UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

DIVISION DE ESTUDIOS SUPERIORES

FACULTAD DE INGENIERIA

EL PROBLEMA LINEAL ANGULAR: UN METODO DE SOLUCION

TESIS DE MAESTRIA

que para obtener el grado de

MAESTRO EN INGENIERIA-INVESTIGACION DE OPERACIONES

p r e s e n t a :

MANUEL DE LOS REYES GARCIA MARTINEZ

DIRECTOR DE TESIS:

DR. SERGIO FUENTES MAYA

Abril 2002

FACULTAD DE INGENIERIA
DIVISION DE ESTUDIOS SUPERIORES
SECCION DE INVESTIGACION DE OPERACIONES

TESIS QUE PRESENTA EL
ING. MANUEL DE LOS REYES GARCIA MARTINEZ

para obtener el grado de

MAESTRO EN INVESTIGACION DE OPERACIONES

CREDITOS POR TESIS:
12(DOCE)

JURADO:

Presidente: M. EN I. LEONARD RAPOPORT ZAWITZ
Vocal: DR. SERGIO FUENTES MAYA
Secretario: DR. JAVIER MARQUEZ DIEZ CANEDO
Suplente: DR. MIGUEL COBIAN SELA
Suplente: M. EN I. JESUS LARA TEJADA

JEFE DE LA SECCION

DR. SERGIO FUENTES MAYA

A mis padres

Manuel García G. y Angela M. de García

In Memoriam.

A mi esposa, Larisa Semenóvna Márkina
Su amor, apoyo, comprensión y
sacrificios me han ayudado siempre
a salir adelante.

A mis hijos Larisa Angela, Manuel
de los Reyes y Yekaterina Manuelyevna,
quienes han compartido los momentos
difíciles.

A mi hermana
Concepción García M.
Con profundo agradecimiento
por su valiosa ayuda.

RECONOCIMIENTO

Al Dr. Sergio Fuentes Maya, prestigiosa autoridad en la materia, director de esta tesis. Por las innumerables enseñanzas recibidas, por la amistad y ayuda brindadas.

I N D I C E

CAPITULO I

INTRODUCCION

Uno de los aspectos actuales más importantes de la programación lineal es la solución eficiente de problemas lineales estructurados con gran número de variables y restricciones. Dichos problemas resultan de la modelación de una variedad de situaciones reales cuyo propósito es el uso óptimo de recursos de acuerdo a objetivos especificados. La preocupación por resolver problemas lineales estructurados por medio de métodos especiales, denominados métodos de descomposición, ha existido desde el inicio de la programación lineal. La motivación fundamental de estos métodos es la explotación de la estructura especial de estos problemas. Los primeros resultados en este sentido están resumidos en el denominado principio de descomposición de Dantzig y Wolfe. Sin embargo, la práctica ha demostrado que los métodos basados en este principio resultan difíciles de programar, y aunque esto es posible, se encuentran dificultades con la convergencia del método. Otra desventaja de estos métodos es que no es posible efectuar un análisis de sensibilidad de la solución óptima.

En forma paralela a los métodos basados en el principio de descomposición de Dantzig y Wolfe, se empezaron a diseñar técnicas especiales para seguir aplicando los métodos tradicionales de programación lineal. Conviene recordar que el fundamento y propósito de estos métodos es la jerarquización

