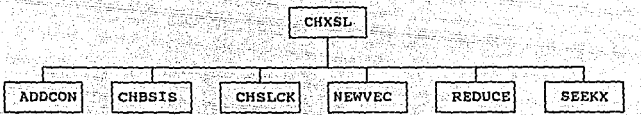
SUBRUTINA	OBJETIVO
PDATA	Leer del archivo PLP.DAT los datos del problema, para realizar las variaciones en los elementos del vector B (C).
PARAB	Realizar los cambios de base que corresponden a las variaciones en un rango de valores, de un elemento del vector B.
PARAC	Realizar los cambios de base que resultan al variar en un rango de valores, un elemento del vector C.
CHXSL	Actualizar los valores de las variables básicas y de las variables de holgura, en un cambio del valor de B(IP).
MIDBI	Regresar a la base óptima factible asociada con el valor inicial de B(IP).
PPRINT	Escribir los resultados del análisis paramétrico.
DISMBI	Disminuir el valor de B(IP) hasta su límite inferior, realizando cambios de base para obtener factibilidad y óptimalidad.
AUMBI	Incrementar el valor de B(IP) (elemento del vector B parametrizado) hasta su límite superior, realizando cambios de base para obtener factibilidad y óptimalidad.

ii)

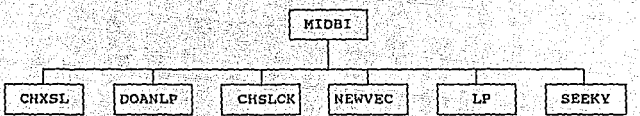


SUBRUTINA	OBJETIVO
MIDCJ	Regresar a la base óptima factible, asociada con el valor inicial de C(JP) (elemento del vector C parametrizado).
CHAGY	Actualizar los valores de las variables duales, cuando el valor de C(JP) es alterado.
DISCJP	Disminuir el valor de C(JP) hasta su límite inferior, revisando en cada paso si la solución actual es óptima.
AUMCJP	Incrementar el valor de C(JP) hasta su límite superior, revisando en cada paso si la solución actual es óptima.
PARTE1	Disminuir el valor de C(JP) hasta el límite inferior del rango, en el que la solución actual del PL es óptima.

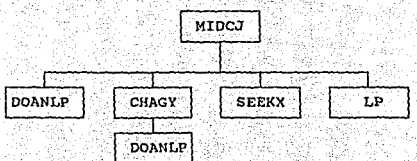
iii)



iv)



v)



Las descripciones que faltan corresponden a las subrutinas del programa LINP (ver capítulo 1, sección 1.6).

3.5 FORMATO PARA DATOS DE ENTRADA (ARCHIVO PLP.DAT)

En esta sección se describe al archivo de datos PLP.DAT, el cual contiene los datos del problema lineal, así como las variaciones paramétricas a realizar en los elementos del vector B (C).

Registro 1

Columna		
1-10	NUMP	Número de parametrizaciones, análisis de rango o de sensibilidad; es decir el número de registros del tipo 2. = 0 no se realizan variaciones paramétricas. El programa sólo resuelve el problema lineal.
11-20	IPRPLP	= 0 Las variables duales y el renglón de la función objetivo se escriben cuando se parametriza c_j. Las variables reales y de holgura se escriben cuando se parametriza b_i. = 1 Las variables duales, el renglón de la función objetivo, las variables reales y de holgura se escriben cuando se parametriza b_i o c_j.

El Registro 2 puede ser de tres tipos.

Registro 2

1. Parametrizar un elemento de B(C) entre límites específicos.

Columna		
1	TIPO	Carácter B si un elemento de B es variado o C si un elemento de C se varía.
8-10	I o J	i-ésimo renglón de la matriz A, en donde se varía su b_i asociado o j-ésima columna de A, en donde se varía su c_j correspondiente.
11-20	TLOW	Límite inferior de la variación paramétrica.
21-30	THIGH	Límite superior de la variación paramétrica.

TLOW y THIGH deben cumplir: $TLOW \leq b_i \leq THIGH$ o $TLOW \leq c_j \leq THIGH$ dependiendo si b_i ó c_j varía.

2. Realizar un análisis de sensibilidad en un elemento de B(C).

Columna		
1	TIPO	B ó C.
8-10	I o J	i-ésimo renglón de la matriz A, en donde se varía su b_i asociado o j-ésima columna de A, en donde se varía su c_j correspondiente.
11-20	B(I) ó C(J)	valor de b_i o c_j, en el problema lineal.
21-30	B(I) ó C(J)	el mismo valor repetido.

3. Realizar un análisis de rango en todos los elementos del vector B(C).

Columna		
1	TIPO	B ó C.
2-10	- 1	todos los elementos de B ó C son analizados.

El Registro 2 se repite NUMP veces; los tres tipos de registro pueden ir en cualquier orden. Los siguientes registros (3 al 12) son exactamente igual al archivo LINP.DAT, del capítulo 1.

Por ejemplo, el problema

$$\begin{aligned} \text{Maximizar } z = \; & x_1 + 3x_2 + 10x_3 \\ \text{sujeto a} \quad & 12x_1 + 5x_2 + 30x_3 \leq 120 \\ & 2x_1 + 10x_2 + 30x_3 \leq 95 \\ & x_1, x_2 \geq 0, \; 0 \leq x_3 \leq 2 \end{aligned}$$

con rangos paramétricos

b_1: $50 \leq b_1 \leq 300$; valor inicial 120
b_2: $95 \leq b_2 \leq 150$; valor inicial 95
c_1: $-1 \leq c_1 \leq 1$; valor inicial 1
c_2: $1 \leq c_2 \leq 5$; valor inicial 3
c_3: $10 \leq c_3 \leq 20$; valor inicial 10

donde se quiere realizar un análisis paramétrico en cada elemento de B y C, dentro de los rangos definidos; un análisis de sensibilidad y un análisis de rango en todos los elementos de B y C, los tres tipos del registro 2, que suman un total de 11 variaciones paramétricas. Se describe tal problema, con sus variaciones, en el siguiente archivo PLP.DAT

```
Columna  10        20        30        40        50 . . . .80

         11         1
B         1        50       300
B         2        95       150
C         1        -1         1
C         2         1         5
C         3        10        20
C        -1
B        -1
B         2        95        95
B         1       120       120
C         1         1         1
C         3        10        10
          2         3         1
   1      1  2      3  3      10
99999999999
   3      2
99999999999
  11    120  21     95
99999999999
          1  1      12  2      5  3      30
          2  1       2  2     10  3      30
99999999999
```

Al inicio el archivo contiene el número de variaciones paramétricas (11) y la opción de impresión de resultados (1); después contiene las peticiones de análisis paramétrico, análisis de rango y análisis de sensibilidad, en los elementos del vector B y C (elementos 1 y 3); y al final se tienen los datos del problema lineal a resolver (ver capítulo 1, sección 1.6). Como IPRPLP = 1, entonces la solución de las variaciones se escribirá en cada paso.

3.6 MANUAL DE USUARIO

El programa está diseñado para correr en una IBM-PC o compatible, con sistema operativo MS-DOS 3.2 o mayor y coprocesador matemático, y con un drive de 3 1\2". El usuario debe crear, por medio de algún editor el archivo PLP.DAT, con los datos del problema que desea resolver. Sólo problemas a maximizar se resuelven (min w = -max -w); de existir algún error en el archivo de datos (no haber asignado un valor a un elemento del vector b, c o de la matriz A), el

programa escribe un mensaje de error y continua su ejecución, pero en ciertos casos (haber declarado un número negativo de restricciones o variables, no declarar en orden las restricciones del problema, tratar de declarar más valores de los que son, etc.) el programa suspende su ejecución hasta que el error sea corregido (editar el archivo PLP.DAT). Los mensajes finales y la solución del problema, se escriben en el archivo PLP.RES.

Si el archivo PLP.DAT no tiene errores, a continuación se resuelve el problema y se realizan las variaciones pedidas, con la instrucción

```
a:> PLP
```

Una vez que el programa termina su ejecución

```
a:> _
```

la instrucción

```
a:> type PLP.RES
```

muestra los resultados de las variaciones paramétricas en el problema.

El problema de la sección 3.5 pag 51, con su respectivo archivo PLP.DAT se toma de ejemplo para mostrar los resultados (archivo PLP.RES), de un problema que es resuelto por PLP. El archivo se muestra en seguida.

```
NUMERO DE PETICIONES PARAMETRICAS      11     VALOR DE IMPRESION   1

B         1          50.000          300.000
B         2          95.000          150.000
C         1          -1.000            1.000
C         2           1.000            5.000
C         3          10.000           20.000
C        -1           0.000            0.000
B        -1           0.000            0.000
B         2          95.000           95.000
B         1         120.000          120.000
C         1           1.000            1.000
C         3          10.000           10.000


EL VALOR   1000000.000 REPRESENTARA MAS INFINITO Y  -1000000.000
REPRESENTARA MENOS INFINITO.

M ( NO. DE RESTRICCIONES ) =   2
N ( NO. DE VARIABLES (NO DE HOLGURA) ) =   3
NO. DE VARIABLES ACOTADAS =   1


 OPTIMO
OBJETIVO              32.04545455

                J......1          2          3

        VECTOR DE COTAS SUPERIORES....
                -1.0000    -1.0000    2.0000

          VECTOR X ....
                 3.8636     2.7273    2.0000
I   VECTOR Y                                                B      B-AX
                ---------------------------------
                (                               )
1    0.0364     (   12.0000     5.0000   30.0000 ) LE  120.00   0.0000
                (                               )
2    0.2818     (    2.0000    10.0000   30.0000 ) LE   95.00   0.0000
                (                               )
                ---------------------------------

          VECTOR C ....
                 1.0000     3.0000   10.0000

            YA - C ....
                 0.0000     0.0000   -0.4545

INBASE
  2  1  -1

    4 ITERACIONES DEL SIMPLEX.
```

```
BASE INICIAL FACTIBLE ENTRE      77.500  Y      270.000
RANGO PARA EL LADO DERECHO DE LA RESTRICCION      1


BASE INICIAL FACTIBLE ENTRE      70.000  Y      180.000
RANGO PARA EL LADO DERECHO DE LA RESTRICCION      2


BASE OPTIMA INICIAL ENTRE      0.600  Y      1.333
RANGO PARA EL ELEMENTO    1 DE LA FUNCION OBJETIVO


BASE OPTIMA INICIAL ENTRE      0.417  Y      3.167
RANGO PARA EL ELEMENTO    2 DE LA FUNCION OBJETIVO


BASE OPTIMA INICIAL ENTRE      9.545  Y      1000000.000
RANGO PARA EL ELEMENTO    3 DE LA FUNCION OBJETIVO


VARIACION PARAMETRICA EN EL LADO DERECHO DE LA RESTRICCION    1


BASE INICIAL FACTIBLE ENTRE      77.500  Y      270.000

OBJETIVO        32.045 VALOR DEL LADO DERECHO    120.000
VALOR DE LAS VARIABLES REALES
      3.8636       2.7273       2.0000
VALOR DE LAS VARIABLES DE HOLGURA
      0.0000       0.0000
VALOR DE LAS VARIABLES DUALES
      0.0364       0.2818
VALOR DE LOS ELEMENTOS DE YA-C
      0.0000       0.0000      -0.4545

BASE FACTIBLE ENTRE      270.000  Y      570.000

OBJETIVO        37.500 VALOR DEL LADO DERECHO    270.000
VALOR DE LAS VARIABLES REALES
     17.5000       0.0000       2.0000
VALOR DE LAS VARIABLES DE HOLGURA
      0.0000       0.0000
VALOR DE LAS VARIABLES DUALES
      0.0333       0.3000
VALOR DE LOS ELEMENTOS DE YA-C
      0.0000       0.1667       0.0000

OBJETIVO        38.500 VALOR DEL LADO DERECHO    300.000
VALOR DE LAS VARIABLES REALES
     20.5000       0.0000       1.8000
```

```
VALOR DE LAS VARIABLES DE HOLGURA
     0.0000       0.0000
VALOR DE LAS VARIABLES DUALES
     0.0333       0.3000
VALOR DE LOS ELEMENTOS DE YA-C
     0.0000       0.1667       0.0000

BASE INICIAL FACTIBLE ENTRE       77.500  Y       270.000

BASE FACTIBLE ENTRE      47.500  Y       77.500

OBJETIVO          30.500 VALOR DEL LADO DERECHO       77.500
VALOR DE LAS VARIABLES REALES
     0.0000       3.5000       2.0000
VALOR DE LAS VARIABLES DE HOLGURA
     0.0000       0.0000
VALOR DE LAS VARIABLES DUALES
     0.0667       0.2667
VALOR DE LOS ELEMENTOS DE YA-C
     0.3333       0.0000       0.0000

OBJETIVO          28.667 VALOR DEL LADO DERECHO       50.000
VALOR DE LAS VARIABLES REALES
     0.0000       9.0000       0.1667
VALOR DE LAS VARIABLES DE HOLGURA
     0.0000       0.0000
VALOR DE LAS VARIABLES DUALES
     0.0667       0.2667
VALOR DE LOS ELEMENTOS DE YA-C
     0.3333       0.0000       0.0000

FIN DE LA VARIACION PARAMETRICA EN LA RESTRICCION     1


VARIACION PARAMETRICA EN EL LADO DERECHO DE LA RESTRICCION     2

BASE INICIAL FACTIBLE ENTRE       70.000  Y       180.000

OBJETIVO          32.045 VALOR DEL LADO DERECHO       95.000
VALOR DE LAS VARIABLES REALES
     3.8636       2.7273       2.0000
VALOR DE LAS VARIABLES DE HOLGURA
     0.0000       0.0000
VALOR DE LAS VARIABLES DUALES
     0.0364       0.2818
VALOR DE LOS ELEMENTOS DE YA-C
     0.0000       0.0000      -0.4545

OBJETIVO          47.545 VALOR DEL LADO DERECHO      150.000
VALOR DE LAS VARIABLES REALES
     1.3636       8.7273       2.0000
VALOR DE LAS VARIABLES DE HOLGURA
```

```
      0.0000       0.0000
VALOR DE LAS VARIABLES DUALES
      0.0364       0.2818
VALOR DE LOS ELEMENTOS DE YA-C
      0.0000       0.0000      -0.4545

FIN DE LA VARIACION PARAMETRICA EN LA RESTRICCION    2


ANALISIS PARAMETRICO EN EL ELEMENTO DE LA FUNCION OBJETIVO ASOCIADO
CON LA VARIABLE     1

BASE OPTIMA INICIAL ENTRE        0.600  Y  1.333

OBJETIVO    32.045 VALOR DEL ELEMENTO EN LA FUNCION OBJETIVO  1.000
VALOR DE LAS VARIABLES DUALES
      0.0364       0.2818
VALOR DE LOS ELEMENTOS DE YA-C
      0.0000       0.0000      -0.4545
VALOR DE LAS VARIABLES REALES
      3.8636       2.7273       2.0000
VALOR DE LAS VARIABLES DE HOLGURA
      0.0000       0.0000

FIN DEL ANALISIS PARAMETRICO EN LA VARIABLE    1


ANALISIS PARAMETRICO EN EL ELEMENTO DE LA FUNCION OBJETIVO ASOCIADO
CON LA VARIABLE     2

BASE OPTIMA ENTRE        1.333  Y  7.200

OBJETIVO    33.333 VALOR DEL ELEMENTO EN LA FUNCION OBJETIVO  1.333
VALOR DE LAS VARIABLES DUALES
      0.0667       0.2667
VALOR DE LOS ELEMENTOS DE YA-C
      0.0000       0.0000       0.0000
VALOR DE LAS VARIABLES REALES
      6.5909       8.1818       0.0000
VALOR DE LAS VARIABLES DE HOLGURA
      0.0000       0.0000

FIN DEL ANALISIS PARAMETRICO EN LA VARIABLE    2


ANALISIS PARAMETRICO EN EL ELEMENTO DE LA FUNCION OBJETIVO ASOCIADO
CON LA VARIABLE     3

BASE OPTIMA INICIAL ENTRE        7.200  Y  10.000

OBJETIVO    72.000 VALOR DEL ELEMENTO EN LA FUNCION OBJETIVO  7.200
VALOR DE LAS VARIABLES DUALES
```

```
      0.6000       0.0000
VALOR DE LOS ELEMENTOS DE YA-C
      0.0000       0.0000       8.0000
VALOR DE LAS VARIABLES REALES
     10.0000       0.0000       0.0000
VALOR DE LAS VARIABLES DE HOLGURA
      0.0000      75.0000

OBJETIVO   100.000 VALOR DEL ELEMENTO EN LA FUNCION OBJETIVO 10.000
VALOR DE LAS VARIABLES DUALES
      0.8333       0.0000
VALOR DE LOS ELEMENTOS DE YA-C
      0.0000       0.0000       8.0000
VALOR DE LAS VARIABLES REALES
     10.0000       0.0000       0.0000
VALOR DE LAS VARIABLES DE HOLGURA
      0.0000      75.0000

FIN DEL ANALISIS PARAMETRICO EN LA VARIABLE   3


ANALISIS DE RANGO EN EL ELEMENTO DE LA FUNCION OBJETIVO ASOCIADO
CON LA VARIABLE    1

BASE OPTIMA INICIAL ENTRE      0.600  Y        1.333
OBJETIVO    30.500 VALOR DEL ELEMENTO EN LA FUNCION OBJETIVO  0.600
VALOR DE LAS VARIABLES DUALES
    0.0000       0.3000
VALOR DE LOS ELEMENTOS DE YA-C
    0.0000       0.0000      -1.0000
VALOR DE LAS VARIABLES REALES
    0.0000       3.5000       2.0000
VALOR DE LAS VARIABLES DE HOLGURA
   42.5000       0.0000

FIN DEL ANALISIS DE RANGO EN LA VARIABLE   1


ANALISIS DE RANGO EN EL ELEMENTO DE LA FUNCION OBJETIVO ASOCIADO
CON LA VARIABLE    2

BASE OPTIMA INICIAL ENTRE      0.417  Y        3.167

OBJETIVO    26.591 VALOR DEL ELEMENTO EN LA FUNCION OBJETIVO  1.000
VALOR DE LAS VARIABLES DUALES
    0.0727       0.0636
VALOR DE LOS ELEMENTOS DE YA-C
    0.0000       0.0000      -5.9091
VALOR DE LAS VARIABLES REALES
    3.8636       2.7273       2.0000
VALOR DE LAS VARIABLES DE HOLGURA
    0.0000       0.0000
```

```
OBJETIVO    32.045 VALOR DEL ELEMENTO EN LA FUNCION OBJETIVO  3.000
VALOR DE LAS VARIABLES DUALES
    0.0364       0.2818
VALOR DE LOS ELEMENTOS DE YA-C
    0.0000       0.0000      -0.4545
VALOR DE LAS VARIABLES REALES
    3.8636       2.7273       2.0000
VALOR DE LAS VARIABLES DE HOLGURA
    0.0000       0.0000

BASE OPTIMA ENTRE      3.167  Y        5.000
OBJETIVO    32.500 VALOR DEL ELEMENTO EN LA FUNCION OBJETIVO  3.167
VALOR DE LAS VARIABLES DUALES
    0.0333       0.3000
VALOR DE LOS ELEMENTOS DE YA-C
    0.0000       0.0000       0.0000
VALOR DE LAS VARIABLES REALES
    6.5909       8.1818       0.0000
VALOR DE LAS VARIABLES DE HOLGURA
    0.0000       0.0000

OBJETIVO    47.500 VALOR DEL ELEMENTO EN LA FUNCION OBJETIVO  5.000
VALOR DE LAS VARIABLES DUALES
    0.0000       0.5000
VALOR DE LOS ELEMENTOS DE YA-C
    0.0000       0.0000       5.0000
VALOR DE LAS VARIABLES REALES
    0.0000       9.5000       0.0000
VALOR DE LAS VARIABLES DE HOLGURA
   72.5000       0.0000

FIN DEL ANALISIS DE RANGO EN LA VARIABLE    2


ANALISIS DE RANGO EN EL ELEMENTO DE LA FUNCION OBJETIVO ASOCIADO
CON LA VARIABLE     3

BASE OPTIMA INICIAL ENTRE      9.545  Y       10.000
OBJETIVO      32.045 VALOR DEL ELEMENTO EN LA FUNCION OBJETIVO
10.000
VALOR DE LAS VARIABLES DUALES
    0.0364       0.2818
VALOR DE LOS ELEMENTOS DE YA-C
    0.0000       0.0000      -0.4545
VALOR DE LAS VARIABLES REALES
    3.8636       2.7273       2.0000
VALOR DE LAS VARIABLES DE HOLGURA
    0.0000       0.0000

FIN DEL ANALISIS DE RANGO EN LA VARIABLE    3
```

```
ANALISIS DE SENSIBILIDAD EN EL LADO DERECHO DE LA RESTRICCION    2

BASE INICIAL FACTIBLE ENTRE     70.000  Y       180.000

OBJETIVO      32.045  VALOR DEL LADO DERECHO       95.000
VALOR DE LAS VARIABLES REALES
      3.8636      2.7273      2.0000
VALOR DE LAS VARIABLES DE HOLGURA
      0.0000      0.0000
VALOR DE LAS VARIABLES DUALES
      0.0364      0.2818
VALOR DE LOS ELEMENTOS DE YA-C
      0.0000      0.0000     -0.4545

OBJETIVO      56.000  VALOR DEL LADO DERECHO      180.000
VALOR DE LAS VARIABLES REALES
      0.0000     12.0000      2.0000
VALOR DE LAS VARIABLES DE HOLGURA
      0.0000      0.0000
VALOR DE LAS VARIABLES DUALES
      0.0364      0.2818
VALOR DE LOS ELEMENTOS DE YA-C
      0.0000      0.0000     -0.4545

OBJETIVO      25.000  VALOR DEL LADO DERECHO       70.000
VALOR DE LAS VARIABLES REALES
      5.0000      0.0000      2.0000
VALOR DE LAS VARIABLES DE HOLGURA
      0.0000      0.0000
VALOR DE LAS VARIABLES DUALES
      0.0364      0.2818
VALOR DE LOS ELEMENTOS DE YA-C
      0.0000      0.0000     -0.4545

FIN DEL ANALISIS DE SENSIBILIDAD EN LA RESTRICCION    2


ANALISIS DE SENSIBILIDAD EN EL LADO DERECHO DE LA RESTRICCION    1

BASE INICIAL FACTIBLE ENTRE     77.500  Y       270.000

OBJETIVO      32.045  VALOR DEL LADO DERECHO      120.000
VALOR DE LAS VARIABLES REALES
      3.8636      2.7273      2.0000
VALOR DE LAS VARIABLES DE HOLGURA
      0.0000      0.0000
VALOR DE LAS VARIABLES DUALES
      0.0364      0.2818
VALOR DE LOS ELEMENTOS DE YA-C
      0.0000      0.0000     -0.4545

OBJETIVO      37.500  VALOR DEL LADO DERECHO      270.000
```

```
VALOR DE LAS VARIABLES REALES
     17.5000      0.0000      2.0000
VALOR DE LAS VARIABLES DE HOLGURA
      0.0000      0.0000
VALOR DE LAS VARIABLES DUALES
      0.0364      0.2818
VALOR DE LOS ELEMENTOS DE YA-C
      0.0000      0.0000     -0.4545

OBJETIVO     30.500  VALOR DEL LADO DERECHO       77.500
VALOR DE LAS VARIABLES REALES
      0.0000      3.5000      2.0000
VALOR DE LAS VARIABLES DE HOLGURA
      0.0000      0.0000
VALOR DE LAS VARIABLES DUALES
      0.0364      0.2818
VALOR DE LOS ELEMENTOS DE YA-C
      0.0000      0.0000     -0.4545

FIN DEL ANALISIS DE SENSIBILIDAD EN LA RESTRICCION    2


ANALISIS DE SENSIBILIDAD EN LA VARIABLE   1

BASE OPTIMA INICIAL ENTRE      0.600  Y        1.333

OBJETIVO   32.045  VALOR DEL ELEMENTO EN LA FUNCION OBJETIVO  1.000
VALOR DE LAS VARIABLES DUALES
      0.0364      0.2818
VALOR DE LOS ELEMENTOS DE YA-C
      0.0000      0.0000      0.0000
VALOR DE LAS VARIABLES REALES
      3.8636      2.7273      2.0000
VALOR DE LAS VARIABLES DE HOLGURA
      0.0000      0.0000

BASE OPTIMA ENTRE      1.333  Y        7.200

OBJETIVO   33.333  VALOR DEL ELEMENTO EN LA FUNCION OBJETIVO  1.333
VALOR DE LAS VARIABLES DUALES
      0.0667      0.2667
VALOR DE LOS ELEMENTOS DE YA-C
      0.0000      0.0000      0.0000
VALOR DE LAS VARIABLES REALES
      6.5909      8.1818      0.0000
VALOR DE LAS VARIABLES DE HOLGURA
      0.0000      0.0000

BASE OPTIMA ENTRE      7.200  Y       10.000

OBJETIVO   72.000  VALOR DEL ELEMENTO EN LA FUNCION OBJETIVO  7.200
VALOR DE LAS VARIABLES DUALES
```

```
      0.0600       0.0000
VALOR DE LOS ELEMENTOS DE YA-C
      0.0000       0.0000       8.0000
VALOR DE LAS VARIABLES REALES
     10.0000       0.0000       0.0000
VALOR DE LAS VARIABLES DE HOLGURA
      0.0000      75.0000

OBJETIVO  100.000  VALOR DEL ELEMENTO EN LA FUNCION OBJETIVO 10.000
VALOR DE LAS VARIABLES DUALES
      0.8333       0.0000
VALOR DE LOS ELEMENTOS DE YA-C
      0.0000       1.1667      15.0000
VALOR DE LAS VARIABLES REALES
     10.0000       0.0000       0.0000
VALOR DE LAS VARIABLES DE HOLGURA
      0.0000      75.0000

FIN DEL ANALISIS DE SENSIBILIDAD EN LA VARIABLE   1


ANALISIS DE SENSIBILIDAD EN LA VARIABLE   3

BASE OPTIMA INICIAL ENTRE     9.545  Y        10.000

OBJETIVO   32.045 VALOR DEL ELEMENTO EN LA FUNCION OBJETIVO  10.000
VALOR DE LAS VARIABLES DUALES
      0.0364       0.2818
VALOR DE LOS ELEMENTOS DE YA-C
      0.0000       0.0000       0.0000
VALOR DE LAS VARIABLES REALES
      3.8636       2.7273       2.0000
VALOR DE LAS VARIABLES DE HOLGURA
      0.0000       0.0000

FIN DEL ANALISIS DE SENSIBILIDAD EN LA VARIABLE   3
```

CAPITULO 4

PROGRAMA QP (PROGRAMACION CUADRATICA)

4.1 INTRODUCCION

La primera incursión en la teoría de la programación no lineal fue el problema conocido como programación cuadrática. Este problema se restringe a maximizar una función cuadrática cóncava de variables sujetas a restricciones lineales; muchos métodos para resolver este caso se han publicado [6]. Matemáticamente ésta es la primera extensión del problema de programación lineal, se resuelve en un número finito de pasos.