construcción de las bases asociadas a las restricciones, de acuerdo al valor de la función objetivo, para determinar el valor óptimo. También se recuerda que la actualización ó cambio de bases y sus inversas es lo que más manipulaciones algebráicas y tiempo requiere en los programas de computadora de programación lineal. Debido a esto, y para el caso especial de problemas estructurados se empezó a desarrollar una técnica de manejo y actualización de bases que dió origen a los nuevos métodos de descomposición. En este respecto se puede mencionar los trabajos de Dantzig y Van-Slyke, Kaul, Benett, Muller-Merbach, Orchard-Hay, Lasdon y otros. Lo interesante de estos trabajos es que la idea básica de los métodos se ha ido refinando y es posible unificarlos respecto a ciertos criterios, un ejemplo de esto es el trabajo de Winkler (ref.34], Los nuevos métodos de descomposición han sido denominados métodos de factorización de la base, pues hacen una explotación exhaustiva de las propiedades de la matriz base, la cual puede expresarse como el producto de dos matrices angulares cuya estructura permite manejar y actualizar la inversa de la matriz base en forma, simple y eficiente desde el punto de vista computacional, ya que los métodos tradicionales resultan demasiado generales y restrictivos. Por ejemplo, los métodos comerciales de programación lineal tienen limitantes respecto al número de restricciones y no consideran en forma explícita la gran cantidad de ceros que existe en la matriz de restricciones. Aunado a esto se tienen restricciones de memoria debido al tamaño del modelo y a la computadora que se utilice. Por otra parte, suponiendo que estos

problemas sean resueltos se puede llegar, sin embargo, a tener que el tiempo de proceso del modelo resulte casi incosteable.

En este trabajo se establecen los fundamentos teóricos para el análisis y solución de los modelos lineales estructurados descritos anteriormente, esto es, se desarrolla el método de solución simplex revisado para el caso del problema lineal angular con restricciones de liga, usando la técnica de factorización de la base. Conviene recordar que la idea fundamental de este método es la jerarquización constructiva de las bases asociadas a la matriz de restricciones, de acuerdo al valor de la función objetivo, hasta obtener el valor óptimo. También conviene recordar que esta actualización ó cambio de bases y sus inversas, es lo que más tiempo consume en los paquetes comerciales de programación lineal, razón por la cual resulta ventajoso el empleo de la técnica de factorización de la base. Asimismo, en este trabajo se implementa un programa de computadora que utiliza la técnica de factorización de la base en el método simplex revisado, denominado programa G-GUB (Generalized-General Upper Bound). Este programa fue elaborado y desarrollado por Winkler (ref. 34) usando todas las facilidades técnicas que ofrece el sistema de computadoras IBM disponible en la Universidad de Stanford. Sin embargo, debido a las diferencias propias del manejo de archivos de los sistemas IBM y CDC, este último disponible en la Secretaría de Agricultura y Recursos Hidráulicos, fue necesario hacer diversas modificaciones para el uso de este programa en la Comisión

del Plan Nacional Hidráulico. También es importante hacer notar que no existía manual de uso del programa, ni se conocía la función desempeñada por cada una de las subrutinas que lo componen y además, la capacidad del programa estaba demasiado restringida para los usos que se le pretende dar, esto es, la solución de modelos hidroagrícolas nacionales de gran número de variables y restricciones, razón por la cual el programa G-GUB hubo de sufrir nuevas modificaciones.

El presente trabajo se desarrolla como sigue: en el capítulo II se tratan los conceptos básicos de programación lineal utilizados en el desarrollo de este trabajo. En el capítulo III se analiza el problema lineal con cotas superiores generalizadas, usando el método simplex revisado con factorización de la base. En el capítulo IV se extiende el análisis precedente al problema lineal angular con restricciones de liga, donde se incluye un ejemplo numérico. Este trabajo contiene también dos apéndices, en el apéndice A se dan algunos resultados de convexidad de utilidad para el desarrollo del capítulo II. Finalmente en el apéndice B, se describe el programa G-GUB, las modificaciones hechas al mismo, su diagrama de bloques y se incluye la corrida del ejemplo resuelto en el capítulo IV.

Deseo agradecer a las autoridades de la Comisión del Plan Nacional Hidráulico, y en forma especial a la Dirección del Centro de Información por las facilidades brindadas en la realización de este trabajo. Quiero expresar mi agradecimiento, también, a la Sra. Cristina Arias González por la labor de mecanografía que tan eficientemente realizó.