El programa QP resuelve el problema de programación cuadrática mediante el método de E. M. L. Beale, A. H. Land y G. Morton [14]. La descripción de variables, parámetros, algunos problemas de prueba y el código fuente del programa, se encuentra en Ceciliano J.L.,Guillén R., Programa QP, [4]. Los problemas deben ser del tipo de maximizar (min w = -max -w), no es necesario que estén en forma estándar, por lo que las restricciones pueden ser cualquier combinación de $\leq, \geq$ ó $=$.

La dimensión del problema más grande que resuelve el programa, para efectos del diskette, es de 30 restricciones y 50 variables. Si alguna variable está acotada superiormente, el programa está diseñado para que tal situación no represente una restricción más en el problema. Toda la aritmética se realiza en doble-precisión, pero por razones de espacio en los formatos de salida, los resultados se escriben en formatos de 12 cifras con cuatro decimales como máximo (F12.4), para reales y cinco cifras para enteros (I5).

Al programa se le deben proporcionar los datos del problema que se quiere resolver, por medio de un archivo QP.DAT y los resultados obtenidos se dan en el archivo de salida QP.RES, como se muestra en figura 4.1.

Figura 4.1. Diagrama de flujo de información

4.2 CONCEPTOS BASICOS

El programa general cuadrático de minimización tiene la forma matricial

$$\begin{aligned} \text{Minimizar } w = \quad & pX + \frac{1}{2}xDx \\ \text{sujeto a} \quad & Ax - b \geq 0 \\ & x \geq 0 \end{aligned}$$

donde D es una matriz simétrica positiva semidefinida, lo cual hace de w una función convexa y garantiza que cualquier mínimo local en la región factible convexa sea un mínimo global.

El problema dual asociado al programa general tiene la forma:

$$\begin{aligned} \text{Maximizar } z = \quad & yb \\ \text{sujeto a} \quad & yA \leq p + Dx \\ & y \geq 0 \end{aligned}$$

Si $\overline{x}$ es un mínimo local, ya que $\nabla g(\overline{x}) = A$ y $\nabla f(\overline{x}) = p + D\overline{x}$, existe $\overline{y} \geq 0$ [14], tal que

$$\begin{aligned} & \overline{y}A \geq p + \overline{x}D \\ & \overline{y}(A\overline{x} - b) = 0 \\ & \overline{x}(p + D\overline{x} - \overline{y}A) = 0 \end{aligned}$$

4.2.1. Condiciones suficientes para un mínimo global de una función cuadrática convexa

Si **f(x)** es convexa en la región definida por las restricciones

$$\nabla f(\overline{x})(x^0 - \overline{x}) \leq f(x^0) - f(\overline{x})$$

haciendo $\overline{c} = \nabla f(\overline{x}) = p + D\overline{x}$, se tiene que

$$\overline{c}(x^0-\overline{x}) \leq f(x^0)-f(\overline{x})$$
$$0 \leq \overline{c}x^0-\overline{c}\,\overline{x} \leq f(x^0)-f(\overline{x})$$

Es decir, $f(\overline{x}) \leq f(x^0)$ para cualquier x^0 que cumpla las restricciones, y entonces un mínimo local es también el mínimo global cuando la función cuadrática es convexa.

4.2.2. Algoritmo de Beale para programación cuadrática

El algoritmo está diseñado para resolver problemas de la forma

$$\text{Maximizar } z = px + \frac{1}{2}xDx$$
$$\text{sujeto a } Ax \backsim b \quad (\textit{restricciones de igualdad y desigualdad})$$
$$x \geq 0$$

Tal problema se optimiza, al menos localmente, si el problema de programación lineal

$$\text{Maximizar } z = \overline{c}x$$
$$\text{sujeto a } Ax \backsim b$$
$$x \geq 0$$

se maximiza en el punto $\overline{x}$, donde $\overline{c} = p + \overline{x}D$.

Además, si la función es cóncava, dentro de un polítopo convexo, el óptimo local es también el óptimo global. El punto óptimo $\overline{x}$, de un problema cuadrático (QP), no necesita ser un punto extremo de la región factible, es decir, un punto básico (figura 4.2).

Con el fin de utilizar un algoritmo Simplex en este tipo de problema, se recurre a agregar restricciones auxiliares en él. De esta manera si la solución se encuentra en una cara k-dimensional de la región factible, k restricciones auxiliares se pueden encontrar que intersecten las otras n-k caras para formar un punto básico.

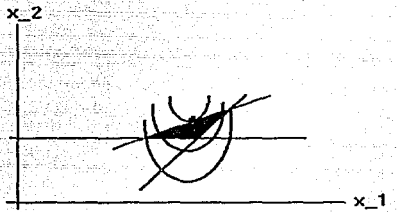


Figura 4.2.

Una restricción auxiliar se puede agregar a un problema lineal (LP), sin afectar el valor óptimo de la función, aunque sí podría producir diferentes puntos básicos. Por ejemplo, agregar una restricción auxiliar en la figura 4.3, produce una tercera solución óptima (figura 4.4).

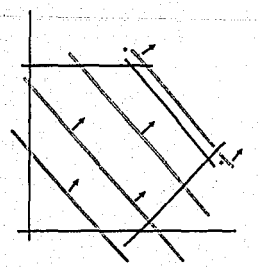


Figura 4.3.

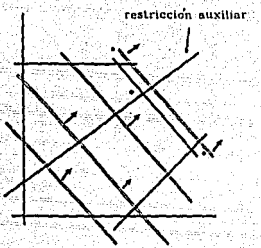


Figura 4.4.

Estas restricciones en realidad no restringen la región de soluciones factibles, ya que ellas son de la forma

$$a^i x \sim b$$

significando que $a^i x$ es libre para ser $\leq b_i$ o $\geq b_i$. La variable de holgura $x_{n+i} = b_i - a^i x$ es una variable libre, y la correspondiente variable dual, y_i, debe ser igual a cero.

Es útil resumir los tipos de restricciones que se usan en estos algoritmos, considerando que los coeficientes de las variables de holgura son en cada caso el vector unitario positivo.

i-ésima restricción	holgura (x_{n+i})	variable dual (y_i)
$a^i x \leq b_i$	$x_{n+i} \geq 0$	$y_i \geq 0$
$a^i x \geq b_i$	$x_{n+i} \leq 0$	$y_i \leq 0$
$a^i x = b_i$	$x_{n+i} = 0$	y_i s.r.s.
$a^i x \sim b_i$	x_{n+i} s.r.s.	$y_i = 0$

Las restricciones sugeridas por Beale, expresadas en términos de variables originales [14], son:

$$(g^R D^R - d^k)x \leq -g^R p^R + p_k$$

donde g^R es el vector $R^{-1}a^k$, columna actualizada de la variable x_k que entra a la base; D^R es el renglón de D asociado con las actuales variables básicas y d^k con la variable que entra a la base; p^R es el elemento de p asociado con las actuales variables básicas y p_k con la variable x_k que entrará a la base.

Tal restricción tiene valor dual cero cuando es impuesta, y ésta permanece en cero, siempre y cuando sólo restricciones adicionales determinen los sucesivos cambios de base. Su valor dual cambia de cero sólo si una restricción original, que no era efectiva cuando la restricción fue agregada, llega a ser efectiva.

Una vez que un punto factible se obtiene por el procedimiento Simplex, el algoritmo procede a calcular la función lineal $c^0 = \nabla f(x^0) = p + Dx^0$ en cada punto básico x^0; entonces se verifica la optimalidad de la función lineal c^0x, si no es correcto sigue la regla del Simplex moviéndose en la dirección indicada por la más grande violación de las condiciones de optimalidad lineal; es decir, incrementar x_k, donde

$$y^0a^k \leq c^0{}_k \qquad (y^0 = c^{0R}R^{-1})$$

En algunas circunstancias cuando la función objetivo se incrementa primero y después se decrementa a lo largo de una cara de la región factible, el movimiento es limitado no por la siguiente restricción que llega a ser efectiva, sino por la restricción auxiliar

$$ya^k \leq c_k \qquad (\textit{tomando } c, y \textit{ como funciones de } x)$$

lo cual es equivalente a

$$(g^RD^R - d^k)x \leq -g^kp^R + p_k$$

como antes. La nueva restricción puede ser efectiva en limitar el movimiento, pero después ésta se trata como

$$(g^RD^R - d^k)x \leftarrow -g^Rp^R + p_k$$

es decir, como tener dos variables de holgura.

Alguna restricción auxiliar que no llega a ser efectiva cuando es impuesta, se elimina inmediatamente. Durante el cálculo, el valor dual de una restricción auxiliar puede cambiar de cero, y la variable de holgura puede entrar a la base (positiva o negativa), la restricción se elimina en este caso también.

El algoritmo termina cuando una solución del PL es alcanzada para la función lineal, la cual es igual al vector gradiente del punto

actual, con el valor dual de alguna restricción auxiliar aun presente en la base, igual a cero.

4.3 ALGORITMO

Este algoritmo es similar al del programa LINP, excepto que la función lineal c es evaluada en cada cambio de base, una vez que se obtuvo factibilidad.

PASO 1 Se forma la base inicial, agregando una variable de holgura en la primera restricción.

PASO 2 Se actualizan los arreglos asociados con la base actual y se verifica la factibilidad en todas las restricciones. Si todas las restricciones son factibles ir al paso 4. De lo contrario, si la no factibilidad existe en una restricción no presente en la base, incorporar esta restricción a la inversa.

PASO 3 Determinar la variable que entre a la base para reducir la no factibilidad. Si tal variable existe ir al paso 5, de lo contrario el problema es no factible e ir al paso 11.

PASO 4 Se remueve de la matriz A, una restricción auxiliar que ya no es efectiva.

PASO 5 Se evalúa la función objetivo $OBJ = \frac{1}{2}(c+p)\bar{x}$ para $\bar{x}$ (valor actual de x) y $p = c+\bar{x}D$. Si mejora el valor de la función objetivo ir al paso 6, de lo contrario ir al paso 7.

PASO 6 Se determina si la solución actual es óptima. Si es óptima ir al paso 11.

PASO 7 Se calcula la columna asociada con la variable que entrará a la base. Si la solución no es un punto de la

región factible, ir al paso 9.

PASO 8 Se agrega una restricción auxiliar en el problema.

PASO 9 Se busca la variable que saldrá de la base. Si el problema es no acotado ir al paso 11. De lo contrario, si la variable que saldrá de la base está asociada con una restricción no presente en la base, agregar esta restricción a la base.

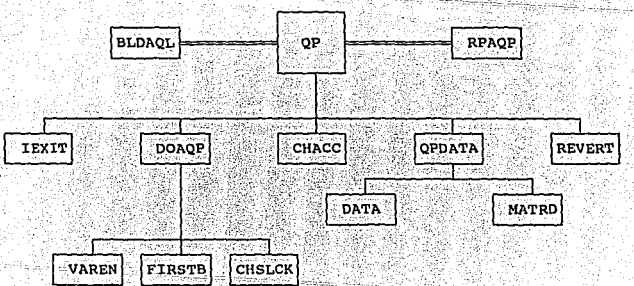
PASO 10 Se realiza el cambio de base, ir al paso 2.

PASO 11 Fin.

4.4 ESTRUCTURA DEL PROGRAMA

El diagrama de estructura es una representación gráfica de como está constituido el programa, cada cuadro representa una rutina y se describe la función y objetivo de éstas. El número de líneas de código del programa es de 6092.

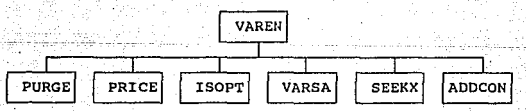
i)



QP	Programa principal
ARPAQP	Archivo que contiene los tamaños de los parámetros que se utilizan en el programa.
BLDAQP	Archivo que contiene la declaración de las variables dimensionadas, que son comunes en el programa.

SUBRUTINA	OBJETIVO
QPDATA	Leer del archivo QP.DAT, los datos del QP que se quiere resolver.
MATRD	Leer la matriz de coeficientes cuadráticos (D), en la función objetivo.
DOAQP	Buscar la solución del problema de programación cuadrática.
VAREN	Buscar la variable a entrar a la base.

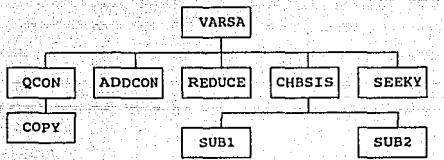
ii)



SUBRUTINA	OBJETIVO
PURGE	Remover de la matriz A y del vector B, todos los elementos asociados con una restricción auxiliar.
PRICE	Evaluar la función objetivo.

VARSA	Buscar la solución del QP.

iii)



SUBRUTINA	OBJETIVO
QCON	Agregar una restricción auxiliar, después de un cambio de base.

La descripción de las otras subrutinas se encuentra en el capítulo 1, sección 1.5.

4.5 FORMATO PARA DATOS DE ENTRADA (ARCHIVO QP.DAT)

En esta sección se describe la forma en que los datos del problema se proporcionan al programa; es decir, la creación del archivo QP.DAT, y se da un ejemplo de éste.

Registro 1

Columna		
1-10	M	Número de restricciones.
11-20	N	Número de variables.
21-30	NUMQ	Número de variables cuadráticas.

Registro 2

Columna		
1-5	ID	La variable asociada con el renglón de la matriz D.
6-10	JD	La variable asociada con la columna de la matriz D.
11-20	D(ID,JD)	Un elemento (ID,JD) no cero D. Por simetría D(ID,JD) y D(JD,ID) tienen el mismo valor y entonces no es necesario especificar ambos coeficientes.
21-25	ID	
26-30	JD	
31-40	D(ID,JD)	
.	.	
.	.	
.	.	
71-80	D(ID,JD)	

Registro 3

Columna		
1-10	999999999	Indica el fin de los elementos de D.

Los registros 4,...,13 son los Registros 1-10 del programa LINP (ver capítulo 1, sección 1.6). El vector p de la función objetivo cuadrática toma el lugar del vector c de LINP.

Por ejemplo, el problema

$$\text{Maximizar } z = px + \frac{1}{2}xDx$$
$$\text{sujeto a} \quad x_1 + x_2 + 2x_3 \leq 3$$
$$x_1, x_2, x_3 \geq 0$$

donde

$$p = [\ 8 \quad 6 \quad 4\ ]$$

$$D = \begin{bmatrix} -4 & -2 & -2 \\ -2 & -4 & 0 \\ -2 & 0 & -2 \end{bmatrix}$$

Dado que no existen variables acotadas superiormente; pensando en un número máximo de iteraciones heurístico y de reinversiones 0, el archivo de datos asociado al problema es:

```
Columna 10        20        30        40        50        60 ..80

        1         3         3
    1   1        -4     1   2        -2     1   3        -2
9999999999
        1         3         0         3
  1     8  2      6  3      4
9999999999
 11     3
9999999999
        1  1      1  2      1  3      2
9999999999
```

4.6 MANUAL DE USUARIO

El programa está diseñado para correr en una IBM-PC ó compatible, con sistema operativo MS-DOS 3.2 o mayor y coprocesador matemático, y con un drive de 3 1\2". El usuario debe crear, usando algún editor, el archivo QP.DAT con los datos del problema que desea resolver. Sólo problemas a maximizar se resuelven (min w = -max -w); de existir algún error en el archivo de datos (no haber asignado un valor a un elemento del vector b, c o de la matriz A), el programa escribe un mensaje de error y continua su ejecución, pero

en ciertos casos (haber declarado un número negativo de restricciones o variables, no declarar en orden las restricciones del problema, tratar de declarar más valores de los que son, etc.) el programa suspende su ejecución hasta que el error sea corregido (editar el archivo QP.DAT). Los mensajes finales y la solución del problema, se escriben en el archivo QP.RES.

Si el archivo QP.DAT no tiene errores, a continuación se resuelve el problema con la instrucción

```
a:> QP
```

Una vez que el programa termina su ejecución

```
a:> _
```

la instrucción

```
a:> type QP.RES
```

muestra los resultados del problema.

El problema de la sección 4.5 pag 74, con su archivo QP.DAT sirve de ejemplo para mostrar los resultados (archivo QP.RES), de un problema que es resuelto por QP. El archivo se muestra a continuación.

```
M =    1
N =    3
NUMERO DE VARIABLES CON COEFICIENTES NO-LINEALES =    3

MOREPR =    3
```

```
M ( NO. DE RESTRICCIONES ) =     1
N ( NO. DE VARIABLES (NO DE HOLGURA) ) =     3
NO. DE VARIABLES ACOTADAS =     0

VECTOR P . . . .
        1         2         3
     8.00      6.00      4.00

MATRIZ D, DE COEFICIENTES NO-LINEALES . . . .

          COLUMNA     1           2           3
ROW     VARIABLE      1           2           3
  1        1       -4.0000     -2.0000     -2.0000
  2        2       -2.0000     -4.0000      0.0000
  3        3       -2.0000      0.0000     -2.0000
```

EL VECTOR C CONTIENE LAS DERIVADAS PARCIALES DE LA FUNCION CUADRATICA.

PUEDEN HABER MAS RESTRICCIONES, DE LAS M DECLARADAS ORIGINALMENTE, PERO ESTAS TENDRAN UN VALOR DUAL CERO.

TODAS LAS VARIABLES NO ACOTADAS TENDRAN UNA COTA SUPERIOR DE 1000000.0000

```
 OPTIMO DEL PROGRAMA CUADRATICO
OBJETIVO              8.88888889

                J.....1         2         3

            VECTOR X ....
                    1.3333    0.7778    0.4444
I   VECTOR Y                                                    B      B-AX
            ------------------------------------
            (                                   )
1    0.2222 (    1.0000    1.0000    2.0000 ) LE    3.00  0.0000
            (                                   )
2    0.0000 (    3.0000    2.0000    1.0000 ) LE    6.00  0.0000
            (                                   )
3    0.0000 (   -0.8000    2.8000   -1.6000 ) LE    0.40  0.0000
            (                                   )
            ------------------------------------
            VECTOR C ....
                    0.2222    0.2222    0.4444

              YA - C ....
                    0.0000    0.0000    0.0000
```

```
INBASE
  2  3  1

   5 ITERACIONES DEL SIMPLEX.
```

MOREPR=3 significa que ni la inversa, ni el archivo de datos son impresos. Para las variables que no están sujetas a una cota superior, el programa les define una cota de -1.0. El vector P contiene los coeficientes lineales de la función objetivo, y el vector Y contiene las variables duales. LE representa una restricción de menor o igual e INBASE indica las variables que se encuentran en la base (un -1 significa que se trata de una variable acotada). El problema tiene solución óptima y se resolvió en cinco iteraciones. Dos restricciones adicionales (auxiliares) fueron agregadas al problema, durante su solución.

Siempre que el número de variables sea menor o igual que ocho, el archivo de resultados tendrá esta forma; de no ser así, el archivo presentará los resultados en forma condensada.

CAPITULO 5

PROGRAMA DDW
(METODO DE DESCOMPOSICION DANTZIG-WOLFE)

5.1 INTRODUCCION

Los problemas grandes de programación lineal suelen tener una estructura especial, la cual se busca aprovechar para desarrollar procedimientos computacionales eficientes que permitan resolverlos. La publicación del principio de descomposición de Dantzig-Wolfe en 1960 [15], inició el extenso trabajo en programación matemática de gran escala. El método es más eficiente cuando se aplica a problemas lineales, cuya matriz de coeficientes tiene estructura **bloque angular**, es decir, uno o más bloques (subproblemas), ligados por restricciones de enlace.

En este capítulo se describe el programa DDW, el cual tiene como propósito resolver problemas lineales con la estructura de bloque angular, utilizando el método de descomposición de Dantzig-Wolfe [16]. La descripción de las variables, parámetros y algunos problemas de prueba se encuentran en Cecilíano J.L.,Guillén I., Programa DDW, [5]. La decisión de implantarlo fue la oportunidad de contar con un programa capaz de resolver problemas de programación lineal (LINP), ya que este método involucra una continua resolución de subproblemas lineales. Es así que nueve rutinas se elaboraron para coordinar el programa LINP (capítulo 1), con la teoría de descomposición, y obtener DDW.

En esta etapa, DDW, sólo puede resolver problemas, cuyas restricciones de enlace tengan signo de menor o igual, y su correspondiente lado derecho sea positivo; el resto del problema (subproblemas), no está restringido a tales condiciones. Es importante señalar que los conjuntos de restricciones de los subproblemas, se suponen acotados. Los subproblemas se resuelven con LINP, utilizando el procedimiento de escalamiento; DDW les asigna una cota superior a las variables, y la dimensión del programa más grande que puede resolver (efectos del diskette), es de 60 restricciones y 90 variables; lo cual significa un máximo de 20 restricciones de enlace y 8 subproblemas.

Se decidió continuar con los formatos y el estilo de los programas descritos en los capítulos anteriores, por lo que DDW obtiene los

datos del problema que se quiere resolver, del archivo DDW.DAT y los resultados se escriben en el archivo DDW.RES, como se muestra en la figura 5.1. El programa DDW sólo resuelve problemas de minimización (max w = -min -w).

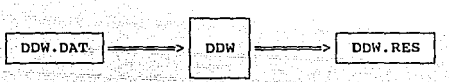


Figura 5.1. Diagrama de flujo de información.

5.2 CONCEPTOS BASICOS

El método de descomposición de Dantzig-Wolfe es un proceso iterativo, en donde en cada iteración se resuelven subproblemas lineales distintos. Las funciones objetivos de estos subproblemas varían de una iteración a otra y están determinadas por cálculos basados en las iteraciones anteriores.

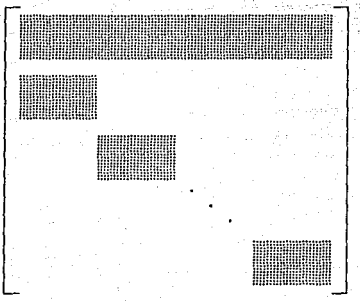
Tal método puede deducirse como una versión especial del simplex revisado, donde los subproblemas corresponden a la evaluación de los coeficientes de costo reducido para el problema principal.

La clase de problemas de programación lineal que se resuelven son los problemas multidivisionales. Su característica especial es que contienen varias divisiones de una organización, en donde las divisiones operan independientemente. Además, el problema se descompone en subproblemas separados que optimizan su operación.

Como resultado de esta característica, la tabla de coeficientes de las restricciones de problemas multidivisionales tiene la estructura angular

COEFICIENTES DE VARIABLES ACTIVAS — RESTRICCIONES EN:

1ª div 2ª div , , , ult. div

A = 

la organización de recursos requeridos por las divisiones.

recursos requeridos sólo para la primera división.

recursos requeridos sólo para la segunda división.

.
.
.

recursos requeridos sólo para la última división.

Los bloques sombreados representan las porciones de la tabla que tienen algún elemento distinto de cero para los coeficientes a_{ij}. Cada bloque contiene los coeficientes de las restricciones de un subproblema. El bloque largo contiene los coeficientes de las restricciones de enlace.

Los problemas son de la forma

$$\begin{aligned} &\text{Minimizar} && z = cx \\ &\text{sujeto a} && Ax = b \\ & && x \geq 0. \end{aligned} \qquad (1)$$

en donde la matriz A tiene la estructura (bloque angular):

$$A = \begin{bmatrix} L_1 & L_2 & \cdots & L_N \\ A_1 & & & \\ & A_2 & & \\ & & \ddots & \\ & & & A_N \end{bmatrix}$$

Al dividir los vectores **x**, **c** y ser **b** compatible con la división de **A**, se puede escribir (1) como

$$\begin{aligned} &\text{Minimizar} \quad z = \sum_{i=1}^{N} c_i x_i \\ &\text{sujeto a} \quad \sum_{i=1}^{N} L_i x_i = b_0 \qquad (2) \\ &\qquad A_i x_i = b_i \\ &\qquad x_i \geq 0, \quad i=1, \ldots, N. \end{aligned}$$

el cual es un problema de minimizar el costo total de N programas lineales distintos e independientes, excepto por la primera restricción (de enlace).

Los subproblemas son de la forma

$$\begin{aligned} &\text{Minimimizar } w = \quad c_i x_i \\ &\text{sujeto a} \qquad\quad A_i x_i = b_i \\ &\qquad\qquad\quad x_i \geq 0 \end{aligned}$$

Se supone que $S_i = \{x_i : A_i x_i = b_i, x_i \geq 0\}$ es acotado en el desarrollo de DDW, y una modificación cuando éste no es el caso puede verse en Lasdon L., Optimization Theory for Large Systems, [15].

El siguiente paso es expresar los puntos de S_i como combinaciones convexas de sus puntos extremos y sustituirlos en (2), obteniéndose de aquí el problema maestro. El desarrollo completo de este procedimiento se encuentra en Luenberger D., Linear and Nonlinear Programming, [16].

5.2.1 Estrategias de solución

Cuando el número de subsistemas es grande o cuando éstos son ya demasiado complejos es difícil resolverlos. En el curso del algoritmo algunos subproblemas se vuelven más activos en la generación de propuestas, y como la convergencia del algoritmo no se pierde si en algunos ciclos, sólo unos cuantos subproblemas se resuelven, podría ahorrarse esfuerzo y tiempo de CPU resolviendo sólo estos subproblemas.

El conocimiento de la naturaleza del problema puede indicar los subproblemas que son más importantes. Por ejemplo, en un modelo de energía multinacional un subproblema que corresponde a un país con un importante déficit o excedente de energía es probable que sea más activo en el proceso del algoritmo. Un subproblema que no es afectado por el último cambio de precios no necesita ser resuelto otra vez.

Este último aspecto sí fue contemplado en DDW. Los tres puntos en el diseño del algoritmo de descomposición son:

a) la resolución del problema maestro y de los subproblemas.

b) el manejo de datos para actualizar los subproblemas y el maestro; y

c) Las estrategias de resolución.

La eficiencia en la implantación del algoritmo depende primeramente de la rutina que se use para resolver los problemas lineales. El acceso al código del programa LINP (capítulo 1), no cubre todo el aspecto del diseño, se requiere una programación para unir el código LINP con los conceptos del algoritmo de descomposición.

5.3 ALGORITMO

El algoritmo que se utilizó para elaborar DDW está basado en Luenberger D., Linear and Nonlinear Programming, [16]; no fue diseñado para realizar dos fases y el método empleado para resolver el problema maestro es el simplex revisado.

PASO 1 Se forma la base inicial del problema maestro.

PASO 2 Se resuelven todos los subproblemas, con su respectiva función objetivo y se elige la solución del que proporcione el mejor incremento en el objetivo del problema maestro. El índice asociado a éste subproblema determina la variable que entra a la base del maestro.

Si alguno de los subproblemas resulta no factible, todo el problema es no factible e ir al paso 6. Si no existe solución que mejore el valor de la función objetivo, la solución actual es óptima e ir al paso 6.

PASO 3 Se calcula la columna del problema maestro asociada a la variable que entrará a la base.

PASO 4 Se determina la variable que saldrá de la base del problema maestro.

Si no existe tal variable, el problema original es no acotado e ir al paso 6.

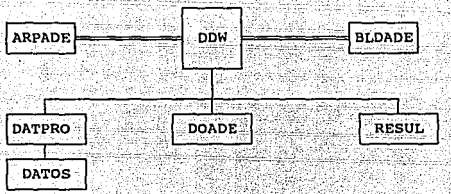
PASO 5 Se realiza el cambio de base en el problema maestro. Se actualizan multiplicadores simplex, que determinan las nuevas funciones objetivos de los subproblemas [4], ir al paso 2.

PASO 6 Fin del algoritmo.

5.4 ESTRUCTURA DEL PROGRAMA

El programa DDW tiene una estructura propia en su ejecución, la cual se muestra con las siguientes figuras. Cada cuadro representa una rutina en el programa; se describe el objetivo y la función de éstas. Son 9 rutinas las que coordinan el programa LINP con la teoría de descomposición, constituyendo 2476 líneas de código. En el apéndice B se encuentra esta programación.

i)



DDW — Programa principal.

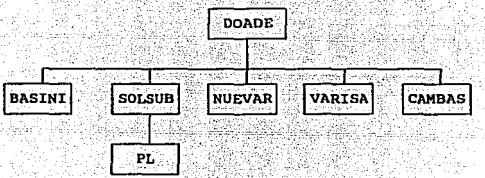
ARPADE — Archivo que contiene el tamaño de los parámetros que utiliza el programa.

BLDADE — Archivo en donde se encuentran dimensionados los arreglos comunes que se utilizan en el programa.

SUBRUTINA	OBJETIVO
DATPRO	Leer los datos del problema, del archivo DDW,DAT, y almacenar las matrices asociadas a las restricciones de enlace y a los subproblemas, en arreglos sin elementos ceros.
DATOS	Leer del archivo de datos DDW.DAT, el número de: restricciones (M), variables (N), restricciones de enlace (ME), subproblemas (NB), máximo de iteraciones (ITMAX), restricciones (MBLK(i)) y variables (NBLK(i)) en el i-ésimo subproblema; la matriz A y los vectores b y c.

DOADE	Resolver el problema con el método de descomposición de Dantzig-Wolfe.
RESUL	Escribir en el archivo DDW.RES, el tipo de solución encontrada (óptima, no factible, no acotada), y en caso de existir, la solución del problema.

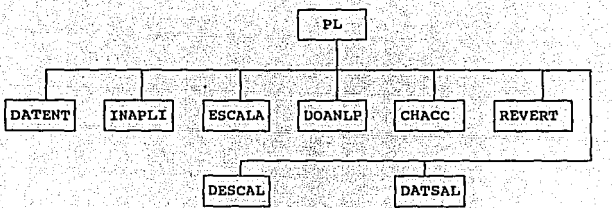
ii)



SUBRUTINA	OBJETIVO
BASINI	Formar la primera base del problema maestro, agregando variables de holgura en las restricciones de enlace, y con una variable de cada restricción de convexidad.
SOLSUB	Resolver los subproblemas obteniendo, si es que existe factibilidad, el bloque (subproblema), que indique la variable que deba entrar a la base del problema maestro.

PL	Resolver un subproblema lineal.
NUEVAR	Calcular la columna del problema maestro asociada a la variable que entrará a la base.
VARISA	Determinar la variable que saldrá de la base del problema maestro.
CAMBAS	Realizar el cambio de base en el problema maestro, actualizando el valor de los multiplicadores simplex y el de la función objetivo.

iii)



SUBRUTINA	OBJETIVO
DATENT	Transferir los datos del subproblema a las variables que utiliza LINP.
INAPLI	Inicializar algunas variables que se utilizan en LINP.
ESCALA	Rutina de LINP que escala el subproblema que se va a resolver.

DOANLP	Rutina de LINP que resuelve el subproblema lineal.
CHACC	Rutina de LINP que verifica la exactitud de la solución encontrada.
REVERT	Rutina de LINP que realiza una reinversión de la base del subproblema, si la exactitud no fue correcta en la solución obtenida.
DESCAL	Rutina de LINP que desescala el subproblema lineal que se resolvió.
DATSAL	Transferir la solución encontrada por LINP, a las variables definidas en DDW.

Las rutinas ESCALA, DOANLP y REVERT llaman a otras subrutinas, su descripción se puede ver en el capítulo 1.

5.5 FORMATO PARA DATOS DE ENTRADA (ARCHIVO DDW.DAT)

En esta sección se describen los datos de entrada para el programa, es decir la creación del archivo DDW.DAT.

Registro 1

Columna		
1-10	M	Número de restricciones.
11-20	NN	Número de variables.
21-30	ME	Número de restricciones de enlace.

31-40	NB	Número de subproblemas.
41-50	ITMAX	Número máximo de iteraciones; Sí es cero, entonces tendrá el valor heurístico de 3*(M+NN).

Registro 2

Columna		
1-2	I	Número de un subproblema.
3-6	MBLK(I)	Número de restricciones del i-ésimo subproblema.
7-10	NBLK(I)	Número de variables del i-ésimo subproblema.
11-12	I	
13-16	MBLK(I)	
17-20	NBLK(I)	
	.	
	.	
	.	
71-72	I	
73-76	MBLK(I)	
77-80	NBLK(I)	

El registro 2 se repite hasta cubrir todos los subproblemas, ocho por registro. Es importante que los subproblemas sean declarados en orden.

Registro 3

Columna		
1-10	999999999	Indica el fin de la descripción de los NB subproblemas.

Registro 4

Columna		
1-3	J	Número de un elemento diferente de cero de la función objetivo (Vector C).
5-10	C(J)	j-ésimo elemento de la función objetivo.
11-13	J	
15-20	C(J)	
	.	
	.	
	.	
71-73	J	
75-80	C(J)	

El registro 4 se repite hasta cubrir todos los elementos diferentes de cero de la función objetivo, ocho por registro.

Registro 5

Columna		
1-10	999999999	Indica el fin de los elementos de C.

Registro 6

Columna		
1-3	I	Número de un elemento del vector b, lado derecho.
4	S(I)	= 0 restricción de igualdad.
		= 1 restricción de menor o igual.

= 2 restricción de mayor o igual.

5-10	B(I)	i-ésimo elemento del vector b.
11-13	I	
14	S(I)	
15-20	B(I)	
	.	
	.	
	.	
71-73	I	
74	S(I)	
75-80	B(I)	

El registro 6 se repite hasta cubrir todos los elementos del vector b, ocho por registro. Cualquier elemento que no sea especificado, el programa asume como restricción de menor o igual con un valor muy grande para el elemento de B.

Registro 7

Columna		
1-10	999999999	Indica el fin de los elementos del vector b.

Registro 8

Columna		
5-10	I	Número de un renglón de la matriz A.
11-13	J	Número de una columna de la matriz A.
15-20	A(IJ)	Elemento diferente de cero de la matriz A.

21-23	J
25-30	A(IJ)
	.
	.
	.
71-73	J
75-80	A(IJ)

Todos los elementos del registro deben pertenecer al mismo renglón, pero no necesariamente en orden correcto en las columnas. Los renglones deben meterse en el orden correcto. El registro 8 debe ser repetido hasta cubrir con todos los elementos diferentes de cero de la matriz A, siete por registro.

Registro 9

Columna		
1-10	999999999	Indica el fin de los elementos diferentes de cero de la matriz A, y fin del archivo.

Por ejemplo, el problema

$$
\begin{aligned}
\text{Minimizar } z = \quad & -x_1 - x_2 - 2x_3 - x_4 \\
\text{sujeto a} \quad & x_1 + 2x_2 + 2x_3 + x_4 \leq 40 \\
& x_1 + 3x_2 \leq 30 \\
& 2x_1 + x_2 \leq 20 \\
& x_3 \leq 10 \\
& x_4 \leq 10 \\
& x_3 + x_4 \leq 15 \\
& x_1, x_2, x_3, x_4 \geq 0
\end{aligned}
$$

con cuatro variables, seis restricciones, de las cuales dos son de enlace, y dos subproblemas, se representa en el archivo DDW.DAT como

```
Columna 10        20        30        40        50        60...80

          6         4         1         2
 1    2    2 2    3   2
9999999999
  1      -1  2      -1  3      -2  4      -1
9999999999
  11     40  21     30  31     20  41     10  51     10  61     15
9999999999
          1  1      1  2       2  3       2  4       1
          2  1      1  2       3
          3  1      2  2       1
          4  3      1
          5  4      1
          6  3      1  4       1
9999999999
```

5.6 MANUAL DE USUARIO

El programa está diseñado para correr en una IBM-PC o compatible, con sistema operativo MS-DOS 3.2 o mayor y coprocesador matemático, y con un drive de 3 1\2". El usuario debe crear, por medio de algún editor el archivo DDW.DAT, con los datos del problema que desea resolver. Sólo problemas a minimizar se resuelven (max z = -min -z); de existir algún error en el archivo de datos (no haber asignado un valor a un elemento del vector b, c o de la matriz A, haber declarado un número negativo de restricciones o variables, no declarar en orden las restricciones del problema, tratar de declarar más valores de los que son, etc.) el programa suspende su ejecución hasta que el error sea corregido (editar de nuevo el archivo DDW.DAT). Los mensajes finales y la solución del problema, se escriben en el archivo DDW.RES.

Si no existen errores en el archivo DDW.DAT, se resuelve el problema con la instrucción

```
a:> DDW
```

Cuando el programa termina su ejecución

```
a:> _
```

con la instrucción

```
a:> type DDW.RES
```

se puede ver la solución que se obtuvo del problema.

El problema de la sección 5.5 pag 91, con su archivo DDW.DAT se toma de ejemplo para mostrar la forma en que DDW da los resultados (archivo DDW.RES), de un problema que resuelve. El archivo es:

```
NUMERO DE RESTRICCIONES =     6
NUMERO DE VARIABLES =    4
NUMERO MAXIMO DE ITERACIONES =    30


LA SOLUCION OPTIMA ES :

     X(  1) =       8.33333
     X(  2) =       3.33333
     X(  3) =      10.00000
     X(  4) =       5.00000

EL VALOR DE LA FUNCION OBJETIVO ES :      -36.66667

NUMERO DE ITERACIONES REALIZADAS     4
```

CAPITULO 6

RESULTADOS Y CONCLUSIONES

6.1 RESULTADOS

Como resultado de haber implantado a Fortran 77 estructurado, documentar y probar los programas LINP, BB, PLP y QP, incorporando un procedimiento de escalamiento al programa LINP; y de haber desarrollado el código necesario para obtener el programa DDW, se tiene:

- La elaboración de cinco documentos en donde se describen en forma detallada los programas (variables, parámetros y problemas de prueba); contienen el código fuente, el cual puede ser modificado o adaptado a las necesidades particulares de los investigadores; y se explica la forma de utilizar estos programas como subrutinas en programas de aplicación más complejos.

- Se incorporó un procedimiento de escalamiento al programa LINP; éste fue probado con los mismos problemas que se probó LINP sin escalamiento y se está utilizando junto con LINP en problemas de aplicación, tal como se menciona adelante. El procedimiento está basado en Murray W., Practical Optimazation, [17], y constituye 780 líneas de código en el programa (apéndice A).

- El código se encuentra en archivos de las terminales Vax del IIE, listos para crear ejecutables, conforme a las necesidades de los investigadores del Departamento.

- El programa LINP ha sido utilizado en la solución de diversos subproblemas de flujos óptimos, originados al resolver el problema de la planeación de la expansión de los medios de generación y transmisión de un sistema de potencia, con la descomposición de Benders. Los subproblemas se resuelven muchas veces en el proceso de solución; el tamaño de éstos es de 213 variables y a lo más de 100 restricciones, para un sistema de potencia consistente de 41 nodos eléctricos, 10 tipos de generación termoeléctrica, 16 embalses y 53 enlaces.

- El programa LINP se utiliza actualmente como una subrutina en un programa binario que utiliza el método de

Balas, el cual se desarrolla en el Departamento.

- El programa BB se utiliza actualmente en un programa para resolver el problema maestro resultante de aplicar la técnica de partición de Benders [18], al problema de compensación reactiva multianual. El problema forma parte de un proyecto vendido por el IIE y consiste en determinar la compensación para diferentes condiciones del sistema eléctrico, para un horizonte de 1 a 10 años, minimizando costos de inversión en equipo y satisfaciendo la operación económica y segura del sistema eléctrico para las diferentes condiciones del mismo. La dimensión de los problemas resueltos está entre 5 restricciones y 100 variables.

- Se aprovechó la estructura y el objetivo del programa LINP para realizar el programa DDW, descrito en el capítulo 5. Así, los subproblemas en el método de descomposición se resuelven con el procedimiento de escalamiento.

- Se dejan los ejecutables de los programas para funcionar en ambiente de computadora personal, dimensionados a mediana escala para propósitos estudiantiles.

6.2 CONCLUCIONES

Con base a las versiones actuales de los programas, se recomienda:

- Cambiar la forma de construir los archivos de datos; por ejemplo, otros formatos serían más sencillos y harían más fácil el elaborar y entender estos archivos.

- Desarrollar una interfase amigable para administrar tanto la información de entrada (Datos), como la de salida (Resultados). Quizá por medio de pantallas y menús podrían unirse los cinco programas.

- El procedimiento de escalamiento sólo fue implantado en LINP; como existe también el escalamiento en problemas enteros-mixtos y cuadráticos, los programas BB y QP serían más confiables si se les incorporara este procedimiento, ya que reducirían posibles errores de precisión en sus cálculos.

- Se considera al programa DDW como una primera versión, es decir, existen todavía muchas mejoras y casos que faltan

por resolver. Pero la oportunidad de contar con un programa como LINP, el más probado y usado de todos, hace que DDW que pueda ser de utilidad en el Departamento.