CAPITULO II

CONCEPTOS BASICOS DEL PROGRAMA LINEAL

2.1 Definición

Un problema de programación lineal consiste en la maximización o minimización de una función lineal de varias variables sujeta a restricciones lineales en estas mismas variables. Una forma particular, a la que cualquier problema lineal puede ser transformado, es la siguiente forma estándar:

$$\text{minimice } c_1x_1 + c_2x_2 + \dots + c_nx_n$$

sujeto a

$$\begin{aligned} & a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \dots + a_{1n}x_n = b_1 \\ \text{(P)} \quad & a_{21}a_1 + a_{22}x_2 + a_{23}x_3 + \dots + a_{2n}x_n = b_2 \\ & \dots\dots\dots\dots\dots\dots\dots\dots\dots \\ & a_{m1}x_1 + a_{m2}x_2 + a_{m3}x_3 + \dots + a_{mn}x_n = b_m \\ & x_1 \geq 0 \ ; \ x_2 \geq 0 \ ; \ \dots \ ; \ x_n \geq 0 \end{aligned}$$

donde los coeficientes a_{ij}, b_i y c_j son números reales, y x_i, $i = 1, \dots, n$ son las variables a determinar. Una forma compacta y usual de escribir (P) es

$$\text{minimice } \ cx$$

$$Ax = b$$

$$x \geq 0$$

donde A, es una matriz nxm; c, un vector hilera de n componentes; b, un vector columna de m componentes y x, un vector columna de n variables.

Considérese el sistema de ecuaciones

$$Ax = b$$

donde A es una matriz de m renglones y n columnas; b, es un vector columna de m componentes, y x es un vector columna de n componentes. Supóngase que el rango de la matriz A es m, por lo que entonces es posible elegir m columnas de A que sean linealente independientes y sin pérdida de generalidad puede considerarse que éstas son las primeras m columnas, las cuales forman una submatriz de A de orden mxm, no singular a la que se denominará B. De esta forma se puede escribir $A = [B,R]$ donde R está formada por las restantes n-m columnas de A.

Se define ahora $X = (X_B, X_R)^T$, donde X_B de dimensión m está asociado a la matriz B, X_R de dimensión n-m está asociada a la matriz R. Entonces se puede escribir

$$Ax = [B,R] \begin{bmatrix} X_B \\ X_R \end{bmatrix} = b$$

Haciendo $X_R = 0$, se obtiene una solución para $Ax = b$, ya que $BX_B = b$ y por ser B no singular, entonces, $X_B = B^{-1}b$, esto es,

$$X = \left[X_B, X_R\right]^T = \left[B^{-1}b, 0\right]^T$$

Esta solución se denomina una solución básica con respecto a la base B y las componentes de X asociados a las columnas de B, ésto es, X_B, se denominan variables básicas. Si una o más de las variables básicas tienen valor cero es que la solución básica es degenerada.

Se observa que en una solución básica no degenerada es inmediata la identificación de las columnas de A que forman la matriz no singular B. En cambio, en una solución degenerada, existe cierta ambiguedad para identificar B, pues las variables básicas con valor cero pueden ser confundidas con las variables no básicas cuyo valor es cero también.

Considérese el problema lineal en forma estándar

$$\text{minimizar } Z = Cx$$

s.a.

$$Ax = b$$

$$x \geq 0$$

donde A es una matriz de orden mxn; b es un vector columna de m componentes; c es un vector hilera de n componentes y x es un vector columna de n componentes a determinar. Se dice que X es una solución factible si satisface las restricciones del problema. Si la solución factible es también básica, se dice que es una solución factible básica. Si en la solución factible básica, una o más de las variables básicas son nulas, entonces se trata de una solución factible básica degenerada.

Finalmente, una solución factible para la cual, la función - objetivo adquiere, el valor mínimo se denomina solución óptima.

2.2 Teorema fundamental de la programación lineal

En la solución de problemas de programación lineal, el papel que desempeñan las soluciones factibles básicas es de primordial importancia. Específicamente, el teorema fundamental de la programación lineal demuestra que en la determinación de - soluciones óptimas del problema lineal, en forma estándar, es únicamente necesario considerar las soluciones factibles básicas de este problema.