APENDICE A

CODIGO DE ESCALAMIENTO

```
      SUBROUTINE ESCALA
C     ***********************************************************
C     *                                                         *
C     *  OBJETIVO                                               *
C     *                                                         *
C     *  TRANSFORMAR EL PROBLEMA ORIGINAL EN UN PROBLEMA EQUIVALENTE,  *
C     *  CUYA CARACTERISTICA ES ESTAR ESCALADO.                 *
C     *                                                         *
C     ***********************************************************
C
C     ---------------------------------
C     * SE INCLUYE AREA DE PARAMETROS *
C     ---------------------------------
      INCLUDE 'APPARL.FOR'
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      COMMON / B      / B      ( 1:MAXM )
      COMMON / AA     / AA     ( 1:MAXA )
      COMMON / IROW   / IROW   ( 1:MAXA+1 )
      COMMON / JCOL   / JCOL   ( 1:MAXA )
      COMMON / BOUND  / BOUND  ( 1:MAXN )
      COMMON / C      / C      ( 1:MAXN )
      COMMON / ESCACO / ESCACO ( 1:MAXN )
      COMMON / ESCARE / ESCARE ( 1:MAXM )
      COMMON / MNOW   / MNOW
      COMMON / N      / N
C     ---------------------------------------------------
C     * SE INICIALIZA EL FACTOR DE ESCALAMIENTO PARA COLUMNAS *
C     ---------------------------------------------------
      DO  J = 1,N
         ESCACO(J) = 1.0
      END DO
C     ------------------------------------------------------
C     * SE INICIALIZA EL FACTOR DE ESCALAMIENTO PARA RENGLONES *
C     ------------------------------------------------------
      DO  I = 1,MNOW
         ESCARE(I) = 1.0
      END DO
C     ---------------------------------------
C     * SE ESCALA LA MATRIZ DE RESTRICCIONES *
C     ---------------------------------------
      CALL ESCAA
C     ---------------------------------------------------------
C     * SE ESCALA EL VECTOR DE COEFICIENTES REDUCIDOS ( C ) *
C     ---------------------------------------------------------
      DO  J = 1,N
         C(J) = C(J)/ESCACO(J)
      END DO
C     -----------------------------------------------------
C     * SE ESCALA EL LADO DERECHO DE LAS RESTRICCIONES ( B ) *
C     -----------------------------------------------------
      DO  I = 1,MNOW
         B(I) = B(I)/ESCARE(I)
      END DO
C     ---------------------------------------
C     * SE ESCALA EL VECTOR DE COTAS SUPERIORES *
C     ---------------------------------------
      DO  J = 1,N
```

```
            IF ( BOUND(J) .NE. 1.0 ) THEN
               BOUND(J) = BOUND(J)*ESCACO(J)
            END IF
         END DO
         RETURN
         END


         SUBROUTINE ESCAA
C        ****************************************************************
C        *                                                              *
C        *  OBJETIVO                                                    *
C        *                                                              *
C        *  ESCALAR LA MATRIZ DE RESTRICCIONES, QUE ESTA YA EN SU FORMA *
C        *  COMPACTADA (VECTOR AA).                                     *
C        *                                                              *
C        ****************************************************************
C
C        ------------------------------
C        * SE INCLUYE AREA DE PARAMETROS *
C        ------------------------------
         INCLUDE 'ARPARL.FOR'
C
         IMPLICIT REAL*8 (A-H,O-Z)
         LOGICAL EXITO
C
         COMMON / AA     / AA       ( 1:MAXA )
         COMMON / ESCACO / ESCACO   ( 1:MAXN )
         COMMON / ESCARE / ESCARE   ( 1:MAXN )
         COMMON / IROW   / IROW     ( 1:MAXN+1 )
         COMMON / JCOL   / JCOL     ( 1:MAXA )
         COMMON / KCOL   / KCOL     ( 1:MAXA )
         COMMON / MNOW   / MNOW
         COMMON / N      / N
C
         EXITO = .TRUE.
         DO WHILE (EXITO)
C           ------------------
C           * CALCULO DE ROO *
C           ------------------
            CALL CALROO(ROO)
C           ----------------------------
C           * ESCALAMIENTO DE RENGLON *
C           ----------------------------
            DO  I = 1,MNOW
               CALL ESCREN(I)
            END DO
C           ----------------------------
C           * ESCALAMIENTO DE COLUMNA *
C           ----------------------------
            DO  I = 1,MNOW
               KCOL(I) = IROW(I)
            END DO
            DO  J = 1,N
               CALL ESCCOL(J)
            END DO
C           ------------------
C           * CALCULO DE RO *
C           ------------------
            CALL CALROO(RO)
```

```
C     ------------------------------
C     A CRITERIO DE FIN DEL ALGORITMO A
C     ------------------------------
      ERR = DABS(RO - ROO)/DABS(ROO)
      EPS   1.0E-23
      IF (ERR .LT. EPS) THEN
         EXITO = .FALSE.
      END IF
   END DO
   RETURN
   END

   SUBROUTINE CALROO(ROO)
C  AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
C  A                                                                           A
C  A  OBJETIVO                                                                 A
C  A                                                                           A
C  A  CALCULAR EL MAXIMO COCIENTE ENTRE DOS ELEMENTOS DE UNA MISMA             A
C  A  COLUMNA, DE LA MATRIZ A.                                                 A
C  A                                                                           A
C  AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
C
C  ------------------------------
C  A SE INCLUYE AREA DE PARAMETROS A
C  ------------------------------
   INCLUDE 'ARPAPL.FOR'

   IMPLICIT REAL*8 (A-H,O-Z)

   COMMON / AA     / AA        ( 1:MAXA )
   COMMON / IROW   / IROW      ( 1:MAXN+1 )
   COMMON / JCOL   / JCOL      ( 1:MAXA )
   COMMON / KCOL   / KCOL      ( 1:MAXA )
   COMMON / KNOW   / KNOW
   COMMON / N      / N
C  ------------------------------------------------------------------------
C  A SE INICIALIZA EL ARREGLO AUXILIAR PARA EL MOVIMIENTO POR A
C  A COLUMNAS ATRAVES DE LA MATRIZ COMPACTADA.                 A
C  ------------------------------------------------------------------------
   DO  I   1,KNOW
      KCOL(I)   IROW(I)
   END DO
C  ------------------------------------------------------------------------
C  A MAXIMO COCIENTE ENTRE DOS ELEMENTOS DE UNA MISMA COLUMNA A
C  ------------------------------------------------------------------------
   ROO = -10.0E-30
C  ------------------
C  A SE CALCULA ROO A
C  ------------------
   DO J = 1,N
      DO I1=1,KNOW
         IF (JCOL(KCOL(I1)) .EQ. J) THEN
            DO I2=1,KNOW
               IF (JCOL(KCOL(I2)) .EQ. J) THEN
C              ------------------------------
C              A SE REALIZA EL COCIENTE A
C              ------------------------------
                  COSI = AA(KCOL(I1))/AA(KCOL(I2))
```

```
                COSI = DABS(COSI)
                IF ( COSI .GT. ROO ) THEN
                  ROO = COSI
                END IF
              END IF
            END DO
          END IF
        END DO
C       ---------------------
C       A SE ACTUALIZA KCOL A
C       ---------------------
        DO I = 1,NNOW
          IF (JCOL(KCOL(I)) .EQ. J) THEN
            KCOL(I) = KCOL(I)+1
          END IF
        END DO
      END DO
      RETURN
      END

      SUBROUTINE ESCREN(I)
C     ********************************************************************
C     *                                                                  *
C     *  OBJETIVO                                                        *
C     *                                                                  *
C     *  REALIZA EL ESCALAMIENTO EN LOS RENGLONES DE LA MATRIZ A         *
C     *                                                                  *
C     ********************************************************************
C
C     ---------------------------------
C     A SE INCLUYE AREA DE PARAMETROS A
C     ---------------------------------
      INCLUDE 'ARPAPL.FOR'
C
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 MAX,MIN
      INTEGER ELEM,FIN
C
      COMMON / AA     / AA        ( 1:MAXA )
      COMMON / ESCARC / ESCARC    ( 1:MAXM )
      COMMON / IROW   / IROW      ( 1:MAXM+1 )
      COMMON / MNOW   / MNOW
C
C     ---------------------------------------------------
C     A ELEMENTO MINIMO EN EL I-ESIMO RENGLON DE A A
C     ---------------------------------------------------
      MIN = 10.0E+30
C     ---------------------------------------------------
C     A ELEMENTO MAXIMO EN EL I-ESIMO RENGLON DE A A
C     ---------------------------------------------------
      MAX = -10.0E-30
C     ---------------------------------------------------------------
C     A CALCULO DEL MAXIMO Y MINIMO ELEMENTO DEL I-ESIMO RENGLON A
C     ---------------------------------------------------------------
      INI = IROW(I)
      FIN = IROW(I+1) - 1
      DO  ELEM = INI,FIN
        AIJ = AA(ELEM)
        AIJ = DABS(AIJ)
```

```
      IF ( AIJ .GT. MAX ) THEN
         MAX = AIJ
      END IF
      IF ( AIJ.LT.MIN ) THEN
         MIN = AIJ
      END IF
   END DO
C  ---------------------------------------------------------------
C  * CALCULO DEL FACTOR DE ESCALAMIENTO (ESCARC), PARA EL *
C  * I-ESIMO RENGLON.                                     *
C  ---------------------------------------------------------------
   SCI = MAX(MIN)
   SCI = SQRT(SCI)
C  ---------------------------------------------------------------
C  * ESCALAMIENTO DEL I-ESIMO RENGLON DE LA MATRIZ A *
C  ---------------------------------------------------------------
   DO  CLEM = INI,FIN
       AA(CLEM) = AA(CLEM)/SCI
   END DO
C  -----------------------------------------
C  * FACTOR DE ESCALAMIENTO PARA ESTE RENGLON *
C  -----------------------------------------
   ESCARC(I) = SCI*ESCARC(I)
   RETURN
   END


   SUBROUTINE ESCCOL(I)
C  ************************************************************************
C  *                                                                      *
C  *  OBJETIVO                                                            *
C  *                                                                      *
C  *  REALIZA EL ESCALAMIENTO EN LAS COLUMNAS DE LA MATRIZ A              *
C  *                                                                      *
C  ************************************************************************
C
C  -----------------------------
C  * SE INCLUYE AREA DE PARAMETROS *
C  -----------------------------
   INCLUDE 'AREAPL.FOR'
C
   IMPLICIT REAL*8 (A-H,O-Z)
   REAL*8 MAX,MIN
   INTEGER J
C
   COMMON / AA     / AA       ( 1:NAXA )
   COMMON / ESCACO / ESCACO   ( 1:NAXN )
   COMMON / IROW   / IROW     ( 1:NAXN+1 )
   COMMON / JCOL   / JCOL     ( 1:NAXA )
   COMMON / KCOL   / KCOL     ( 1:NAXA )
   COMMON / NNOW   / NNOW
C
C  ---------------------------------------------
C  * ELEMENTO MINIMO EN LA J-ESIMA COLUMNA DE A *
C  ---------------------------------------------
   MIN =  10.0E+30
C  ---------------------------------------------
C  * ELEMENTO MAXIMO EN LA J-ESIMA COLUMNA DE A *
C  ---------------------------------------------
   MAX = -10.0E-30
```

```
C     ------------------------------------------------------------
C     * CALCULO DEL MAXIMO Y MINIMO ELEMENTO DE LA J-ESIMA COLUMNA *
C     ------------------------------------------------------------
      DO I = 1,MNOW
         IF (KCOL(I) .LT. IROW(I+1)) THEN
            INI=KCOL(I)
            MJ = JCOL(INI)
            IF ( MJ .EQ. J ) THEN
               AIJ = AA(INI)
               AIJ = DABS(AIJ)
               IF ( AIJ .GT. MAX ) THEN
                  MAX = AIJ
               END IF
               IF ( AIJ .LT. MIN ) THEN
                  MIN = AIJ
               END IF
            END IF
         END IF
      END DO
C     ------------------------------------------------------------
C     * CALCULO DEL FACTOR DE ESCALAMIENTO (ESCACO), PARA LA *
C     * J-ESIMA COLUMNA.                                     *
C     ------------------------------------------------------------
      SCI = MAXMIN
      SCI = DSQRT(SCI)
C     ------------------------------------------------------------
C     * ESCALAMIENTO DE LA J-ESIMA COLUMNA DE LA MATRIZ A *
C     ------------------------------------------------------------
      DO I = 1,MNOW
         IF (KCOL(I) .LT. IROW(I+1)) THEN
            INI=KCOL(I)
            MJ = JCOL(INI)
            IF ( MJ .EQ. J ) THEN
               AA(INI) = AA(INI)/SCI
               KCOL(I)=KCOL(I)+1
            END IF
         END IF
      END DO
C     ------------------------------------------------
C     * FACTOR DE ESCALAMIENTO PARA ESTA COLUMNA *
C     ------------------------------------------------
      ESCACO(J) = ESCACO(J)*SCI
      RETURN
      END


      SUBROUTINE DESCAL
C     ***************************************************************
C     *                                                             *
C     *  OBJETIVO                                                   *
C     *                                                             *
C     *  VOLVER EL PROBLEMA A SU FORMA ORIGINAL Y TRANSFORMAR LOS   *
C     *  RESULTADOS OBTENIDOS, A SU VALOR REAL.                     *
C     *                                                             *
C     ***************************************************************
C
C     ------------------------------------
C     * SE INCLUYE AREA DE PARAMETROS *
C     ------------------------------------
      INCLUDE 'AREAPL.FOR'
```

```
      IMPLICIT REAL*8 (A-H,O-Z)
C
      COMMON / AA      / AA          ( 1:MAXA )
      COMMON / B       / B           ( 1:MAXM )
      COMMON / BOUND   / BOUND       ( 1:MAXN )
      COMMON / SLACK   / SLACK       ( 1:MAXM )
      COMMON / C       / C           ( 1:MAXN )
      COMMON / ESCACO  / ESCACO      ( 1:MAXN )
      COMMON / ESCARE  / ESCARE      ( 1:MAXM )
      COMMON / JCOL    / JCOL        ( 1:MAXA )
      COMMON / IROW    / IROW        ( 1:MAXA+1 )
      COMMON / X       / X           ( 1:MAXN )
      COMMON / Y       / Y           ( 1:MAXM )
      COMMON / YAC     / YAC         ( 1:MAXN )
      COMMON / MMOW    / MMOW
      COMMON / N       / N
C     ------------------------------------------
C     * SE DESESCALAN LAS VARIABLES PRIMALES *
C     ------------------------------------------
      DO  J = 1,N
         X(J) = X(J)/ESCACO(J)
      END DO
C     ------------------------------------------
C     * SE DESESCALAN LAS VARIABLES DUALES *
C     ------------------------------------------
      DO  I = 1,MMOW
         Y(I) = Y(I)/ESCARE(I)
      END DO
C     ------------------------------------------
C     * SE DESESCALAN LAS VARIABLES DE HOLGURA *
C     ------------------------------------------
      DO  I = 1,MMOW
         SLACK(I) = SLACK(I)/ESCARE(I)
      END DO
C     ------------------------------
C     * SE DESESCALA EL VECTOR B *
C     ------------------------------
      DO  I = 1,MMOW
         B(I) = B(I)*ESCARE(I)
      END DO
C     ------------------------------
C     * SE DESESCALA EL VECTOR C *
C     ------------------------------
      DO  J = 1,N
         C(J) = C(J)*ESCACO(J)
      END DO
C     ------------------------------
C     * SE DESESCALA EL VECTOR YAC *
C     ------------------------------
      DO  J = 1,N
         YAC(J) = YAC(J)*ESCACO(J)
      END DO
C     ------------------------------------------------------
C     * SE DESESCALA EL VECTOR DE COTAS SUPERIORES (BOUND) *
C     ------------------------------------------------------
      DO  J = 1,N
         IF ( BOUND(J).NE.-1.0 ) THEN
            BOUND(J) = BOUND(J)/ESCACO(J)
```

```
      END IF
      END DO
C     ------------------------------------------------
C     * SE DESESCALA LA MATRIZ DE RESTRICCIONES *
C     ------------------------------------------------
      CALL DESCAA
      RETURN
      END


      SUBROUTINE DESCAA
C     ***************************************************************
C     *                                                             *
C     *  OBJETIVO                                                   *
C     *                                                             *
C     *  DESESCALAR LA MATRIZ DE RESTRICCIONES, A.                  *
C     *                                                             *
C     ***************************************************************
C
C     ---------------------------------
C     * SE INCLUYE AREA DE PARAMETROS *
C     ---------------------------------
      INCLUDE 'ARPARPL.FOR'
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      COMMON / AA     / AA        ( 1:MAXA )
      COMMON / ESCACO / ESCACO    ( 1:MAXN )
      COMMON / ESCARE / ESCARE    ( 1:MAXM )
      COMMON / IROW   / IROW      ( 1:MAXN+1 )
      COMMON / JCOL   / JCOL      ( 1:MAXA )
      COMMON / MNOW   / MNOW
C
C     -------------------------------
C     * SE DESESCALA LA MATRIZ       *
C     -------------------------------
      DO  I = 1,MNOW
         INI = IROW(I)
         IFIN = IROW(I+1) - 1
         DO  IELEM = INI,IFIN
            J = JCOL(IELEM)
C           ------------------------------------------------------
C           * SE MULTIPLICA POR EL INVERSO DE LOS FACTORES      *
C           * DE ESCALAMIENTO                                   *
C           ------------------------------------------------------
            AA(IELEM) = AA(IELEM)*ESCARE(I)*ESCACO(J)
         END DO
      END DO
      RETURN
      END


      SUBROUTINE LP

C     ***************************************************************
C     *                                                             *
C     *  OBJETIVO                                                   *
C     *                                                             *
C     *  LLAMAR A DOALLP PARA RESOLVER EL PROBLEMA DE PL. SI EXISTE *
C     *  UNA SOLUCION PARA EL PROBLEMA (OPTIMA, NO ACOTADA O INCAC  *
```

```
C     *  TIBLE), LA RUTINA CHACC VERIFICA SU EXACTITUD: SI ESTA NO      *
C     *  ES CORRECTA SE REALIZA UNA REINVERSION.                        *
C     *                                                                 *
C     *******************************************************************
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      COMMON / ITP     / ITP
      COMMON / INREV   / INREV
      COMMON / IR      / IR
      COMMON / ISTATE  / ISTATE
      COMMON / IRMAX   / IRMAX
C
C     ---------------------------------------------------------------
C     * EN REALIDAD ESTAS PRIMERAS INSTRUCCIONES, HASTA ANTES  DEL  *
C     * DO WHILE, SON INNECESARIAS PARA EL PROGRAMA DE PROGRAMACION *
C     * LINEAL; ITP E IP CUENTAN YA INICIALIZADAS EN INICIA, ADEMAS *
C     * ISTATE EN ESTE MOMENTO NUNCA SERA IGUAL A 10.               *
C     * SU RAZON DE EXISTIR EN LP, ES CUANDO LA RUTINA SE OCUPA EN  *
C     * PROGRAMAS MAS COMPLEJOS (PROGRAMACION CUADRATICA, ENTERA Y  *
C     * PARAMETRICA.                                                *
C     ---------------------------------------------------------------
C
C     ---------------------------------------------------
C     * SE INICIALIZA EL CONTADOR DE NUMERO DE ITERACIONES *
C     ---------------------------------------------------
      ITP = 0
C     ---------------------------------------------------
C     * SE INICIALIZA EL CONTADOR DE NUMERO DE REINVERSIONES *
C     ---------------------------------------------------
      IR = 0
C     ---------------------------------
C     * SE ESCALA EL PROBLEMA LINEAL *
C     ---------------------------------
      CALL ESCALA
C     ---------------------------------
C     * ISTATE   10 HAY UNA BASE DISPONIBLE *
C     ---------------------------------
      IF ( ISTATE .EQ. 10 ) THEN
C        ---------------------------------------------------------
C        * HAN SIDO AGREGADAS / HABILITADAS LAS VARIABLES DE HOLGURA *
C        ---------------------------------------------------------
         INREV = 1
      END IF
C     ---------------------------------------------
C     * MIENTRAS EXISTA O NO UNA BASE DISPONIBLE *
C     ---------------------------------------------
      DO WHILE ( (ISTATE.EQ.10).OR.(ISTATE.EQ.11).OR.(ISTATE.EQ.0).OR.
     >           (ISTATE.EQ.12) )
C        ---------------------------
C        * BUSCA SOLUCION AL PROBLEMA *
C        ---------------------------
         CALL DUALLP
C        ---------------------------------------------------------
C        * SI LA SOLUCION ES COMPLETA (OPTIMA,NO ACOTADA O INCACTIBLE) *
C        ---------------------------------------------------------
         IF ( ISTATE .GE. 0 ) THEN
C           ---------------------
C           * VERIFICA EXACTITUD *
C           ---------------------
```

```
            CALL CHACC
          END IF
C         -------------------------------------------
C         A ISTATE = 7 LA EXACTITUD NO ES CORRECTA A
C         -------------------------------------------
          IF ((ISTATE .EQ. 7).AND.(IR .LT. (RMAX)) THEN
C            ------------------------------
C            A SE REALIZA UNA REINVERSION A
C            ------------------------------
             CALL REVERT
C            ---------------------------
C            A EXISTE UNA BASE COMPLETA A
C            ---------------------------
             ISTATE = 11
          END IF
        END DO
C       ------------------------------------
C       A SE DESESCALA EL PROBLEMA LINEAL A
C       ------------------------------------
        CALL DESCAL
        RETURN
        END
```

APENDICE B

CODIGO DE DESCOMPOSICION

```
C     PROGRAM PRUEBA
C     ************************************************************
C     *                                                          *
C     *  OBJETIVO                                                *
C     *                                                          *
C     *  RESOLVER PROBLEMAS DE PROGRAMACION LINEAL DE GRAN ESCALA QUE *
C     *  PRESENTEN LA ESTRUCTURA DE BLOQUE ANGULAR, MEDIANTE EL METODO *
C     *  DE DESCOMPOSICION DE DANTZIG WOLFE.                     *
C     *                                                          *
C     ************************************************************
C
C     ------------------------------
C     * SE INCLUYE AREA DE PARAMETROS *
C     ------------------------------
      INCLUDE 'ARPARL.FOR'
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      REAL*8   XE(MAXN)
      LOGICAL  NERR
C
C     ------------------------------
C     * APERTURA DEL ARCHIVO DE DATOS *
C     ------------------------------
      OPEN ( UNIT = 1, FILE= ' DDW.DAT ',
     >       STATUS = 'OLD',
     >       FORM = 'FORMATTED',
     >       ORGANIZATION = 'SEQUENTIAL' )
C     ------------------------------ ---- ------ ----
C     * APERTURA DEL ARCHIVO DE RESULTADOS *
C     ------------------------------------------
      OPEN ( UNIT = 2, FILE= ' DDW.RES ',
     >       STATUS = 'NEW',
     >       FORM = 'FORMATTED',
     >       ORGANIZATION = 'SEQUENTIAL' )
      NERR =.TRUE.
C     ------------------------------ ------
C     * SE LEEN LOS DATOS DEL PROBLEMA *
C     ------------------------------ --
      CALL DATPRO( NERR )
C     ------------------------------ ------ ------
C     * SI NO EXISTE ERROR EN EL ARCHIVO DE DATOS *
C     ------------------------------ ------ ------
      IF ( NERR ) THEN
         ISTAT = LIB$INIT_TIMER()
C        ------------------------
C        * SE RESUELVE EL PROBLEMA *
C        ------------------------
         CALL DUAL0( INFO,UBJR,XE )
         ISTAT = LIB$SHOW_TIMER('...')
C        ------------------------
C        * SE ESCRIBEN RESULTADOS *
C        ------------------------
         CALL RESUL( INFO,UBJR,XE )
      END IF
      STOP
      END
```

```
      SUBROUTINE DATPRO( NERR )
C     ***********************************************************
C     *                                                         *
C     *  OBJETIVO                                               *
C     *                                                         *
C     *  LEER LOS DATOS DEL PROBLEMA, DEL ARCHIVO DOM.DAT, Y ALMACENAR *
C     *  LAS MATRICES ASOCIADAS A LAS RESTRICCIONES DE ENLACE Y A LOS *
C     *  SUBPROBLEMAS EN ARREGLOS, SIN ELEMENTOS CERO.          *
C     *                                                         *
C     ***********************************************************
C
C     ------------------------------------
C     * SE INCLUYE AREA DE PARAMETROS *
C     ------------------------------------
      INCLUDE 'ARPAPL.FOR'
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      INTEGER  RENI,COLJ,C1,NBLK(MAXS),NBLKS(MAXS)
      REAL*8   A(MAXE*MAXM,MAXN),MA,LI
      LOGICAL NERR
C
      COMMON / NE    / ME
      COMMON / NB    / NB
      COMMON / SMALL / SMALL
      COMMON / IB    / IP      ( 1:MAXS )
      COMMON / JX    / JX      ( 1:MAXS+1 )
      COMMON / BIG   / BIG
      COMMON / IREN  / IREN    ( 1:MAXMA )
      COMMON / JJCO  / JJCO    ( 1:MAXC )
      COMMON / KBLK  / KBLK    ( 1:MAXS )
      COMMON / LI    / LI      ( 1:MAXMA )
      COMMON / MA    / MA      ( 1:MAXA )
      COMMON / RENI  / RENI    ( 1:MAXN+1 )
      COMMON / COLJ  / COLJ    ( 1:MAXA )
      COMMON / KBLKS / KBLKS   ( 1:MAXS )
C
C     ------------------------------------------
C     * VALORES REPRESENTATIVOS EN EL PROGRAMA *
C     ------------------------------------------
      SMALL = 1.0E-8
      BIG = 1.0E8
C     --------------------
C     * LECTURA DE DATOS *
C     --------------------
      CALL DATOS( NERR,NBLK,NBLKS,A )
C     ------------------------------------------
C     * NO EXISTE ERROR EN EL ARCHIVO DE DATOS *
C     ------------------------------------------
      IF ( NERR ) THEN
C        ------------------------------------------------------
C        * SE LLENAN LOS ARREGLOS QUE APUNTAN A LAS DIVISIONES *
C        * ENTRE LOS SUBPROBLEMAS Y LAS RESTRICCIONES DE ENLACE *
C        ------------------------------------------------------
C
C        --------------------
C        * POR RENGLONES *
C        --------------------
         IB(1) = 1 + ME
         DO I = 1,NB
            IB(I+1) = IB(I) + NBLK(I)
```

```
      END DO
C     ------------------
C     * POR COLUMNAS *
C     ------------------
      JX(1) = 1
      DO J = 1,NB
         JX(J+1) = JX(J) + NBLK(J)
      END DO
C     ------------------------------------------------------------
C     * SE GUARDAN LOS COEFICIENTES DE LAS RESTRICCIONES *
C     * DE ENLACE EN FORMA COMPACTADA                    *
C     ------------------------------------------------------------
      K = 1
      CJ = 1
C     ------------------------------------------------------
C     * APUNTADOR AL RENGLON DEL I-ESIMO ELEMENTO *
C     ------------------------------------------------------
      IREN(1) = 1
C     ------------------------------------------------------
C     * APUNTADOR ALA COLUMNA DEL I-ESIMO ELEMENTO *
C     ------------------------------------------------------
      IJCO(1) = 1
C     ------------------------------------------------------
C     * APUNTADOR AL I-ESIMO BLOQUE(SUBPROBLEMA) *
C     ------------------------------------------------------
      KBLK(1) = 1
      DO KB = 1,NB
         DO J = JX(KB),JX(KB+1)-1
            DO I = 1,IO(1)-1
C           ------------------------------------------------------
C           * SE ALMACENAN SOLO ELEMENTOS DISTINTOS DE CERO *
C           ------------------------------------------------------
               IF ( A(I,J).NE.0 ) THEN
                  LI(K) = A(I,J)
                  IREN(K) = I
                  K = K + 1
               END IF
            END DO
            IJCO(J+1) = K
            CJ = CJ + 1
         END DO
         KBLK(KB+1) = CJ
      END DO
C     ------------------------------------------------------------
C     * SE GUARDA LA MATRIZ DE COEFICIENTES (A1) DEL I-ESIMO *
C     * SUBPROBLEMA, EN FORMA COMPACTADA.                    *
C     ------------------------------------------------------------
      K = 1
      CJ = 1
C     ------------------------------------------------------
C     * APUNTADOR AL RENGLON DEL I-ESIMO ELEMENTO *
C     ------------------------------------------------------
      REN1(1) = 1
C     ------------------------------------------------------
C     * APUNTADOR ALA COLUMNA DEL I-ESIMO ELEMENTO *
C     ------------------------------------------------------
      COL1(1) = 1
C     ------------------------------------------------------
C     * APUNTADOR AL I-ESIMO BLOQUE(SUBPROBLEMA) *
C     ------------------------------------------------------
```

```
      KBLKS(1) = 1
      II = 1
      DO KB = 1,NB
         DO I = IB(KB),IB(KB+1)-1
            JJ = 1
            DO J = JX(KB),JX(KB+1)-1
C              ----------------------------------------------
C              * SE ALMACENAN SOLO ELEMENTOS DISTINTOS DE CERO *
C              ----------------------------------------------
               IF ( A(I,J).NE.0 ) THEN
                  MA(K) = A(I,J)
                  COLJ(K) = JJ
                  K = K + 1
               END IF
               JJ = JJ + 1
            END DO
            RENI(II+1) = K
            CJ = CJ + 1
            II = II + 1
         END DO
         KBLKS(KB+1) = CJ
      END DO
      END IF
      RETURN
      END

      SUBROUTINE BASINI
C     ***************************************************************
C     *                                                             *
C     *  OBJETIVO                                                   *
C     *                                                             *
C     *  FORMAR LA PRIMERA BASE PARA EL PROBLEMA MAESTRO, AGREGANDO *
C     *  VARIABLES DE HOLGURA EN LAS RESTRICCIONES DE ENLACE Y CON  *
C     *  UNA VARIABLE DE CADA RESTRICCION DE CONVEXIDAD.            *
C     *                                                             *
C     ***************************************************************
C
C     ---------------------------------
C     * SE INCLUYE AREA DE PARAMETROS *
C     ---------------------------------
      INCLUDE 'ARPAPL.FOR'
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      INTEGER XBLK
      REAL*8  LAMDA,INVM
C
      COMMON / ME     / ME
      COMMON / NB     / NB
      COMMON / VB     / VB       ( 1:MAXM+MAXE )
      COMMON / IB     / IB       ( 1:MAXS )
      COMMON / INDI   / INDI     ( 1:MAXS )
      COMMON / INDJ   / INDJ     ( 1:MAXPE )
      COMMON / XBLK   / XBLK     ( 1:MAXS )
      COMMON / LAMDA  / LAMDA    ( 1:MAXE+MAXS )
      COMMON / XRG    / XRG      ( 1:MAXE+MAXM )
      COMMON / INVM   / INVM     ( 1:MAXE+MAXS,1:MAXE+MAXS )
      COMMON / XM     / XM       ( 1:MAXS,1:MAXPE )
```

```
C     ------------------------------------------------------------------
C     * SE AGREGAN VARIABLES DE HOLGURA EN LAS RESTRICCIONES DEL  *
C     * PROBLEMA MAESTRO, ASOCIADAS A LAS RESTRICCIONES DE ENLACE *
C     ------------------------------------------------------------------
      DO I = 1,ME
         DO J = 1,ME
C           ------------------------------------
C           * MATRIZ INVERSA DEL PROBLEMA MAESTRO *
C           ------------------------------------
            INVM(I,J) = 0.0
         END DO
         INVM(I,I) = 1.0
C        ------------------------------------
C        * VALOR DE LA I-ESIMA VARIABLE BASICA *
C        ------------------------------------
         XBG(I) = VB(I)
C        ---------------------------
C        * MULTIPLICADORES SIMPLEX *
C        ---------------------------
         LAMDA(I) = 0.0
C        ------------------------------------------------------------------
C        * ARREGLOS QUE SIRVEN PARA IDENTIFICAR LAS SOLUCIONES BASICAS *
C        * EN EL PROBLEMA MAESTRO                                       *
C        ------------------------------------------------------------------
         INDI(I) = 0
         INDJ(I) = 0
      END DO
      II = 1
C     ------------------------------------------------------------------
C     * INTEGRA LA BASE EL I-ESIMO COEFICIENTE DE PONDERACION *
C     * DEL I-ESIMO SUBPROBLEMA                               *
C     ------------------------------------------------------------------
      DO I = IB(1),NR+ME
         DO J = IB(1),NR+ME
            INVM(I,J) = 0.0
         END DO
         INVM(I,I) = 1.0
         XBG(I) = 1.0
C        ---------------------------
C        * MULTIPLICADORES SIMPLEX *
C        ---------------------------
         LAMDA(I) = 0.0
C        ------------------------------------------------------------------
C        * VALOR DE LAS VARIABLES BASICAS, NO DE HOLGURA, EN LA *
C        * BASE INICIAL                                         *
C        ------------------------------------------------------------------
         VM(II,I) = XBG(I)
C        ------------------------------------------------------------------
C        * INDICADOR DEL NUMERO DE PUNTOS EXTREMOS QUE SE HAN  *
C        * ENCONTRADO, CORRESPONDIENTES AL I-ESIMO SUBPROBLEMA *
C        ------------------------------------------------------------------
         KBLK(II) = 1
C        ------------------------------------------------------------------
C        * ARREGLOS QUE SIRVEN PARA IDENTIFICAR LAS SOLUCIONES BASICAS *
C        * EN EL PROBLEMA MAESTRO                                      *
C        ------------------------------------------------------------------
         INDI(II) = II
         INDJ(II) = 1
         II = II + 1
      END DO
```

```
      RETURN
      END

      SUBROUTINE SOLSUB( R,EXITO,EXITO1,IPFS,RK,KMIN,VX )
C     ***************************************************************
C     *                                                             *
C     *  OBJETIVO                                                   *
C     *                                                             *
C     *  RESOLVER LOS SUBPROBLEMAS OBTENIENDO, SI ES QUE NO EXISTE  *
C     *  INFACTIBILIDAD O UNA SOLUCION NO ACEPTADA EN ALGUNO DE ELLOS.*
C     *  EL BLOQUE(SUBPROBLEMA) QUE INDIQUE QUE VARIABLE ENTRARA ALA*
C     *  BASE DEL PROBLEMA MAESTRO.                                 *
C     *                                                             *
C     ***************************************************************
C
C     ---------------------------------
C     * SE INCLUYE AREA DE PARAMETROS *
C     ---------------------------------
      INCLUDE 'ARPAPL.EQP'
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      INTEGER  RENI,COLJ,RENX,ESTADO,EST(MAXS)
      REAL*8   MA,LI,LAMDA,VC(MAXN),VX(MAXN),FOB(MAXS)
      LOGICAL EXITO,EXITO1
C
      COMMON / NB     / NB
      COMMON / VR     / VR      ( 1:MAXN+MAXE )
      COMMON / VS     / VS      ( 1:MAXN+MAXE )
      COMMON / CC     / CC      ( 1:MAXN )
      COMMON / ID     / ID      ( 1:MAXS )
      COMMON / JX     / JX      ( 1:MAXS+1 )
      COMMON / IREN   / IREN    ( 1:MAXNA )
      COMMON / JJCO   / JJCO    ( 1:MAXE )
      COMMON / KBLK   / KBLK    ( 1:MAXS )
      COMMON / LI     / LI      ( 1:MAXNA )
      COMMON / MA     / MA      ( 1:MAXA )
      COMMON / RENI   / RENI    ( 1:MAXN+1 )
      COMMON / COLJ   / COLJ    ( 1:MAXA )
      COMMON / KBLKS  / KBLKS   ( 1:MAXS )
      COMMON / LAMDA  / LAMDA   ( 1:MAXE+MAXS )
      COMMON / SMALL  / SMALL
      COMMON / ITMAX  / ITMAX
C
C     ----------------------------------------
C     * SE RESUELVE EL I-ESIMO SUBPROBLEMA *
C     ----------------------------------------
      I = 1
      DO WHILE( I.LE.NB .AND. EXITO )
         IF ( EXITO1 ) THEN
C        ------------------------------------------------
C           * CALCULO DE LA FUNCION OBJETIVO DEL *
C           * I-ESIMO SUBPROBLEMA                 *
C        ------------------------------------------------
C           ------------------
C           * NUMERO DE VARIABLES *
C           ------------------
            NV = JX(I+1) - JX(I)
            DO J = 1,NV
```

```
            VC(J) = CS(JX(I)+J-1)
            TNI = JJCU(KBLK(I))-1)
            IFI = JJCU(KBLK(I)+1)-1
            DO K = INI,IFI
C           ---------------------------------------------------
C              A J-ESIMO ELEMENTO EN LA FUNCION OBJETIVO      A
C              A DEL I-ESIMO SUBPROBLEMA.                     A
C              ---------------------------------------------
               VC(J) = VC(J) - L1(K)*LAMBDA(IREN(K))
            END DO
            VC(J) = -VC(J)
         END DO
C        ------------------------
C        A NUMERO DE VARIABLES A
C        ------------------------
         NR = IB(I+1) - IB(I)
C        ------------------------------
C        A NUMERO MAXIMO DE ITERACIONES A
C        ------------------------------
         ITMX = 3A(NV+NP)
C        ---------------------------------
C        A NUMERO MAXIMO DE REINVERSIONES A
C        ---------------------------------
         REMX = NV
C        ----------------
C        A VALOR OBJETIVO A
C        ----------------
         EOBJ = 0.0
C        ------------------------
C        A ESTADO DE LA SOLUCION A
C        ------------------------
         ESTADO = 0
C        ---------------------------
C        A SE RESUELVE EL PROBLEMA A
C        ---------------------------
         CALL PLC( RENI(KBLKS(I)),COLJ(RENI(KBLKS(I))),VN(IB(I)),
     >             MA(RENI(KBLKS(I))),VC,NR,NV,VS(IB(I)),ITMX,
     >             REMX,VX(JX(I)),EOBJ,ESTADO )
         EST(I) = ESTADO
         EOB(I) = EOBJ
      END IF
C     ---------------------------
C     A SI LA SOLUCION NO ES OPTIMA A
C     ---------------------------
      IF ( EST(I).NE.1 ) THEN
         IF ( EST(I).EQ.2 ) THEN
C           ---------------------------------
C           A EL PROBLEMA ES INFACTIBLE A
C           ---------------------------------
            IRES = 2
            EXITO = .FALSE.
         END IF
      ELSE
C        ----------------
C        A VALOR OBJETIVO A
C        ----------------
         BI = -EOB(I) - LAMBDA(IB(1):I-1)
C        ---------------------------------------
C        A SI ES MEJOR ESTE VALOR QUE EL ANTERIOR A
C        ---------------------------------------
```

```
            IF ( RI.LT.R ) THEN
               R = RI
C              ------------------------------------
C              * POR RAZONES DE ESTABILIDAD NUMERICA *
C              ------------------------------------
               IF ( DABS(R).LT.SMALL ) THEN
                  R = 0.0
               END IF
               RR = FOB(I)
C              --------------------------------------------
C              * SUBPROBLEMA QUE DETERMINA QUE VARIABLE DEBE *
C              * ENTRAR A LA BASE.                           *
C              --------------------------------------------
               KBIN = I
            END IF
         END IF
C        ------------------------------------
C        * SE RESUELVE EL SIGUIENTE SUBPROBLEMA *
C        ------------------------------------
         I = I+1
      END DO
      RETURN
      END

      SUBROUTINE NUEVAR( KBIN,VX,Y )
C     ***************************************************************
C     *                                                             *
C     * OBJETIVO                                                    *
C     *                                                             *
C     * CALCULAR LA COLUMNA DEL PROBLEMA MAESTRO ASOCIADA A LA      *
C     * VARIABLE QUE ENTRARA A LA BASE.                             *
C     *                                                             *
C     ***************************************************************
C
C     ---------------------------------
C     * SE INCLUYE AREA DE PARAMETROS *
C     ---------------------------------
      INCLUDE 'ARPAPL.FOR'
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      REAL*8  VX(MAXN),Y(MAXE+MAXS),Z(MAXE+MAXS),1),INVM
C
      COMMON / ME     / ME
      COMMON / NB     / NB
      COMMON / IB     / IB      ( 1:MAXS )
      COMMON / JX     / JX      ( 1:MAXS+1 )
      COMMON / LI     / LI      ( 1:MAXMA )
      COMMON / JJCO   / JJCO    ( 1:MAXE )
      COMMON / IREN   / IREN    ( 1:MAXMA )
      COMMON / KBLK   / KBLK    ( 1:MAXS )
      COMMON / INVM   / INVM    ( 1:MAXE+MAXS,1:MAXE+MAXS )
C
C     -------------------------------------------------------------
C     * SE INICIALIZA ARREGLO AUXILIAR PARA EL CALCULO DE LA COLUMNA *
C     -------------------------------------------------------------
      DO I = 1,ME
         Z(I) = 0.0
      END DO
```

```
C     ---------------------------
C     * SE MULTIPLICA  L(I)A(J) *
C     ---------------------------
      I = 1
      DO J = KBLK(KB1N),KBLK(KB1N+1)-1
         DO K = JJCO(J),JJCO(J+1)-1
            Z(IPEN(K)) = Z(IPEN(K)) + LI(K)*AV(I)*JX(KB1N-1)
         END DO
         I = I+1
      END DO
C     ---------------------------
C     * SE LLENA EL ARREGLO Z *
C     ---------------------------
      DO I = 1,NB
         IF ( I.EQ.KB1N) THEN
            Z(ME+I) = 1.0
         ELSE
            Z(ME+I) = 0.0
         END IF
      END DO
C     ---------------------------------------------
C     * SE MULTIPLICA  L(I)A(J) (Z) POR LA INVERSA *
C     ---------------------------------------------
      DO I = 1,ME+NB
         G = 0.0
         DO J = 1,ME+NB
            G = G + INVR(I,J)*Z(J)
         END DO
C        ---------------------------------------------
C        * I-ESIMO ELEMENTO DE LA COLUMNA CALCULADA *
C        ---------------------------------------------
         Y(I) = G
      END DO
      RETURN
      END


      SUBROUTINE VARISA( ISAL,Y )
C     ***************************************************************************
C     *                                                                         *
C     *  OBJETIVO                                                               *
C     *                                                                         *
C     *  DETERMINAR LA VARIABLE QUE SALDRA DE LA BASE DEL PROBLEMA              *
C     *  MAESTRO                                                                *
C     *                                                                         *
C     ***************************************************************************


C     ---------------------------------
C     * SE INCLUYE AREA DE PARAMETROS *
C     ---------------------------------
      INCLUDE 'ARPAPL.FOP'

      IMPLICIT REAL*8 (A-H,O-Z)

      REAL*8  Y(MAXF+MAXS)

      COMMON / ME    / ME
      COMMON / NB    / NB
      COMMON / BIG   / BIG
      COMMON / SMALL / SMALL
```

```
      COMMON / XRG    / XRG    ( 1:MAXE+MAXS )
C
C     ---------------------------------------------
C     A RENGLON DE LA INVERSA ASOCIADO A LA VARIABLE A
C     A QUE SALDRA DE LA BASE                      A
C     ---------------------------------------------
      ISAL = 0
C     ---------------------
C     A ELEMENTO PIVOTE A
C     ---------------------
      PIV = BIG
C     -----------------------------
C     A SE REALIZAN LOS COCIENTES A
C     -----------------------------
      DO I = 1,ME+NB
         IF ( Y(I).GT.0.0 ) THEN
            COC = XRG(I)/Y(I)
C           -------------------------
C           A ESTABILIDAD NUMERICA A
C           -------------------------
            IF ( DABS(COC).LT.SMALL ) THEN
               COC = 0.0
            END IF
            IF ( COC.LT.PIV ) THEN
               PIV = COC
               ISAL = I
            END IF
         END IF
      END DO
      RETURN
      END


      SUBROUTINE CAMBAS( EXITO1,ISAL,Y,RR,KBIN,ITM,ORIN )
C     ***************************************************************************
C     *                                                                         *
C     *  OBJETIVO                                                               *
C     *                                                                         *
C     *  REALIZAR EL CAMBIO DE BASE EN EL PROBLEMA MAESTRO, CALCULANDO          *
C     *  EL VALOR DE LOS MULTIPLICADORES SIMPLEX Y DE LA FUNCION.               *
C     *                                                                         *
C     ***************************************************************************
C
C     ---------------------------------
C     A SE INCLUYE AREA DE PARAMETROS A
C     ---------------------------------
      INCLUDE 'ARPAPL.FOR'
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      INTEGER XBLK
      REAL*8  LAMDA,INVH,Y(MAXE+MAXS)
      LOGICAL EXITO1
C
      COMMON / ME     / ME
      COMMON / NB     / NB
      COMMON / IND1   / IND1    ( 1:MAXS )
      COMMON / IND3   / IND3    ( 1:MAXPE )
      COMMON / IND    / IND     ( 1:30 )
      COMMON / XBLK   / XBLK    ( 1:MAXS )
```

```
      COMMON / LAMDA / LAMDA  ( 1:MAXX*MAXX )
      COMMON / XRB   / XRB    ( 1:MAXX,1:MAXX )
      COMMON / INVA  / INVA   ( 1:MAXX*(MAXX,1:MAXX*MAXX )
      COMMON / XB    / XB     ( 1:MAXX,1:MAXXE )
      COMMON / SMALL / SMALL
C
C     ------------------------------------------------------------
C     * SE ACTUALIZA EL RENGLON DE LA INVERSA ASOCIADO A LA      *
C     * VARIABLE QUE SALE DE LA BASE.                            *
C     ------------------------------------------------------------
      DO 3 J = 1,MXBND
C        ---------------------------------------
C        * MATRIZ INVERSA DEL PROBLEMA MAESTRO *
C        ---------------------------------------
         INVM(JSAL,J) = INVM(JSAL,J)/Y(JSAL)
C        ------------------------
C        * ESTABILIDAD NUMERICA *
C        ------------------------
         IF ( DABS(INVM(JSAL,J)).LT.SMALL ) THEN
            INVM(JSAL,J) = 0.0
         END IF
    3 END DO
C     ------------------------------
C     * VALOR DE UNA VARIABLE BASICA *
C     ------------------------------
      XRB(JSAL) = XRB(JSAL)/Y(JSAL)
C     ------------------------
C     * ESTABILIDAD NUMERICA *
C     ------------------------
      IF ( DABS(XRB(JSAL)).LT.SMALL ) THEN
         XRB(JSAL) = 0.0
      END IF
C     ------------------------------------------------------------
C     * SE ACTUALIZA EL VALOR DE LA FUNCION OBJETIVO             *
C     ------------------------------------------------------------
      OBJN = OBJN + CRAXRB(JSAL)
C     ------------------------------------------------------------
C     * SE ACTUALIZAN LOS ARREGLOS QUE SIRVEN PARA               *
C     * IDENTIFICAR LAS VARIABLES QUE SE ENCUENTRAN              *
C     * EN LA BASE                                               *
C     ------------------------------------------------------------
      IF ( EXITO1 ) THEN
         XBLK(KBIN) = XBLK(KBIN) + 1
      END IF
      INDI(ITR) = XBLK(KBIN)
C     ------------------------------------------------------------
C     * VARIABLE DEL PROBLEMA MAESTRO QUE SALE DE LA BASE        *
C     ------------------------------------------------------------
      IF ( INDI(JSAL).NE.0 ) THEN
         XM(INDI(JSAL),INDJ(JSAL)) = 0.0
      END IF
C     ------------------------------------------------------------
C     * VARIABLE DEL PROBLEMA MAESTRO QUE ENTRA A LA BASE        *
C     ------------------------------------------------------------
      XM(KBIN,XBLK(KBIN))   XRB(JSAL)
      INDI(JSAL) = KBIN
      INDJ(JSAL) = XBLK(KBIN)
C     ------------------------------
C     * SE ACTUALIZA LA MATRIZ INVERSA *
C     ------------------------------
```

```
      DO I = 1,ME+NP
         IF ( I.NE.JSAL ) THEN
            DO J = 1,ME+NP
               INVM(I,J) = INVM(I,J) - Y(I)*INVM(JSAL,J)
C              ------------------------
C              * ESTABILIDAD NUMERICA *
C              ------------------------
               IF ( DABS(INVM(I,J)).LT.SMALL ) THEN
                  INVM(I,J) = 0.0
               END IF
            END DO
            XRG(I) = XRG(I) - Y(I)*XRG(JSAL)
C           ------------------------
C           * ESTABILIDAD NUMERICA *
C           ------------------------
            IF ( DABS(XRG(I)).LT.SMALL ) THEN
               XRG(I) = 0.0
            END IF
C           ---------------------------------------
C           * VARIABLES BASICAS NO DE HOLGURA *
C           ---------------------------------------
            IF ( INDI(I).GT.0 ) THEN
               XM(INDI(I),INDJ(I)) = XRG(I)
            END IF
         END IF
      END DO
      EXITO1 = .FALSE.
C     ---------------------------------------------
C     * SE ACTUALIZAN LOS MULTIPLICADORES SIMPLEX *
C     ---------------------------------------------
      DO J = 1,ME
         ELAM = 0.0
C        ------------------------------
C        * J-ESIMO MULTIPLICADOR SIMPLEX *
C        ------------------------------
         ELAM = LAMDA(J) - RRAINVM(JSAL,J)
C        ------------------------
C        * ESTABILIDAD NUMERICA *
C        ------------------------
         IF ( DABS(ELAM).LT.SMALL ) THEN
            ELAM = 0.0
         END IF
C        ------------------------
C        * SI CAMBIO SU VALOR *
C        ------------------------
         IF ( DABS(ELAM-LAMDA(J)).GT.SMALL ) THEN
            LAMDA(J) = ELAM
            EXITO1 = .TRUE.
         END IF
      END DO
      DO J = ME+1,ME+NP
C        ------------------------------
C        * I-ESIMO MULTIPLICADOR SIMPLEX *
C        ------------------------------
         LAMDA(J) = LAMDA(J) - RRAINVM(JSAL,J)
C        ------------------------
C        * ESTABILIDAD NUMERICA *
C        ------------------------
         IF ( DABS(LAMDA(J)).LT.SMALL ) THEN
            LAMDA(J) = 0.0
```

```
      END IF
    END DO
    RETURN
    END


    SUBROUTINE DATOS( NERR,MBLK,NBLK,A )
C   ************************************************************
C   *                                                          *
C   *  OBJETIVO                                                *
C   *                                                          *
C   *  LEER DEL ARCHIVO DE DATOS DCW.DAT, LOS DATOS DEL PROBLEMA  *
C   *  QUE SE QUIERE RESOLVER. SI EXISTE ALGUN ERROR SIGNIFICATIVO *
C   *  EN ESTE ARCHIVO, EL PROGRAMA SUSPENDE SU EJECUCION.     *
C   *                                                          *
C   ************************************************************
C
C   ---------------------------------
C   * SE INCLUYE AREA DE PARAMETROS *
C   ---------------------------------
      INCLUDE 'AREAPL.FOR'
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      INTEGER  NBLK(MAXS),MBLK(MAXS),K1(8),K2(8),K3(8)
      REAL*8   A(MAXE+MAXH,MAXN),K4(8)
      LOGICAL  NERR
C
      COMMON / AE    / NE
      COMMON / AB    / NB
      COMMON / AH    / NH
      COMMON / VR    / VR     ( 1:MAXH+MAXE )
      COMMON / VS    / VS     ( 1:MAXH+MAXE )
      COMMON / CC    / CC     ( 1:MAXN )
      COMMON / ITMAX / ITMAX
C
C   ------------------------------------------------------------------
C   * SE LEEN NUMERO DE RESTRICCIONES (N), NUMERO DE VARIABLES (N), *
C   * NUMERO DE RESTRICCIONES DE ENLACE (NE), NUMERO DE SUBSISTEMAS *
C   * (NB), NUMERO MAXIMO DE ITERACIONES (ITMAX) Y OPCION DE LA IM- *
C   * PRESION (IMP).                                               *
C   ------------------------------------------------------------------
      READ(1,200)  N,NV,NE,NB,ITMAX
C   ---------------------------------------------
C   * NO SE ESPECIFICO EL NUMERO MAXIMO DE ITERACIONES *
C   ---------------------------------------------
      IF ( ITMAX.LE.0 ) THEN
C       ---------------------------------
C       * SE ASIGNA UN NUMERO ARBITRARIO *
C       ---------------------------------
        ITMAX = 3*(N+NV)
      END IF
C     ------------------------------------------------------
C     * SI EL PROBLEMA ES MUY GRANDE PARA EL PROGRAMA *
C     ------------------------------------------------------
      IF ( N.GT.MAXN .OR. NV.GT.MAXV ) THEN
C       ------------------------
C       * SE ESCRIBE UN MENSAJE *
C       ------------------------
        WRITE(2,201) MAXN,MAXV
```

```
      NERR = .FALSE.
    ELSE
C   ------------------------------------------------------------
C   * NO EXISTE ERROR EN LOS DEMAS DATOS DEL PROBLEMA *
C   ------------------------------------------------------------
      IF ((M.GT.0 .AND. NN.GT.0) .AND. (NE.GT.0 .AND. ND.GE.1)) THEN
C     ---------------------------------------------------------
C     * SE ESCRIBE EL NUMERO MAXIMO DE ITERACIONES *
C     ---------------------------------------------------------
        WRITE(3,202) M,NN,ITMAX
      ELSE
C       -------------------------------
C       * SE ESCRIBE UN MENSAJE DE ERROR *
C       -------------------------------
        WRITE(3,203)
        NERR = .FALSE.
      END IF
    END IF
    IF ( NERR ) THEN
C     ----------------------------
C     * SE LEE EL PRIMER REGISTRO *
C     ----------------------------
      CARD = 1
      K1(1) = 1
C     ------------------------------------------
C     * MIENTRAS NO SEA EL FIN DEL REGISTRO *
C     ------------------------------------------
      DO WHILE ( K1(1).NE.99 .AND. NERR )
        READ(1,304)  ( (K1(JK),K2(JK),K3(JK)),JK=1,3 )
        IF ( K1(1).NE.99 ) THEN
          DO J = 1,3
C           ------------------------------------------------------
C           * NO SE REBASA EL NUMERO DE SUBSISTEMAS DECLARADO *
C           ------------------------------------------------------
            IF ( K1(J).LE.NS ) THEN
C             ------------------------------------------------------------
C             * NUMERO DE RESTRICCIONES DEL J-ESIMO SUBSISTEMA *
C             ------------------------------------------------------------
              MBLK(K1(J)) = K2(J)
C             ------------------------------------------
C             * NUMERO DE VARIABLES DEL J-ESIMO SUBSISTEMA *
C             ------------------------------------------
              NBLK(K1(J)) = K3(J)
            ELSE
C             -------------------------------
C             * SE ESCRIBE UN MENSAJE DE ERROR *
C             -------------------------------
              WRITE(3,305) ND,K1(J)
              NERR = .FALSE.
            END IF
          END DO
        END IF
      END DO
C     ----------------------------
C     * SE LEE EL SEGUNDO REGISTRO *
C     ----------------------------
      CARD = 2
      K1(1) = 1
C     ------------------------------------------
C     * MIENTRAS NO SEA EL FIN DEL REGISTRO *