TEOREMA FUNDAMENTAL DE LA PROGRAMACION LINEAL. Dado el problema

$$\begin{aligned} &\min \quad Z = Cx \\ &\text{sujeto a} \qquad\qquad (2.2.1) \\ &\qquad Ax = b \\ &\qquad x \geq 0 \end{aligned}$$

donde A es una matriz de m renglones, n columnas y además de rango m, b es un vector columna de m componentes, c es un vector renglón de n componentes y x es un vector columna de n componentes.

i. Si existe una solución factible, existe una solución factible que es básica.

ii. Si existe una solución factible óptima, existe una solución factible básica que es óptima.

El teorema anterior establece que se puede reducir la búsqueda de soluciones óptimas del problema lineal en forma estándar a su subconjunto formado por las soluciones factibles básicas, el cual es finito. En particular, se observa que el número de soluciones factibles básicas en un problema de programación lineal con m restricciones y n variables es a lo más

$$\binom{n}{m} = \frac{n!}{m!\,(n-m)!}$$

correspondiente al número de maneras de seleccionar m de las n columnas de la matriz A.

Teóricamente, se ha encontrado ya una solución al problema de programación lineal y aquí resulta conveniente el puntualizar que el teorema fundamental es sólo una alternativa para resolver el problema lineal, pero en general, esta alternativa resulta práctica y computacionalmente ineficiente, ya que si por ejemplo, se tiene un problema de 10 variables y 7 restricciones, se tendrían que analizar 120 bases. Sin embargo, la extensión de los argumentos de prueba de este teorema han servido de base para diseñar métodos de solución eficientes, tal como el método simplex.

El concepto de solución básica en un problema de programación lineal está relacionado con el concepto de punto extremo de un cierto politope convexo. Esta relación de conceptos, uno algebráico y otro geométrico se tratan a continuación.

2.3 Relaciones importantes de la programación lineal con convexidad.

Uno de los conceptos más importantes en la programación lineal es el de convexidad, ya que geométrica y analíticamente, es sencillo manipular este concepto. Asimismo, al utilizar la convexidad se obtienen los resultados analíticos más relevantes de la programación lineal. Por lo anterior, en el apéndice A, se tratan los resultados más relevantes sobre conjuntos convexos y optimización.

Es interesante hacer notar que existe una relación bien definida entre las soluciones factibles básicas de un sistema lineal $Ax = b$, $x \geq 0$ y los puntos extremos del politope formado por las soluciones de este sistema.

TEOREMA 2.3.1 (Equivalencia de puntos extremos y soluciones básicas). Sean A una matriz mxn de rango m y b un vector columna de m componentes. Sea el politope convexo.

$$K = \{x \in R^n \; ; \; Ax = b, \; x \geq 0\}$$

Entonces, un vector x es un punto extremo de K, si y sólo si $x \geq 0$ y x es una solución básica de $Ax = b$. (Ver definición A.10 del apéndice A).

Prueba. Teorema A.1 del apéndice A.

Corolario 2.3.1 Si el conjunto convexo K del teorema anterior es no vacío, entonces posee al menos un punto extremo.

Colorario 2.3.2 Si el politope convexo K del teorema 2.3.1 es acotado, entonces K consiste de las combinaciones convexas de puntos extremos.

Proposición 2.3.1 En un problema de programación lineal, la solución óptima ocurre en un punto extremo del poliedro convexo K del teorema 2.3.1

Prueba. Teorema A.2 del apéndice A.

2.4 Problemas lineales duales

2.4.1 Generalidades. Considérese los problemas lineales.

	minimice cx	maximice λb	
(P)	s.a.	s.a.	(D)
	$Ax \geq b$	$\lambda A \leq C$	
	$x \geq 0$	$\lambda \geq 0$	

donde A es una matriz mxn; b, un vector columna de m componentes; c, un vector hilera de n componentes; x, un vector - columna de n incógnitas y λ, un vector hilera de m variables. Estos problemas se denominan problemas lineales duales y se dice que (P) es el problema primal y (D) el correspondiente dual. También se dice que estos problemas están en forma simétrica, pues el vector de variables a determinar en ambos problemas es no negativo y el número de restricciones del - primal (dual) es igual al número de variables a determinar del problema dual (primal). La forma asimétrica de los problemas lineales duales será cuando el vector de variables es restringido (no libres) en un problema y en el otro,no lo es.

En general, se cumple que si alguna de las desigualdades del problema primal se cambia a igualdad, el componente corres-

pondiente del vector λ en el problema dual será una variable no restringida. Recíprocamente, si alguno de los componentes del vector x en el problema primal es no restringido, la desigualdad correspondiente en el problema dual será igualdad. Las reglas que aquí se han mencionado no son arbitrarias, sino que se desprenden como consecuencia de los problemas duales establecidos.

A continuación se dan dos teoremas de alternativas, de los - varios que se establecen en teoría de desigualdades, que son esencialmente útiles en la demostración del teorema de dualidad, tema de la siguiente sección.

TEOREMA 2.4.1 (Farkas) Sea A una matriz nxn y b un vector - en R^m. Entonces, uno y sólo uno de los siguientes sistemas - de desigualdades lineales tiene solución.

i. $Ax = b \; ; \; x \geq 0$

ii. $\lambda A \geq 0 \; ; \; \lambda b < 0$

Demostración. La prueba de este teorema se puede consultar - en el texto "Programación Lineal" de M. Simmonard, ref. [29]

TEOREMA 2.4.2 Sea A una matriz mxn y b un vector de m componentes. Entonces, uno y sólo uno de los siguientes sistemas de desigualdades tiene solución.

i. $Ax \leq b \; ; \; x \geq 0$

ii. $\lambda A \geq 0 \; ; \; \lambda b < 0 \; ; \; \lambda \geq 0$

Demostración. *Primeramente se observa que i es equivalente a*

$$[A, I] \begin{bmatrix} x \\ y \end{bmatrix} = b \quad ; \quad x \geq 0 \; ; \; y \geq 0$$

Por lo tanto, del teorema 2.4.1, se tiene que el sistema anterior tiene solución si y sólo si el sistema de desigualdades

$$\lambda A \geq 0 \; ; \; \lambda I \geq 0 \; ; \; \lambda b < 0$$

no tiene solución. Este resultado prueba el teorema.

2.4.2 Teorema de Dualidad

Con el propósito de establecer el teorema de dualidad, es conveniente considerar el siguiente resultado de los problemas lineales duales.

Proposición 2.4.1

$$\min \{cx \; ; \; Ax \geq b, \; x \geq 0\} \geq \max \{\lambda b; \; \lambda A \leq C, \lambda \geq 0\}$$

Prueba. Sean x y λ soluciones factibles (arbitrarias) de los respectivos problemas lineales duales. Entonces

$$cx \geq (\lambda A)X = \lambda(Ax) \geq \lambda b$$

que es equivalente al resultado de la proposición.

Corolario 2.4.1 Sean x^* y λ^* soluciones factibles de los problemas lineales anteriores. Si $cx^* = \lambda^* b$ se tiene que x^* y λ^* son soluciones óptimas de estos problemas.

TEOREMA DE DUALIDAD. Considérense los problemas duales

$$
\begin{array}{lll}
 & \min Z = cx & \max W = \lambda b \\
(P) & \text{s.a.} & \text{s.a.} \qquad (D) \\
 & Ax \geq b & \lambda A \leq c \\
 & x \geq 0 & \lambda \geq 0
\end{array}
$$

i. Si (P) y (D) son factibles, entonces min Z = max W (finito)

ii. Si (P) factible y (D) no factible, tenemos min Z no acotado

iii. Si (P) no factible y (D) factible, se tiene max W no acotado

ix. (P) y (D) pueden ser ambos no factibles.