```

```
C     ------------------------------------------------------------------
      DO WHILE ( K1(1).NE.999 .AND. NERR )
         READ(1,206)   ( (K1(JK),K2(JK),K4(JK)),JK=1,8 )
         IF ( K1(1).NE.999 ) THEN
            DO J = 1,8
C           ------------------------------------------------------------
C           * NO SE REBASA EL NUMERO DE VARIABLES DECLARADO *
C           ------------------------------------------------------------
               IF ( K1(J).LE.NN ) THEN
C              ---------------------------------------------------------
C              * J-ESIMO ELEMENTO DE LA FUNCION OBJETIVO (VECTOR C) *
C              ---------------------------------------------------------
                  CC(K1(J)) = K4(J)
               ELSE
C                 ------------------------------
C                 * SE ESCRIBE UN MENSAJE DE ERROR *
C                 ------------------------------
                  WRITE(2,207) NN,K1(J)
                  EXITO = .FALSE.
               END IF
            END DO
         END IF
      END DO
C     ------------------------------
C     * SE LEE EL TERCER REGISTRO *
C     ------------------------------
      CARD = 3
      K1(1) = 1
C     ------------------------------------
C     * MIENTRAS NO SEA EL FIN DEL REGISTRO *
C     ------------------------------------
      DO WHILE ( K1(1).NE.999 .AND. NERR )
         READ(1,206)   ( (K1(JK),K2(JK),K4(JK)),JK=1,8 )
         IF ( K1(1).NE.999 ) THEN
            DO J = 1,8
C           ------------------------------------------------------------
C           * SI ES UN NUMERO DE RESTRICCION REPRESENTATIVO *
C           ------------------------------------------------------------
               IF ( K1(J).GT.0 ) THEN
C                 ------------------------------------------------------
C                 * SI SE TRATA DE UNA RESTRICCION DE ENLACE *
C                 ------------------------------------------------------
                  IF ( K1(J).LE.ME ) THEN
C                    ---------------------------------------------------
C                    * NO EXISTE ERROR EN EL SIGNO DE ESTA RESTRICCION *
C                    ---------------------------------------------------
                     IF ( K2(J).EQ.1 ) THEN
C                       ------------------------------------------------
C                       * SIGNO DE LA K1(J)-ESIMA RESTRICCION DE ENALACE *
C                       ------------------------------------------------
                        VS(K1(J)) = K2(J)
C                       ------------------------------------------------
C                       * LADO DERECHO DE LA K1(J)-ESIMA RESTRICCION *
C                       * DE ENALACE                                 *
C                       ------------------------------------------------
                        VB(K1(J)) = K4(J)
                     ELSE
C                       ------------------------------
C                       * SE ESCRIBE UN MENSAJE DE ERROR *
C                       ------------------------------
```

```
                        WRITE(2,208) K1(J),K2(J)
                        NERR = .FALSE.
                      END IF
                    ELSE
C                   ------------------------------------------
C                   * SI ESTE NUMERO NO REBASA EL NUMERO DE *
C                   * RESTRICCIONES DECLARADO.               *
C                   ------------------------------------------
                    IF ( K1(J).LE.M ) THEN
C                     ----------------------------------------------------
C                     * SI EL SIGNO DE LA RESTRICCION ES APROPIADO *
C                     ----------------------------------------------------
                      IF ( K2(J).EQ.0 .OR. K2(J).EQ.1 .OR. K2(J).
     >                   EQ.2 ) THEN
C                       ------------------------------------------
C                       * SIGNO DE LA K1(J)-ESIMA RESTRICCION *
C                       ------------------------------------------
                        VS(K1(J)) = K2(J)
C                       ------------------------------------------
C                       * LADO DERECHO DE LA K1(J)-ESIMA RESTRICCION *
C                       ------------------------------------------
                        VB(K1(J)) = K4(J)
                      ELSE
C                       ------------------------------------
C                       * SE ESCRIBE UN MENSAJE DE ERROR *
C                       ------------------------------------
                        WRITE(2,209) K1(J),K2(J)
                        NERR = .FALSE.
                      END IF
                    ELSE
C                     ------------------------------------
C                     * SE ESCRIBE UN MENSAJE DE ERROR *
C                     ------------------------------------
                      WRITE(2,210) M,K1(J)
                      NERR = .FALSE.
                    END IF
                  END IF
                END IF
              END DO
            END IF
          END DO
C         ------------------------------
C         * SE LEE EL CUARTO REGISTRO *
C         ------------------------------
          CARD = 4
C         ------------------------------------------
C         * NUMERO DE RENGLON DE LA MATRIZ A LEIDO *
C         ------------------------------------------
          NM = 1
          K1(1) = 1
C         ------------------------------------
C         * MIENTRAS NO SEA EL FIN DEL REGISTRO *
C         ------------------------------------
          DO WHILE ( K1(1).NE.999 .AND. NERR )
            READ(1,206)  ( (K1(JK),K2(JK),K4(JK)),JK=1,8 )
            IF ( K1(1).NE.999 ) THEN
C             ------------------------------------
C             * NO SE CONTINUA EN EL MISMO RENGLON *
C             ------------------------------------
              IF ( K4(1).NE.NM ) THEN
```

```
C                  ------------------------
C                  A CAMBIO DE RENGLON A
C                  ------------------------
                   NM = NM + 1
C                  ------------------------------------------
C                  A NO SE TRAIA DEL SIGUIENTE RENGLON A
C                  ------------------------------------------
                   IF ( MA(1).NE.NM ) THEN
C                     ---------------------------------
C                     A SE ESCRIBE UN MENSAJE DE ERROR A
C                     ---------------------------------
                      WRITE(2,210)
                      NERR = .FALSE.
                   END IF
                END IF
                IF ( NERR ) THEN
C                  ------------------------------------------------
C                  A NO SE REBASA EL NUMERO DE RENGLONES DECLARADO A
C                  ------------------------------------------------
                   IF ( MA(1).LE.N ) THEN
C                     ----------------------------------------
C                     A SE RECORREN LAS COLUMNAS DE ESTE RENGLON A
C                     ----------------------------------------
                      DO J = 2,8
C                        ------------------------------------
C                        A SI EXISTE ELEMENTO EN ESTA COLUMNA A
C                        ------------------------------------
                         IF ( MI(J).GT.0 ) THEN
C                           ------------------------------------------------------
C                           A SI EL NUMERO DE COLUMNA NO ES MAYOR QUE EL A
C                           A NUMERO DE COLUMNAS DECLARADO.                      A
C                           ------------------------------------------------------
                            IF ( MI(J).LE.NM ) THEN
C                              ---------------------------
C                              A ELEMENTO EN LA MATRIZ A A
C                              ---------------------------
                               A(NM,MI(J)) = MA(J)
                            ELSE
C                              ---------------------------------
C                              A SE ESCRIBE UN MENSAJE DE ERROR A
C                              ---------------------------------
                               WRITE(2,211) MI(J),NM
                               NERR = .FALSE.
                            END IF
                         END IF
                      END DO
                   ELSE
C                     ---------------------------------
C                     A SE ESCRIBE UN MENSAJE DE ERROR A
C                     ---------------------------------
                      WRITE(2,212) MA(1),N
                      NERR = .FALSE.
                   END IF
                END IF
             END IF
          END DO
       END IF
       RETURN
200    FORMAT(3110)
201    FORMAT(5X,'EL PROBLEMA ES MAS GRANDE QUE PUEDE RESOLVER EL PROBLEMA E
```

```
     >E DE ',/,5X,I5,' RESTRICCIONES Y ',I5,' VARIABLES'
      FORMAT(5X,'NUMERO DE RESTRICCIONES : ',I5,/,5X,'NUMERO DE VARIABLE
     >S : ',I5,/,5X,'NUMERO MAXIMO DE ITERACIONES : ',I5,/)
      FORMAT(5X,'LOS DATOS QUE DECLARASTE DEL PROBLEMA NO SON NORMALES.
     >REVISAR EL ARCHIVO DE DATOS')
      FORMAT(I2,I4,I1)
      FORMAT(5X,'SE DECLARARON ',I5,' SUBSISTEMAS. NO ES CORRECTO QUE EN
     > EL ARCHIVO EXISTA EL SUBSISTEMA ',I5)
      FORMAT(2(I2,11,F6.0))
      FORMAT(5X,'SE DECLARO UN NUMERO DE VARIABLES ',I5,' EL VECTOR C, N
     >O PUEDE CONTENER EL ELEMENTO ',I5)
      FORMAT(5X,'LA RESTRICCION ',I5,' QUE CORRESPONDE A UNA RESTRICCION
     > DE ENLACE',/,5X,'DEBE TENER COMO SIGNO MENOR O IGUAL (-). ',I5, '
     >NO ES CORRECTO')
      FORMAT(5X,'EL SIGNO DE LA RESTRICCION ',I5,' NO ES CORRECTO. ',I5,
     >' NO TIENE SIGNIFICADO EN EL PROGRAMA')
      FORMAT(5X,'SE DECLARO UN NUMERO DE RESTRICCIONES ',I5,' EL VECTOR
     >B, NO PUEDE CONTENER EL ELEMENTO ',I5)
      FORMAT(5X,'EN LA MATRIZ A NO PUEDE EXISTIR LA COLUMNA ',I5,'. YA
     >QUE ',I5,' FUE EL NUMERO DE VARIABLES DECLARADO')
      FORMAT(5X,'EN LA MATRIZ A NO PUEDE EXISTIR EL RENGLON ',I5,'. YA
     >QUE ',I5,' FUE EL NUMERO DE RESTRICCIONES DECLARADO')
      FORMAT(5X,'ES IMPOSIBLE QUE LOS RENGLONES DE LA MATRIZ A, ESTEN O
     >RDENADOS. REVISAR EL ARCHIVO DE DATOS')
      END


      SUBROUTINE MDANF( IRES,ORIN,XF )
C     ********************************************************************
C     *                                                                  *
C     *  OBJETIVO                                                        *
C     *                                                                  *
C     *  RESOLVER EL PROBLEMA, MEDIANTE EL METODO DE DESCOMPOSICION      *
C     *  DE DANTZIG-WOLFE.                                               *
C     *                                                                  *
C     ********************************************************************
C
C     ----------------------------------
C     * SE INCLUYE AREA DE PARAMETROS *
C     ----------------------------------
      INCLUDE 'ARPARL.FOR'
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      INTEGER  PR(MAXS),JX(MAXP+1)
      REAL*8   VX(MAXN),XC(MAXP*MP),Y(MAXE+MAXS),XF(MAXN)
      LOGICAL  EXITO, EXIT01
C
      COMMON / IX     / IX       ( 1:MAXS+1 )
      COMMON / MP     / MP
      COMMON / IND    / IND      ( 1:50 )
      COMMON / BIG    / BIG
      COMMON / ITMAX  / ITMAX
      COMMON / XB     / XB       ( 1:MAXS,1:MAXPP )
C
C     ----------------------------------------------------------------
C     * SE INICIALIZA EL VALOR DE LAS VARIABLES EN EL PROBLEMA *
C     ----------------------------------------------------------------
      DO J = 1,JX(MP+1)-1
         XF(J) = 0.0
```

```
      END DO
C     ------------------------------------------------------
C     | NUMERO DE ITERACIONES DEL PROBLEMA MAESTRO |
C     ------------------------------------------------------
      ITM = 1
C     ------------------------------------------------------
C     | SE FORMA LA PRIMERA BASE DEL PROBLEMA MAESTRO |
C     ------------------------------------------------------
      CALL BASINI
C     ------------------------------------------------------
C     | APUNTADORES PARA ENCONTRAR LA SOLUCION DEL PROBLEMA |
C     ------------------------------------------------------
      KBX(1) = 1
      JXX(1) = 1
      IRES = 0
C     ------------------------------
C     | VALOR DE LA FUNCION OBJETIVO |
C     ------------------------------
      OBJM = 0.0
C     ------------------------------------------------------
C     | APUNTADOR PARA IDENTIFICAR LA SOLUCION DE ACUERDO AL |
C     | NUMERO DE ITERACIONES MAESTRAS                       |
C     ------------------------------------------------------
      DO I = 1,ITMAX
         IND(I) = 0
      END DO
      EXITO = .TRUE.
      EXITO1 = .TRUE.
C     ------------------------------------------------------
C     | MIENTRAS NO SE ENCUENTRE LA SOLUCION AL PROBLEMA |
C     ------------------------------------------------------
      DO WHILE ( EXITO .AND. ITM.LE.ITMAX )
         R = BIG
C        ------------------------------------------------------------------
C        | SE RESUELVEN LOS NR SUBPROBLEMAS, CON SU RESPECTIVA FUNCION    |
C        | OBJETIVO Y SE ELIGE LA SOLUCION DEL QUE PROPORCIONE EL MEJOR   |
C        | INCREMENTO EN EL OBJETIVO DEL PROBLEMA MAESTRO                 |
C        ------------------------------------------------------------------
         CALL SOLSUB( R,EXITO,EXITO1,IRES,RE,KBIN,V)
C        ------------------------------------------------------
C        | SI NO EXISTE PROBLEMA EN LA SOLUCION DE LOS SUBPROBLEMAS |
C        ------------------------------------------------------
         IF ( EXITO ) THEN
C           ------------------------------------------------
C           | YA NO PUEDE EXISTIR INCREMENTO EN EL VALOR DE LA |
C           | FUNCION.                                         |
C           ------------------------------------------------
            IF ( R.GE.0.0 ) THEN
C              ------------------------
C              | LA SOLUCION ES OPTIMA |
C              ------------------------
               IRES = 1
               EXITO = .FALSE.
            ELSE
C              ------------------------------------------------
C              | SE GUARDA LA SOLUCION DEL SUBPROBLEMA QUE DETERMINA |
C              | MAYOR INCREMENTO EN LA FUNCION DEL PROBLEMA MAESTRO |
C              ------------------------------------------------
               DO I = 1,JXX(KBIN+1) - JXX(KBIN)
                  VX(JXX(ITM)+I-1) = VX(JXX(KBIN)+I-1)
```

```
            END DO
C           ------------------------------------------------------------
C           A APUNTADORES PARA IDENTIFICAR LA SOLUCION ENCONTRADA A
C           A DEL SUBPROBLEMA GANADOR                                  A
C           ------------------------------------------------------------
            IXX(ITH+1) = IXX(ITH) + IX(KBIN(1) - IX(KBIN)
            KBX(ITH) = KBIN
C           ------------------------------------------------------------
C           A SE CALCULA LA COLUMNA DEL PROBLEMA MAESTRO ASOCIADA A LA A
C           A VARIABLE QUE ENTRARA A LA BASE                           A
C           ------------------------------------------------------------
            CALL NUEVAR( KBIN,VX,Y )
C           ------------------------------------------------------------
C           A SE DETERMINA LA VARIABLE QUE SALDRA DE LA BASE           A
C           ------------------------------------------------------------
            CALL VARISA( JSAL,Y )
C           ----------------------------
C           A SI NO EXISTE TAL VARIABLE A
C           ----------------------------
            IF ( JSAL.EQ.0 ) THEN
C              ------------------------------
C              A EL PROBLEMA ES NO ACOTADO A
C              ------------------------------
               IRES = 3
               EXITO = .FALSE.
            ELSE
C              ----------------------------
C              A SE REALIZA EL CAMBIO DE BASE A
C              ----------------------------
               CALL CAMBAS( EXITO1,JSAL,Y,RR,KBIN,ITH,OBJM )
C              ------------------------------------------------------
C              A SE INCREMENTA EL NUMERO DE ITERACIONES               A
C              ------------------------------------------------------
               ITH = ITH + 1
            END IF
         END IF
      END IF
   END DO
C  ------------------------------------------------------------
C  A SI SE ENCONTRO UNA SOLUCION OPTIMA AL PROBLEMA           A
C  ------------------------------------------------------------
   IF ( IRES.EQ.1 ) THEN
C     ----------------------------
C     A SE DETERMINA LA SOLUCION A
C     ----------------------------
      DO I = 1,ITH-1
         JJ = JX(KBX(I))
         DO J = JXX(I),JXX(I+1)-1
            XE(JJ) = XE(JJ) + XM(KBX(I),IND(I))*AXX(J)
            JJ = JJ + 1
         END DO
      END DO
      OBJM = -OBJM
   ELSE
      IF ( ITH.GT.ITMAX ) THEN
C        ------------------------------------------------------
C        A EL NUMERO MAXIMO DE ITERACIONES FUE REBASADO A
C        ------------------------------------------------------
         IRES = 4
      END IF
```

```
                    EXITO1 = .FALSE.
                  END IF
                END IF
              END IF
            END IF
            J = J + 1
          END DO
        END IF
        RETURN
 9000   FORMAT(2X,'ERROR DE ',F16.8,' ENCONTRADO EN B-BLACK-AX DE LA RESTR
     >ICCION ',I6)
 9004   FORMAT(2X,'ERROR RELATIVO DE ',F16.8,' ENCONTRADO EN B-BLACK-AX DE
     >LA RESTRICCION ',I6/1H,' EL ERROR ABSOLUTO ES ',F16.8)
 9008   FORMAT(2X,'ERROR DE ',F16.8,' ENCONTRADO EN YA-C DE LA VARIABLE BA
     >SICA ',I6)
 9012   FORMAT(1H0,'ERROR RELATIVO DE ',F16.8,' ENCONTRADO EN YAC DE LA VA
     >RIABLE BASICA ',I6/1H,' EL ERROR ABSOLUTO ES ',F16.8)
        END


        SUBROUTINE REVERT
C       ******************************************************************
C       *                                                                *
C       *  OBJETIVO                                                      *
C       *                                                                *
C       *  DESPUES DE QUE LA SUBRUTINA CHACC VERIFICA LA EXACTITUD EN    *
C       *  LA SOLUCION ENCONTRADA, Y ESTA NO ES CORRECTA (ISTATC = 7);   *
C       *  ESTA SUBRUTINA INVIERTE DIRECTAMENTE LA BASE ACTUAL  DE LA    *
C       *  FORMA MAS PRECISA POSIBLE Y REINICIA LA SUBRUTINA  DOANLP     *
C       *  CON UNA NUEVA SOLUCION BASICA.                                *
C       *                                                                *
C       ******************************************************************
C
C       --------------------------------
C       * SE INCLUYE AREA DE PARAMETROS *
C       --------------------------------
        INCLUDE 'ARPARL.FOR'
C
        IMPLICIT REAL*8 (A-H,O-Z)
        REAL A8 INV
        INTEGER XBASIC,YBASIC, SIZE, SIZE1
        LOGICAL ENTRO
C
        COMMON / B       / B       ( 1:MAXB )
        COMMON / BOUND   / BOUND   ( 1:MAXZ )
        COMMON / C       / C       ( 1:MAXN )
        COMMON / DRIVER  / DRIVER
        COMMON / GR      / GR      ( 1:MXSIZE )
        COMMON / ISBND   / ISBND
        COMMON / INBASE  / INBASE  ( 1:MAXN )
        COMMON / INREV   / INREV
        COMMON / INV     / INV     ( 1:MXSIZ1,1:MXSIZE )
        COMMON / IR      / IR
        COMMON / ITR     / ITR
        COMMON / ISCTT   / ISCTT   ( 1:MAXN )
        COMMON / N       / N
        COMMON / NEGINV  / NEGINV
        COMMON / NEWX    / NEWX
        COMMON / NEWY    / NEWY
        COMMON / NUMSLK  / NUMSLK
        COMMON / OBJ     / OBJ
```

```
      COMMON / C       / C       ( 1:MAXN )
      COMMON / SIZE    / SIZE
      COMMON / SMALL   / SMALL
      COMMON / TOL     / TOL     ( 1:9 )
      COMMON / X       / X       ( 1:MAXN )
      COMMON / XBASIS  / XBASIS  ( 1:MXSIZE )
      COMMON / XR      / XR      ( 1:MXSIZE )
      COMMON / YBASIS  / YBASIS  ( 1:MXSIZE )
      COMMON / YR      / YR      ( 1:MXSIZE )
C
C     ------------------------------------------------
C     | SE INCREMENTA EL NUMERO DE REINVERSIONES |
C     ------------------------------------------------
      IR = IR + 1
C     ------------------------
C     | NUMERO DE ITERACIONES |
C     ------------------------
      ITHOLD = ITR
C     ------------------------------------------------
C     | SE CALCULA LA INVERSA DE LA BASE ACTUAL |
C     ------------------------------------------------
      INREV = 1
C     ------------------------
C     | SE ASIGNA UN VALOR PEQUENO |
C     ------------------------
      HOLD = SMALL
      SMALL = 0.0
C     ------------------------------------------------------------
C     | TOLERANCIA QUE AYUDA A DETERMINAR SI AL FINAL DE LA |
C     | REINVERSION LA BASE OBTENIDA ES NO SINGULAR. |
C     ------------------------------------------------------------
      TOL8 = TOL ( 8 )
C     ------------------------------------------------------------------
C     | SI UNA VARIABLE DE HOLGURA ERA BASICA ANTES DE LA REINVERSION |
C     | SE ASEGURA QUE SU LUGAR (EN XBASIS), SEA POSICIONADO EN EL |
C     | RENGLON QUE CORRESPONDE A SU COLUMNA (EN YBASIS). |
C     ------------------------------------------------------------------
      DO  K = 1, SIZE
          J = N + K
          DO WHILE ( J.GT.N )
C             ------------------------------------
C             | SI UNA VARIABLE DE HOLGURA ERA BASICA |
C             ------------------------------------
              IF ( XBASIS(K) .GT. N ) THEN
C                 ------------------------------------------------------
C                 | YA QUE XBASIS(K) ES MAYOR QUE N, NO PUEDEN EXISTIR MAS |
C                 | DE M RESTRICCIONES. |
C                 ------------------------------------------------------
                  I = XBASIS ( K ) - N
C                 ------------------------
C                 | RESTRICCION ASOCIADA |
C                 ------------------------
                  L = ISEED ( I )
C                 ------------------------------------------------
C                 | SI NO CORRESPONDE A LA VARIABLE QUE ESTA EN LA BASE |
C                 ------------------------------------------------
                  IF ( K .NE. L ) THEN
C                     ------------------------------------------
C                     | REPOSICION DE LA VARIABLE EN LA NUEVA BASE |
C                     ------------------------------------------
```

```
            XBASIS ( K ) = XBASIS ( L )
            XBASIS ( L ) = I + N
C           ------------------------------------------
C           A QUE CLASE DE VARIABLE ESTA EN LA BASE A
C           ------------------------------------------
            I = XBASIS ( K )
          ELSE
            J = 0
          END IF
        ELSE
          J = 0
        END IF
      END DO
    END DO
C   ------------------------------------------------------------
C   A LA MATRIZ INVERSA ES REEMPLAZADA POR UNA MATRIZ IDENTIDAD A
C   ------------------------------------------------------------
    DO  K = 1 , SIZE
C     ------------------------------------
C     A SI ES UNA VARIABLE BASICA NO DE HOLGURA A
C     ------------------------------------
      IF ( XBASIS(K) .LE. M ) THEN
C       ------------------------------------------------------------------
C       A SE MULTIPLICA POR -1 PARA INDICAR QUE VARIABLES NO HAN A
C       A SIDO INTRODUCIDAS YA A LA BASE.                          A
C       ------------------------------------------------------------------
        XBASIS(K) = - XBASIS(K)
      END IF
C     ------------------------------
C     A SE FORMA LA MATRIZ IDENTIDAD A
C     ------------------------------
      DO  L = 1, SIZE
        INV ( K, L ) = 0.0
      END DO
      INV ( K, K ) = 1.0
    END DO
C   ------------------------------------------------------------
C   A TODAS LAS VARIABLES NO DE HOLGURA SON INDICADAS COMO A
C   A NO BASICAS.                                          A
C   ------------------------------------------------------------
    DO  J = 1, M
C     ------------------------------------------------------
C     A SI NO SE TRATA DE UNA VARIABLE QUE ALCANZO SU CUTA A
C     A SUPERIOR.                                          A
C     ------------------------------------------------------
      IF ( INBASE(J) .NE. -1 ) THEN
C       ------------------------------
C       A YA NO ES BASICA ESTA VARIABLE A
C       ------------------------------
        INBASE(J) = 0
      END IF
    END DO
C   ------------------------------------------------------------
C   A SE INTRODUCE CADA VARIABLE (NO DE HOLGURA). NEWX, A LA A
C   A BASE ACTUAL.                                          A
C   ------------------------------------------------------------
    DO  K = 1, SIZE
C     ------------------------------
C     A VARIABLE QUE SERA INTRODUCIDA A
C     ------------------------------
```

```
      NEWX = XBASIS ( K )
C     ------------------------------
C     A SI NO ES UNA VARIABLE DE HOLGURA A
C     ------------------------------
      IF ( NEWX .GT. 0 .AND. NEWX .LE. P ) THEN
C        ------------------------------------------------------
C        A RENGLON DE LA INVERSA ASOCIADO CON UNA VARIABLE QUE A
C        A SALE DE LA BASE.                                    A
C        ------------------------------------------------------
         NEWY = 1
         IHOLD = 1
         DO WHILE ( NEWY.NE.0 .AND. IHOLD.NE.0 )
C           ----------------------------
C           A SE ACTUALIZA EL VECTOR GR A
C           ----------------------------
            CALL NEWVEC
C           ----------------------------
C           A SE REINICIALIZA NEWY A
C           ----------------------------
            NEWY = 0
C           ------------------------------------------------------
C           A SE BUSCA EL RENGLON QUE CONTENDRA AL PIVOTE (ESTE NO A
C           A PODRA HABER SIDO YA PIVOTE Y DEBE SER EL DE MENOR    A
C           A DIFERENCIA ABSOLUTA DE UNO.                          A
C           ------------------------------------------------------
            DO  KK = 1, SIZE
C              ------------------------------
C              A SI ES UNA VARIABLE NO BASICA A
C              ------------------------------
               IF ( XBASIS(KK) .LE. 0 ) THEN
C                 ------------------------------------------------
C                 A VALOR ABSOLUTO DEL POSIBLE ELEMENTO PIVOTE A
C                 ------------------------------------------------
                  ABDIF = DABS ( GR(KK) )
C                 ------------------------------------
C                 A ES CANDIDATO Y LA VARIABLE PUEDE SER A
C                 A INTRODUCIDA.                        A
C                 ------------------------------------
                  IF ( ABDIF .GE . IHLD ) THEN
C                    ------------------------------------------
C                    A NO EXISTE RENGLON PARA DEPOSITAR ESTA A
C                    A VARIABLE.                             A
C                    ------------------------------------------
                     IF ( NEWY .EQ. 0 ) THEN
C                       ------------------------------------
C                       A DIFERENCIA ABSOLUTA ENTRE EL POSIBLE A
C                       A PIVOTE Y UNO.                       A
C                       ------------------------------------
                        BEST = DABS ( 1.0 - ABDIF )
C                       ------------------------------
C                       A RENGLON PARA LA NUEVA VARIABLE A
C                       ------------------------------
                        NEWY = KK
                     ELSE
C                       ------------------------------------
C                       A DIFERENCIA ABSOLUTA ENTRE EL POSIBLE A
C                       A PIVOTE Y UNO.                       A
C                       ------------------------------------
                        ABDIF = DABS ( 1.0 - ABDIF )
C                       ----------------------------
```

```
                        A SI ES MENOR LA DIFERENCIA A
                        ----------------------------
                        IF ( ABDIF .LT. BEST ) THEN
                        ---------------------------
C                          A NUEVA DIFERENCIA PARA ELEGIR EL A
C                          A PIVOTE.                          A
C                          ---------------------------------
                           BEST = ABDIF
C                          ---------------------------------
C                          A RENGLON PARA DEPOSITAR LA NUEVA A
C                          A VARIABLE.                       A
C                          ---------------------------------
                           NEWY = KK
                        END IF
                     END IF
                  END IF
               END DO
C              ------ ----- ------ -- ------- - -----
C              A SI EXISTE RENGLON PARA DEPOSITAR UNA NUEVA A
C              A VARIABLE.                                  A
C              ---------------------------------------------
               IF ( NEWY .NE. 0 ) THEN
C              ------------------------------------------------------
C                 A SE BUSCA LA SIGUIENTE VARIABLE A ENTRAR A LA BASE A
C                 A DENTRO DE XBASIS.                                 A
C                 -------------------------------------------------------
                  IHOLD = -XBASIS ( NEWY )
C                 ------------------------
C                 A SI NEWY PERTENECE AL MISMO RENGLON QUE NEWX A
C                 ----------------------------------------------
                  IF ( IHOLD .EQ. NEWX ) THEN
C                 ---------------------------
C                    A NO SE BUSCA OTRA VARIABLE, NEWX ES LA VARIABLE A
C                    A A ENTRAR A LA BASE.                            A
C                    ---------------------------------------------------
                     IHOLD = 0
                  END IF
C                 ------------------------------------------------------
C                 A SE REALIZA EL CAMBIO DE BASE INTRODUCIENDO A NEWX A
C                 ------------------------------------------------------
                  CALL CHBAS
C                 ----------
C                 A NUEVA VARIABLE BASICA A
C                 -------------------------
                  XBASIS ( NEWY ) = NEWX
C                 ----------------------
C                 A SI EXISTE OTRA CANDIDATA A ENTRAR A LA BASE A
C                 ----------------------------------------------
                  IF ( IHOLD .NE. 0 ) THEN
C                 ------------------------
C                    A VARIABLE QUE ENTRA A LA BASE A
C                    -------------------------------
                     NEWX   IHOLD
                  END IF
               END IF
            END DO
         END IF
      END DO
C     ---------------------------------------
```

```
C     A NO EXISTEN VARIABLES DE HOLGURA EN LA BASE A
C     ----------------------------------------------
      NUMSLK = 0
C     ----------------------------------------------------------------
C     A SE MUESTRAN LAS VARIABLES ORIGINALES AHORA EN LA BASE A
C     ----------------------------------------------------------------
      DO  K = 1, SIZE
C        ------------------------
C        A ESTA O NO EN LA BASE A
C        ------------------------
         J = XBASIS ( K )
C        ------------------------
C        A SI ES UNA VARIABLE BASICA A
C        ------------------------
         IF ( J .GT. 0 ) THEN
C           ---------------------------------------------
C           A SI ES UNA VARIABLE BASICA NO DE HOLGURA A
C           ---------------------------------------------
            IF ( J .LE. N ) THEN
C              ---------------------
C              A LUGAR DENTRO DE LA BASE A
C              ---------------------
               INBASE(J) = K
            ELSE
C              ---------------------------------------------------
C              A SE INCREMENTA EL NUMERO DE VARIABLES DE HOLGURA A
C              A EN LA BASE.                                     A
C              ---------------------------------------------------
               NUMSLK = NUMSLK + 1
            END IF
         ELSE
C           ---------------------------------------------
C           A SE AGREGA UNA VARIABLE DE HOLGURA A LA BASE A
C           ---------------------------------------------
            I = YBASIS ( K )
            XBASIS ( K ) = N + I
            NUMSLK = NUMSLK + 1
         END IF
      END DO
C     ---------------
C     A VALOR PEQUENO A