Corolario 2.4.2 Si alguno de los problemas lineales duales (P) ó (D) tienen solución óptima, lo mismo es cierto del otro problema y el correspondiente valor de la función objetivo es el mismo. Por otra parte, si uno de los problemas tiene función objetivo no acotada, el otro problema no tiene solución factible.

2.4.3 Teorema de Complementaridad. Sean los problemas lineales duales

$$
\begin{array}{lll}
 & \min Z = C_x & \max W = \lambda b \\
(P) & \text{s.a.} & \text{s.a.} \qquad (D) \\
 & Ax \geq b & \lambda A \leq C \\
 & x \geq 0 & \lambda \geq 0
\end{array}
$$

Sean x^* y λ^* soluciones factibles de (P) y (D) respectivamente. Entonces, una condición necesaria y suficiente para que x^* y λ^* sean soluciones óptimas es que satisfagan las relaciones

a. $x_i^* > 0$ implica $\lambda a_i^* = c_i$

b. $x_i^* = 0$ si $\lambda a_i^* < c_i$

c. $\lambda_j^* > 0$ implica $a^j x^* = b_j$

d. $\lambda_j^* = 0$ si $a^j x^* > b_j$

donde a_i (a^j) es el i-ésimo (j-ésimo) vector hilera (columna) de la matriz A.

Prueba. Si x^* y λ^* son soluciones factibles de (P) y (D) respectivamente, entonces

$$\alpha = \lambda^*(Ax^* - b) \geq 0$$

$$\beta = (C - \lambda^* A)x^* \geq 0$$

de donde, $\alpha + \beta = cx^* - \lambda^* b$. Sin embargo, el teorema de dualidad establece que x^* y λ^* son soluciones óptimas sí y sólo si se cumple que $cx^* = \lambda^* b$, lo que es equivalente a que $\alpha = 0$ y $\beta = 0$, con lo que la prueba termina, pues las relaciones del teorema son fáciles de implicar dado que α y β resultan del producto de vectores no negativos.

El resultado de este teorema tiene una importante interpretación económica. Específicamente, se puede decir que si una

restricción del problema primal (P) es activa, esto es, $a^j x^* = b_j$ entonces, el precio a que se compraría una unidad adicional al recurso j es igual a λ_j^*. En particular, nótese que si la restricción es no activa, esto es, $a^j x^* > b_j$, el precio a que se compraría la unidad adicional de recurso es igual a cero. De esto se justifica el que los elementos λ_j^*, j=1,...,m sean denominados precios sombra o precios de oportunidad.

2.5 Método simplex revisado

Sea el problema de programación lineal

$$\min \quad Z = Cx$$

s.a.

$$Ax = b$$

$$x \geq 0$$

donde A es una matriz de mxn de rango completo, c es un vector renglón de n componentes, b es un vector columna de m - componentes y x es un vector columna de n componentes.

Supóngase que la base B consiste de las primeras m columnas de A. Además se parte a A, X y C de manera que

$$A = (B,R) \; ; \quad X = (X_B, X_R)^T \; ; \quad C = (C_B, C_R)$$

De lo anterior el problema lineal equivalente es:

$$\min Z = C_B X_B + C_R X_R$$

s.a.

$$BX_B + RX_R = b$$

$$X_B \geq 0 \; ; \; X_R \geq 0$$

cuya solución básica factible asociada con la base B es

$X = (X_B, X_R)^T = (B^{-1}b, 0)^T$. Sin embargo, para otro valor de X_R, el vector X_B es dado por $X_B = B^{-1}b - B^{-1}R\, X_R$, de donde

$$Z = C_B\,(B^{-1}b - B^{-1}R\, X_R) + C_R X_R$$

$$= C_B\, B^{-1}b + (C_R - C_B\, B^{-1}R) X_R$$

donde el valor de la función objetivo está expresado en términos del vector de variables no básicas.

2.5.1 Descripción del simplex revisado. Supóngase que B es un problema lineal y que la solución $X = (X_B, 0)^T = (B^{-1}, 0)^T$ es una solución factible.