C     ---------------
      SMALL = HOLD
C     ----------------------------------------------------------------
C     A SE INICIALIZAN XR (VALOR DE LAS VARIABLES PRIMALES) Y A
C     A YR (VALOR DE LAS VARIABLES DUALES).                         A
C     ----------------------------------------------------------------
      DO  K = 1, SIZE
         XR ( K ) = 0.0
         YR ( K ) = 0.0
      END DO
C     ----------------------
C     A SE CALCULAN YR Y XR A
C     ----------------------
      DO  K = 1, SIZE
C        ---------------------------------------
C        A RESTRICCION ACTUAL EN LA INVERSA A
C        ---------------------------------------
         I = YBASIS( K )
C        ---------------
```

```
C        A VARIABLE BASICA A
C        ------------------
         J = XBASIS ( I )
C        ------------------------------------------------------------
C        A SE INICIALIZA AUXILIAR PARA LOS ELEMENTOS DEL VECTOR C A
C        ------------------------------------------------------------
         TC = 0.0
C        ------------------------------------
C        A SI ES UNA VARIABLE BASICA NO DE HOLGURA A
C        ------------------------------------
         IF ( J .LE. N ) THEN
C           ---------------------------------------------------------
C           A VALOR DE UN COEFICIENTE DE LA FUCION OBJETIVO (C) A
C           ---------------------------------------------------------
            TC = C ( J )
         END IF
C        ------------------------------------
C        A VALOR DE UN TERMINO INDEPENDIENTE (B) A
C        ------------------------------------
         TB = B ( I )
C        ------------------------------------------------------------
C        A SI EXISTEN VARIABLES QUE ALCANZARON SU COTA SUPERIOR SE A
C        A ACTUALIZA TB.                                           A
C        ------------------------------------------------------------
         DO JJ = 1, N
C           ---------------------------------------------------------
C           A SI EXISTE UNA VARIABLE QUE ALCANZO SU COTA SUPERIOR A
C           ---------------------------------------------------------
            IF ( INDBAS(JJ) .EQ. -1 ) THEN
               TB = TB - BOUND ( JJ ) * A (I,JJ)
            END IF
         END DO
C        ------------------
C        A CALCULO DE YR Y XR A
C        ------------------
         DO  L = 1, SIZE
            XR ( L ) = XR ( L ) + TRAINV(L,K)
            YR ( L ) = YR ( L ) + TCAINV(K,L)
C           ------------------------------------
C           A SI YR(L) ES UN VALOR MUY PEQUENO A
C           ------------------------------------
            IF ( DABS(YR(L)) .LE. SMALL ) THEN
C              -------------------------
C              # ESTABILIDAD NUMERICA A
C              -------------------------
               YR(L) = 0.0
            END IF
C           ------------------------------------
C           A SI XR(L) ES UN VALOR MUY PEQUENO A
C           ------------------------------------
            IF ( DABS(XR(L)) .LE. SMALL ) THEN
C              -------------------------
C              A ESTABILIDAD NUMERICA A
C              -------------------------
               XR(L) = 0.0
            END IF
         END DO
      END DO
C     ------------------------------------------------
C     A SE INICIALIZA EL RENGLON DE LA INVERSA INDADECUADO A
```

```
      NEGINV = 0
      T = 0.0
C     ------------------------------------------------
C     * POSIBLE EXISTENCIA DE INFACTIBILIDAD *
C     ------------------------------------------------
      DO  K = 1, SIZE
C        ---------------------------
C        * VALOR DE LA VARIABLE PRIMAL *
C        ---------------------------
         XRK = XR ( K )
C        ------------------
C        * VARIABLE BASICA *
C        ------------------
         J = XBAS(K)
C        ------------------------------------------
C        * SI ES UNA VARIABLE BASICA NO DE HOLGURA *
C        ------------------------------------------
         IF ( J .LE. N ) THEN
C           ------------------------------------------------------------
C           * SI EXISTEN VARIABLES CON COTAS SUPERIORES Y COTAS *
C           * COTAS NO SON IGUAL A MENOS UNO.                   *
C           ------------------------------------------------------------
            IF ( ISBND.NE.0 .AND. BOUND(J).NE.-1 ) THEN
C              ------------------------------------------------------------
C              * SI EL VALOR DE LA VARIABLE ES MAS GRANDE QUE SU COTA *
C              ------------------------------------------------------------
               IF ( XRK .GT. BOUND (J) ) THEN
C                 ---------------------------
C                 * NUEVO VALOR DE LA VARIABLE *
C                 ---------------------------
                  XRK = BOUND(J) - XR(K)
               END IF
            END IF
C           ------------------------
C           * SI ES UN VALOR NEGATIVO *
C           ------------------------
            IF ( XRK .LT. T ) THEN
C              ---------------------
C              * SE LE CAMBIA EL SIGNO *
C              ---------------------
               T = -1.0*ABS(XRK)
C                 ------------------
C                 * RENGLON INFACTIBLE *
C                 ------------------
               NEGINV = K
C              ------------------------------------------------
C              * DEBE INCREMENTARSE EL VALOR DE ESTA VARIABLE *
C              ------------------------------------------------
               DRIVER = 1.0
C              ------------------------------
C              * SI SU VALOR ES MAYOR QUE CERO *
C              ------------------------------
               IF ( XR(K) .GT. 0.0 ) THEN
C                 ------------------------------------------------------
C                 * REDUCIENDO SU VALOR SE ELIMINA INFACTIBILIDAD *
C                 ------------------------------------------------------
                  DRIVER = -1.0
               END IF
            END IF
```

```
      ELSE
C     ------------------------------------------------------------
C     A COMO J ES MAYOR QUE N, NO PUEDEN EXISTIR MAS DE N      A
C     A RESTRICCIONES.                                         A
C     ------------------------------------------------------------
      I = J - N
      ENTRO = S(I).NE.0.0 .AND. XMEAS(I).GE.T .OR. S(I).EQ.0.0
     .AND. DABS(XBR(A(I,1))).GE.T
C     ------------------------------
C     A SI NO EXISTE FACTIBILIDAD A
C     ------------------------------
      IF ( .NOT.ENTRO ) THEN
C        ---------------------
C        A SE LE CAMBIA EL SIGNO A
C        ---------------------
         T = -1.0*DABS(XBR(
C        ---------------------
C        A RENGLON INFACTIBLE A
C        ---------------------
         NEGINV = K
C        ------------------------------------------------------
C        A DEBE INCREMENTARSE EL VALOR DE ESTA VARIABLE A      A
C        ------------------------------------------------------
         DRIVER = 1.0
C        ---------------------
C        A SI SU VALOR ES MAYOR QUE CERO A
C        ---------------------
         IF ( XR(K) .GT. 0.0 ) THEN
C           ------------------------------------------------------
C           A REDUCIENDO SU VALOR SE ELIMINA INFACTIBILIDAD A
C           ------------------------------------------------------
            DRIVER = -1.0
         END IF
      END IF
   END IF
END DO
C  ---------------------------------------------
C  A SI EXISTEN VARIABLES DE HOLGURA EN LA BASE A
C  ---------------------------------------------
IF ( NUMSLK .GE. 1 ) THEN
C  ------------------------------------------------------
C     A SE REMUEVE UNA VARIABLE DE HOLGURA EN LA INVERSA A
C  ------------------------------------------------------
   CALL REDUCE
END IF
C  ---------------------------------------
C  A SE ACTUALIZAN LOS VECTORES  Y, X Y SLACK A
C  ---------------------------------------
CALL CHSLCK
C  ---------------------
C  A SE PRUEBA OPTIMALIDAD A
C  ---------------------
CALL ISOPT
C  ---------------------
C  A NUMERO DE ITERACIONES A
C  ---------------------
ITR = ITHOLD
C  ---------------------------------
C  A INDICADOR VUELTO A SU VALOR DE ENTRADA A
C  ---------------------------------
```

```
      INREV = 0
      OBJ = 0.0
C     ------------------------------------------
C     * SE ACTUALIZA EL VALOR DE LA FUNCION OBJETIVO *
C     ------------------------------------------
      DO J = 1, N
C        ------------------------------------
C        * SI SE TRATA DE UNA VARIABLE BASICA *
C        ------------------------------------
         IF ( INDASE(J) .NE. 0 ) THEN
C           ------------------------------
C           * VALOR DE LA FUNCION OBJETIVO *
C           ------------------------------
            OBJ = OBJ + X(J)*C(J)
         END IF
      END DO
C     -------------------------
C     * SE ESCRIBE UN MENSAJE *
C     -------------------------
      WRITE (2,9000) ITR
      RETURN
9000  FORMAT(1H,20X,'SE REALIZO UNA REINVERSION EN LA ITERACION ',I6)
      END
```

```
      ELSE IF ( IRES.EQ.4 ) THEN
C     ---------------------------
C        A SE ESCRIBE UN MENSAJE A
C     ---------------------------
         WRITE(2,105)
      END IF
      RETURN
100   FORMAT(5X,'LA SOLUCION OPTIMA ES : '/)
101   FORMAT(10X,'X(',I3,') = ',E12.5)
102   FORMAT(/,5X,'EL VALOR DE LA FUNCION OBJETIVO ES : ',E12.5)
103   FORMAT(5X,'NO FUE ENCONTRADA SOLUCION OPTIMA FACTIBLE PARA EL PROB
     >LEMA')
104   FORMAT(5X,'NO FUE ENCONTRADA SOLUCION OPTIMA ACOTADA PARA EL PROBL
     >EMA')
105   FORMAT(5X,'EL NUMERO MAXIMO DE ITERACIONES FUE REBASADO')
      END
```

```
      SUBROUTINE PL( RENI,COLJ,VB,NA,VC,NR,NV,VS,
     >                 ITMX,REMX,VX,FOBJ,ESTADO )
C     ****************************************************************
C     *                                                              *
C     *  OBJETIVO                                                    *
C     *                                                              *
C     *  LLAMAR A ODANLP PARA RESOLVER EL PROBLEMA DE PL. SI EXISTE  *
C     *  UNA SOLUCION PARA EL PROBLEMA (OPTIMA, NO ACOTADA O INFAC-  *
C     *  TIBLE), LA RUTINA CHACC VERIFICA SU EXACTITUD: SI ESTA NO   *
C     *  ES ADECUADA SE REALIZA UNA REINVERSION.                     *
C     *                                                              *
C     ****************************************************************
C
C     --------------------------------
C     * SE INCLUYE AREA DE PARAMETROS *
C     --------------------------------
      INCLUDE 'ARPAPL.FOR'
      IMPLICIT REAL*8 (A-H,O-Z)
C
      REAL*8 NA(MAXA),VB(MAXM),VC(MAXN),VX(MAXN)
      REAL*8 VS(MAXM)
      INTEGER COLJ(MAXA),RENI(MAXM+1),ESTADO,IREN(MAXM+1),REMX
C
      COMMON / ISTATE / ISTATE
      COMMON / ITR    / ITR
      COMMON / INREV  / INREV
      COMMON / IR     / IR
C
C     ------------------------------------------------------------
C     * ARREGLO QUE INDICA EL INICIO DE UN RENGLON DE LA MATRIZ A *
C     * EN EL ARREGLO COMPACTADO AA.                              *
C     ------------------------------------------------------------
      IREN(1) = RENI(1)/RENI(1)
      DO I = 2,NR+1
         IREN(I) = RENI(I) - RENI(I-1) + IREN(I-1)
      END DO
C     --------------------
C     * DATOS DEL PROBLEMA *
C     --------------------
      CALL DATENT( NR,NV,ITMX,REMX,COLJ,IREN,NA,VB,VC,VS,
     >             ESTADO,FOBJ )
C     ---------------------------------
C     * SE INICIALIZAN VARAIBLES GLOBALES *
C     ---------------------------------
      CALL INAPLI
C     -----------------------
C     * SE ESCALA EL PROBLEMA *
C     -----------------------
      CALL ESCALA
C     ------------------------------------------
C     * MIENTRAS EXISTA O NO UNA BASE DISPONIBLE *
C     ------------------------------------------
      DO WHILE ((ISTATE.EQ.10).OR.(ISTATE.EQ.11).OR.(ISTATE.EQ.0))
C        ---------------------------------
C        * BUSCA SOLUCION AL PROBLEMA DE PL *
C        ---------------------------------
         CALL ODANLP
C        ------------------------------------------------------------
C        * SI LA SOLUCION ES COMPLETA (OPTIMA,NO ACOTATA O INFACTIBLE) *
C        ------------------------------------------------------------
```

```
      IF ( ISTATE .LE. 3 ) THEN
C     ------------------------
C     | VERIFICA EXACTITUD |
C     ------------------------
        CALL CHACC
      END IF
C     ------------------------------------------
C     | ISTATE = 7 LA EXACTITUD NO ES CORRECTA |
C     ------------------------------------------
      IF ((ISTATE .EQ. 7).AND.(IP .LT. IRMAX)) THEN
C       ------------------------------
C       | SE REALIZA UNA REINVERSION |
C       ------------------------------
        CALL REVERT
C       ---------------------------
C       | EXISTE UNA BASE COMPLETA |
C       ---------------------------
        ISTATE = 11
      END IF
    END DO
C   --------------------------
C   | SE DESESCALA EL PROBLEMA |
C   --------------------------
    CALL DESCAL
C   -------------------------
C   | SOLUCION DEL PROBLEMA |
C   -------------------------
    CALL DATSAL( ESTADO,ZOBJ,UX )
    RETURN
    END


    SUBROUTINE DATENT( NR,NV,ITHX,REHX,COLJ,IREN,BA,VB,
   >                   VC,VS,ESTADO,ZOBJ )
C   ************************************************************************
C   *                                                                      *
C   *  OBJETIVO                                                            *
C   *                                                                      *
C   *  TRANSFERIR LOS DATOS DEL PROBLEMA A LAS VARIABLES GLOBALES          *
C   *  CORRESPONDIENTES QUE UTILIZA LA SUBRUTINA DL                        *
C   *                                                                      *
C   ************************************************************************
C
C   ---------------------------------
C   | SE INCLUYE AREA DE PARAMETROS |
C   ---------------------------------
    INCLUDE 'ARPARL.FOR'
C
    IMPLICIT REAL*8 (A-H,O-Z)
C
    REAL*8 BA(MAXA),VB(MAXB),VC(MAXN)
    REAL*8 VS(MAXB)
    INTEGER COLJ(MAXA),IREN(MAXB+1),REMX
C
    COMMON / BANDA  / BNDA
    COMMON / OBJ    / OBJ
    COMMON / N      / N
    COMMON / M      / M
    COMMON / ITRMAX / ITRMAX
    COMMON / IRMAX  / IRMAX
```

```
      COMMON / ISBND  / ISBND
      COMMON / ISTATE / ISTATE
      COMMON / JCOL   / JCOL    ( 1:MAXA )
      COMMON / IROW   / IROW    ( 1:MAXN+1 )
      COMMON / AA     / AA      ( 1:MAXA )
      COMMON / B      / B       ( 1:MAXN )
      COMMON / C      / C       ( 1:MAXN )
      COMMON / BOUND  / BOUND   ( 1:MAXN )
      COMMON / S      / S       ( 1:MAXN )
C
      MNOW = MR
      M    = MNOW
      N    = NV
      ITRMAX = ITMX
      IRMAX  = REMX
      ISBND  = N
      ISTATE = ESTADO
      OBJ  = FOBJ
C
      DO  I = 1,N
          C(I) = VC(I)
          BOUND(I) = 10E5
      END DO
      DO  I = 1,MNOW
          B(I) = VB(I)
          S(I) = VS(I)
          IF ( S(I).EQ.2 ) THEN
             S(I) = -1.0
          END IF
      END DO
      DO  I = 1,MNOW+1
          IROW(I) = IREN(I)
      END DO
      I = 1
      DO WHILE( MA(I).NE.0 )
         AA(I) = MA(I)
         JCOL(I) = COLJ(I)
         I = I+1
      END DO
      RETURN
      END


      SUBROUTINE DATSAL( ESTADO,FOBJ,VX )
C     ***********************************************************************
C     *                                                                     *
C     *  OBJETIVO                                                           *
C     *                                                                     *
C     *  TRANSFERIR LOS RESULTADOS OBTENIDOS A LAS VARIABLES ESPERADAS-     *
C     *  POR LA ITERACION MAESTRA                                           *
C     *                                                                     *
C     ***********************************************************************
C
C     ---------------------------------
C     * SE INCLUYE AREA DE PARAMETROS *
C     ---------------------------------
      INCLUDE 'ARPAPL.FOR'
C
      IMPLICIT REAL A8 (A-H,O-Z)
      REAL A8 FOBJ,VX(MAXN)
```

```
      INTEGER ESTADO
C
      COMMON / NNOW    / NNOW
      COMMON / N       / N
      COMMON / ISTATE  / ISTATE
      COMMON / OBJ     / OBJ
      COMMON / X       / X        ( IMAXN )
C
C     ------------------------------------
C     | VALOR DE LAS VARIABLES PRIMALES |
C     ------------------------------------
      DO  I = 1,N
          VX(I) = X(I)
      END DO
C     ------------------------------------
C     | ESTADO DE LA SOLUCION DEL PROBLEMA |
C     ------------------------------------
      ESTADO = ISTATE
C     ------------------------------------
C     | VALOR DE LA FUNCION OBJETIVO |
C     ------------------------------------
      FOBJ = OBJ
      RETURN
      END


      SUBROUTINE INAPLI
C     ***************************************************************************
C     *                                                                         *
C     |  OBJETIVO                                                             |
C     *                                                                         *
C     |  INICIALIZAR UNA GRAN PARTE DE LAS VARIABLES GLOBALES QUE             |
C     |  SERAN UTILIZADAS EN LA SUBRUTINA PI.                                 |
C     *                                                                         *
C     ***************************************************************************
C
      IMPLICIT REAL*8 (A-H,O-Z)
      INTEGER SIGE
C
      COMMON / INREV   / INREV
      COMMON / IP      / IP
      COMMON / ISBIG   / ISBIG
      COMMON / ISDONE  / ISDONE
      COMMON / ITP     / ITP
      COMMON / KB      / KB
      COMMON / KC      / KC
      COMMON / NEGINV  / NEGINV
      COMMON / NEGROW  / NEGROW
      COMMON / NEWX    / NEWX
      COMMON / NEWY    / NEWY
      COMMON / P       / P
      COMMON / R       / R
      COMMON / SIGE    / SIGE
      COMMON / YANINC  / YANINC
      COMMON / TOL     / TOL      ( 10 )
C
C     ------------------------------------------------------------------------
C     | ISDONE = 0    NO ERRORES EN EL ARCHIVO DE DATOS LINP.DAT              |
C     | ISDONE = 1    ERROR EN EL ARCHIVO DE DATOS O FIN DEL ALGORITMO        |
C     ------------------------------------------------------------------------
```

```
ISDONE = 0
-----------------------------------------------------------
A NUMERO DE ELEMENTOS QUE CONTIENE EL VECTOR C A
-----------------------------------------------------------
KC = 0
-----------------------------------------------------------
A NUMERO DE ELEMENTOS QUE CONTIENE EL VECTOR D A
-----------------------------------------------------------
KD = 0
-----------------------------------------------------------------------
A INDICADOR PARA LAS SUBRUTINAS CHOLCH (HAN SIDO ALTERADAS LAS A
A VARIABLES DE HOLGURA),CHNSIS (SOLO SE REALIZARA EL CAMBIO DE A
A BASE EN LOS ELEMENTOS DE LA INVERSA) Y REVERT.              A
-----------------------------------------------------------------------
INREV = 0
---------------------------
A NUMERO DE REINVERSIONES A
---------------------------
IR = 0
---------------------------
A NUMERO DE ITERACIONES A
---------------------------
ITR = 0
-----------------------------------------------------------
A RENGLON DE LA INVERSA ASOCIADO A UNA VARIABLE INFACTIBLE A
-----------------------------------------------------------
NEGINV = 0
-----------------------------------------------------------
A RENGLON DE LA MATRIZ A ASOCIADO A UNA VARIABLE INFACTIBLE A
-----------------------------------------------------------
NEGROW = 0
-----------------------------
A VARIABLE QUE ENTRA A LA BASE A
-----------------------------
NEWX = 0
-----------------------------------------------------------
A RENGLON DE LA INVERSA ASOCIADO CON LA VARIABLE QUE SALE A
A DE LA BASE.                                             A
-----------------------------------------------------------
NEWY = 0
-----------------------------------------
A LIMITE DE LA VARIABLE QUE ENTRA A LA BASE A
-----------------------------------------
R = 0.0
-------------------------------
A TAMANO DE LA MATRIZ INVERSA A
-------------------------------
SIZE = 0
-----------------------------------------
A TAMANO MAXIMO ALCANZADO POR LA INVERSA A
-----------------------------------------
ISBIG = 1
-----------------------------------------------------------
A AUXILIAR PARA DETERMINAR LA VARIABLE QUE ENTRA A LA BASE A
-----------------------------------------------------------
YAMINC = 0.0
-----------------
A TOLERANCIAS A
-----------------
TOL(1) = 1.0E-3
```

```
      TOL(2) = 1.0E-3
      TOL(3) = 1.0E-3
      TOL(4) = 1.0E-3
      TOL(5) = 1.0E-3
      TOL(6) = 1.0E-3
      TOL(7) = 1.0E-3
      TOL(8)   TOL(5)*10
      RETURN
      END


      SUBROUTINE CHACC
C     ****************************************************************
C     *                                                              *
C     *  OBJETIVO                                                    *
C     *                                                              *
C     *  VERIFICAR LA EXACTITUD EN LA SOLUCION.  AL  REALIZAR  LOS   *
C     *  CALCULOS DE  (YA - C)  EN LAS VARIABLES BASICAS, Y  (B - AX  *
C     *  MENOS EL VECTOR DE HOLGURAS) EN TODAS LAS RESTRICCIONES. SE  *
C     *  DESEARIA QUE ESTOS FUERAN IGUAL A CERO; DE NO SUCEDER  ASI,  *
C     *  SE PRETENDE QUE DIFIERAN DE CERO POR NO MAS DE UNA TOLERAN-  *
C     *  CIA ACORDADA; SI NO SE SATISFACE ALGUNA TOLERANCIA, LA  SO-  *
C     *  LUCION ENCONTRADA NO ES EXACTA (ISTATE = 7).                 *
C     *                                                              *
C     ****************************************************************
C
C     ---------------------------------
C     * SE INCLUYE AREA DE PARAMETROS *
C     ---------------------------------
      INCLUDE 'ARPAPL.FOR'
C
      IMPLICIT REAL*8 (A-H,O-Z)
      INTEGER XBASIS, SIZE, SIZE1
      LOGICAL EXITO,EXITO1
C
      COMMON / ISEFF  / ISEFF  ( 1:MAXM )
      COMMON / ISTATE / ISTATE
      COMMON / SLACK  / SLACK  ( 1:MAXM )
      COMMON / YAC    / YAC    ( 1:MAXN )
      COMMON / B      / B      ( 1:MAXM )
      COMMON / C      / C      ( 1:MAXN )
      COMMON / INBASE / INBASE ( 1:MAXN )
      COMMON / MNOW   / MNOW
      COMMON / N      / N
      COMMON / SIZE   / SIZE
      COMMON / TOL    / TOL    ( 1:8 )
      COMMON / X      / X      ( 1:MAXN )
      COMMON / XR     / XR     ( 1:MXSIZE )
      COMMON / Y      / Y      ( 1:MAXN )
      COMMON / AA     / AA     ( 1:MAXA )
      COMMON / IROW   / IROW   ( 1:MAXM+1 )
      COMMON / JCOL   / JCOL   ( 1:MAXA )
      COMMON / XBASIS / XBASIS ( 1:MXSIZE )
      COMMON / NUMSLK / NUMSLK
C
C     ------------------------------------------------ ---
C     * SI EXISTEN VARIABLES DE HOLGURA EN LA BASE *
C     ------------------------------------------------
      IF ( NUMSLK .NE. 0 ) THEN
C     ------------------------------------------------------------
```

```
C        & VECTOR DE HOLGURAS CORRECTO EN ALGUNA RESTRICCION QUE &
C        & ESTE EN LA BASE                                       &
C        ---------------------------------------------------------
         DO  K = 1, SIZE
C           ------------------------------------
C           & SI ES UNA VARIABLE BASICA DE HOLGURA &
C           ------------------------------------
            IF ( XBASIS(K) .GT. N ) THEN
C              ------------------------------
C              & INDICE EN EL VECTOR DE HOLGURAS &
C              ------------------------------
               I = XBASIS ( K ) - N
C              ------------------------------
C              & VALOR DE LA VARIABLE DE HOLGURA &
C              ------------------------------
               SLACK ( I ) = XB ( K )
            END IF
         END DO
      END IF
C     ------------------------------------
C     & PRIMER PASO PARA CALCULAR YAC(J) &
C     ------------------------------------
      DO  J = 1, N
C        ---------------------------
C        & SI ES UNA VARIABLE BASICA &
C        ---------------------------
         IF ( INBASIC(J) .GT. 0 ) THEN
            YAC ( J ) = - C ( J )
         END IF
      END DO
C     ---------------------------------------------------------------
C     & TOLERANCIAS &
C
C     EL VALOR DE ESTAS TOLERANCIAS SUE CONSIDERADO POR EL USUARIO
C     COMO UN NIVEL DE EXACTITUD PARA:
C
C       TOL(3)  FACTIBILIDAD DE LAS VARIABLES BASICAS DE HOLGURA (SLACK).
C       TOL(4)  OPTIMALIDAD EN LAS VARIABLES DEL RENGLON DE LA FUNCION.
C               OBJETIVO, YAC(J).
C       TOL(6)  VALOR DE LAS VARIABLES PRIMALES.
C       TOL(7)  VALOR DE LAS VARIABLES DUALES.
C
C     ---------------------------------------------------------------
      EXITO = .TRUE.
      I = 1
C     ---------------------------------------------------------------
C     & SE CALCULA  YAC(J)  EN LAS VARIABLES BASICAS Y LOS ELEMENTOS &
C     & DE RAYCL (B - AX - SLACK).                                   &
C     ---------------------------------------------------------------
      DO WHILE ( I.LE.NROW .AND. EXITO )
C        ------------------------------
C        & RESTRICION EFECTIVA EN LA BASE &
C        ------------------------------
         ISEFFI = ISEFF ( I )
C        ------------------------------
C        & VALOR DE LA VARIABLE DUAL &
C        ------------------------------
         Y1 = Y ( I )
C        ------ ------
C        & B-AX-SL &
```

```
C        ----------
         BAXSL = B ( I ) - SLACK ( I )
         ISTART = IROW ( I )
         LAST = IROW ( I + 1 ) - 1
C        ------------------------------
C        A CALCULO DE YA-C  Y DE B-AX-SL A
C        ------------------------------
         DO  LOOK = ISTART, LAST
C           ---------------------
C           A COLUMNA DE LA MATRIZ A A
C           ---------------------
            J = JCOL ( LOOK )
            INJ = INBASE ( J )
C           ------------------------------
C           A SI LA J-ESIMA VARIABLE ES BASICA A
C           ------------------------------
            IF ( INJ .NE. 0 ) THEN
C              ---------------------
C              A ELEMENTO EN LA MATRIZ A A
C              ---------------------
               AIJ = AA ( LOOK )
C              ----------------
C              A B-AX (SLCK=0) A
C              ----------------
               BAXSL = BAXSL - X ( J ) * AIJ
C              ------------------------------------------------------
C              A SI LA VARIABLE ES BASICA Y LA RESTRICCION ASOCIADA A
C              A ES EFECTIVA EN LA BASE.                            A
C              ------------------------------------------------------
               IF ( INJ .GT. 0 .AND. ISEFEI .NE. 0 ) THEN
C                 ----------------
C                 A CALCULO DE YA-C A
C                 ----------------
                  YAC(J) = YAC(J) + YI*AIJ
               END IF
            END IF
         END DO
         ERR = BANS ( BAXSL )
C        ------------------------------
C        A SI NO EXISTE EXACTITUD EN  B-AX-SL A
C        ------------------------------
         IF ( ERR .GT. TOL(2) ) THEN
C           ----------------
C           A SE ESCRIBE UN MENSAJE A
C           ----------------
            WRITE(2,9000) ERR,I
C           ------------------------
C           A EXACTITUD NO ES CORRECTA A
C           ------------------------
            ISTATE = 7
            EXITO  .FALSE.
         ELSE
            ABSB = BANS ( B(I) )
            IF ( ABSB .LT. 1.0 ) THEN
               ABSB  1.0
            END IF
C           ------------------------------------------------
C           A SI LA EXACTITUD RELATIVA DE  B(I) NO ES CORRECTA A
C           ------------------------------------------------
            IF ( ERR/ABSB .GT. TOL(2) ) THEN
```

```
                RELERR   ERR/ABSC
C               ---------------------
C               A SE ESCRIBE UN MENSAJE A
C               ---------------------
                WRITE(2,9004) RELERR,I,ERR,ABSC
C               ---------------------
C               A EXACTITUD NO ES CORRECTA A
C               ---------------------
                ISTATE = 7
                EXITO = .FALSE.
             END IF
          END IF
          I = I + 1
       END DO
C      ---------------------------------
C      A SI LA EXACTITUD HA SIDO CORRECTA A
C      ---------------------------------
       IF ( EXITO ) THEN
          I = 1
          EXITO1 = .TRUE.
C         ---------------------------------------------------
C         A SE VERIFICA EXACTITUD EN  YAC(J) Y CH  C(J) A
C         ---------------------------------------------------
          DO WHILE ( J.LE.N .AND. EXITO1 )
C            ---------------------------------
C            A SI LA J-ESIMA VARIABLE ES BASICA A
C            ---------------------------------
             IF ( INBASE(J) .GT. 0 ) THEN
                ERR = DABS ( YAC(J) )
C               ---------------------------------------------------
C               A SI LA EXACTITUD PARA  YAC(J) NO ES CORRECTA A
C               ---------------------------------------------------
                IF ( ERR .GT. TOL(4) ) THEN
C                  ---------------------
C                  A SE ESCRIBE UN MENSAJE A
C                  ---------------------
                   WRITE(2,9008) ERR,J
C                  ---------------------
C                  A EXACTITUD NO ES CORRECTA A
C                  ---------------------
                   ISTATE = 7
                   EXITO1 = .FALSE.
                ELSE
                   ABSC = DABS ( C(J) )
                   IF ( ABSC .LT. 1.0 ) THEN
                      ABSC = 1.0
                   END IF
C                  ---------------------------------------------------
C                  A SI LA EXACTITUD RELATIVA PARA C(J) NO ES CORRECTA A
C                  ---------------------------------------------------
                   IF ( ERR/ABSC .GT. TOL(7) ) THEN
                      RELERR = ERR/ABSC
C                     ---------------------
C                     A SE ESCRIBE UN MENSAJE A
C                     ---------------------
                      WRITE(2,9012) RELERR,J,ERR,ABSC
C                     ---------------------
C                     A EXACTITUD NO ES CORRECTA A
C                     ---------------------
                      ISTATE = 7
```

```
      END IF
      RETURN
      END

      SUBROUTINE RESUL( IRES,OBJM,XE )
C     ***************************************************************
C     *                                                             *
C     *  OBJETIVO                                                   *
C     *                                                             *
C     *  ESCRIBIR EN EL ARCHIVO DE RESULTADOS, BBW.RES, EL TIPO DE  *
C     *  SOLUCION ENCONTRADA (OPTIMA,INFACTIBLE,NO ACOTADA), Y EN   *
C     *  CASO DE EXISTIR, LA SOLUCION DEL PROBLEMA.                 *
C     *                                                             *
C     ***************************************************************
C
C     --------------------------------
C     * SE INCLUYE AREA DE PARAMETROS *
C     --------------------------------
      INCLUDE 'ARPAPL.FOR'
C
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8  XE(MAXN)
C
      COMMON / NN / NN
C
C     ------------------------------------------
C     * SI EL PROBLEMA TUVO SOLUCION OPTIMA *
C     ------------------------------------------
      IF ( IRES.EQ.1 ) THEN
C        --------------------------
C        * SE ESCRIBE LA SOLUCION *
C        --------------------------
         WRITE(2,100)
         DO J = 1,NN
            WRITE(2,101) J,XE(J)
         END DO
C        -------------------------------
C        * VALOR DE LA FUNCION OBJETIVO *
C        -------------------------------
         WRITE(2,102) OBJM
C     ---------------------------------
C     * SI EL PROBLEMA ES INFACTIBLE *
C     ---------------------------------
      ELSE IF ( IRES.EQ.2 ) THEN
C        -------------------------
C        * SE ESCRIBE UN MENSAJE *
C        -------------------------
         WRITE(2,103)
C     ---------------------------------
C     * SI EL PROBLEMA ES NO ACOTADO *
C     ---------------------------------
      ELSE IF ( IRES.EQ.3 ) THEN
C        -------------------------
C        * SE ESCRIBE UN MENSAJE *
C        -------------------------
         WRITE(2,104)
C     ------------------------------------------------------
C     * SI EL NUMERO MAXIMO DE ITERACIONES FUE REBASADO *
C     ------------------------------------------------------
```

REFERENCIAS

[1] Abadie J. 1967. Nonlinear Programming. Amsterdam: North-Holland. pp 143-160.

[2] Bazaraa M., and Jarvis J. 1977. Linear Programming and Network Flows. New York: John Wiley and Sons.

[3] Ceciliano J.L., Guillén I. Programa PLP. Reporte Interno. Instituto de Investigaciones Eléctricas. Noviembre de 1991.

[4] Ceciliano J.L., Guillén I. Programa QP. Reporte Interno. Instituto de Investigaciones Eléctricas. Enero de 1992.

[5] Ceciliano J.L., Guillén I. Programa DDW. Reporte Interno. Instituto de Investigaciones Eléctricas. Mayo de 1992.

[6] Cottle R., and Krarup J. 1974. Optimization Methods for Resource Allocation. New York: The English Universities Press.

[7] Fulkerson D., and Wolfe P. 1962. "An algorithm for scaling matrices." SIAM Review. 4: 142-146.

[8] Garfinkel R., and Nemhauser G. 1972. Integer Programming. New York: John Wiley and Sons. pp 111-122.

[9] Greenberg H. 1978. Design and Implementation of Optimization Software. The Netherlands: Sijthoff & Noordhoff International Publishers. pp 225-259.

[10] Guillén I., Ceciliano J.L. Programa LINP. Reporte Interno. Instituto de Investigaciones Eléctricas. Julio de 1991.

[11] Guillén I., Ceciliano J.L. Programa BB. Reporte Interno. Instituto de Investigaciones Eléctricas. Septiembre de 1991.

[12] Hillier F., and Lieberman G. 1974. Operations Research. San Francisco: Holden-Day.

[13] James K., and Etienne L. 1981. "An advanced Implementation of the Dantzig-Wolfe decomposition algorithm for linear programming." Mathematical Programming. 20: 303-326.

[14] Land A., and Powell S. 1972. Fortran Codes for Mathematical Programming. New York: John Wiley and Sons.

[15] Lasdon L. 1970. Optimization Theory for Large Systems. New York: Macmillan Publishing. pp 144-162.

[16] Luenberger D. 1989. Linear and Nonlinear Programming. Mass: Addison-Wesley.

[17] Murray W., et al. 1981. Practical Optimization. New York: Academic Press.

[18] Salkin H. 1975. Integer Programming. Mass: Addison-Wesley.

[19] Tromlin J. 1975. "On scaling linear programming problem." Mathematical Programming Study. 4: 142-145.

[20] Vajda S. 1961. Mathematical Programming. Massachusetts: Addison Wesley. pp 222-231.