1. Calcúlese el vector de costos relativos $\bar{C}_R = C_R - \lambda R$, donde $\lambda = C_B B^{-1}$. $\bar{C}_R \geq 0$ la solución es óptima.

2. Determínese el vector a_j que entra a la base, esto es, aquél con costo relativo más negativo. Calcúlese $y_j = B^{-1}a_j$ que expresa el vector a_j en términos de la base actual.

3. Obténgase el vector que sale de la base, esto es, si $\bar{b} = B^{-1}b$ y $y_j = B^{-1}a_j$, determínese el índice K tal que

$$\bar{b}_k/y_k = \min \{\bar{b}_i/y_i \text{ , si } y_i > 0\}$$

donde $\bar{b}_i$ (y_i) denota el i-ésimo elemento del vector $\bar{b}(y)$.

4. Actualícese la base y su inversa y regrésese a 1.

NOTA: Los teoremas cuyas demostraciones no se incluyen en este capítulo, pueden consultarse en ref. [11] ó ref. [29].

CAPITULO III

EL PROBLEMA LINEAL CON COTAS SUPERIORES GENERALIZADAS

3.1 Definición

Considere el problema lineal

$$
\begin{aligned}
&\text{minimice } z = c_0x_0 + c_1x_1 + \ldots + c_px_p \\
&D_0x_0 + D_1x_1 + D_2x_2 + \ldots + D_px_p = b_0 \\
&\qquad \mathbb{1}_1x_1 \qquad\qquad\qquad\qquad = 1 \\
&\qquad\qquad \mathbb{1}_2x_2 \qquad\qquad\qquad = 1 \qquad\qquad (P) \\
&\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\
&\qquad\qquad\qquad\qquad\qquad\qquad \mathbb{1}_px_p = 1 \\
&x_0 \geq 0 \; ; \; x_1 \geq 0 \; ; \; x_2 \geq 0 \; ; \; \ldots \; ; \; x_p \geq 0
\end{aligned}
$$

donde D_i matriz $m_0 x n_i$; $\mathbb{1}_i$, vector hilera con n_i elementos unitarios; b_0 vector columna de n_0 componentes; c_i vector hilera de n_i elementos; y x_i, vector columna de n_i variables, $i=0,\ldots p$. Este es el denominado problema lineal con cotas superiores generalizadas.

En este problema se supone que la matriz de restricciones tiene rango completo, esto es, no existen restricciones redundantes.

Uno de los primeros propósitos para el análisis y solución de un problema lineal de esta naturaleza, es la caracterización general de las matrices bases. En este respecto se tiene que :

Sea el problema lineal.

$$\text{minimice } -x_0$$

$$5x_0 + 10x_1 + 5x_2 + 15x_3 + 10x_4 + 15x_5 + 5x_6 + 10x_7 = 34$$

$$5x_0 + 10x_1 + 10x_2 + 5x_3 + 20x_4 + 15x_5 + 10x_6 + 10x_7 = 38$$

$$x_1 + x_2 = 1$$

$$x_3 + x_4 + x_5 = 1$$

$$x_6 + x_7 = 1$$

$$x_0 \geq 0 \; ; \; x_1 \geq 0 \; . \; . \; . \; . \; . \; . \; . \; . \; ; \; x_7 \geq 0$$

Con el propósito de establecer una forma general de las matrices base de un problema con cotas superiores generalizadas, considérense los siguientes ejemplos de bases asociadas con las restricciones del problema anterior.

$$B = \begin{bmatrix} 5 & 5 & 10 & 15 & 10 \\ 5 & 10 & 10 & 15 & 10 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$B = \begin{bmatrix} 5 & 10 & 5 & 15 & 10 \\ 5 & 20 & 10 & 5 & 10 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Es sencillo demostrar que, reordenando las columnas, toda base del problema lineal con cotas superiores es de la forma

$$B = \begin{bmatrix} A_0 & A_1 & A_2 & \cdots & A_p \\ & 1 & & & \\ & & 1 & & \\ V & & & \ddots & \\ & & & & 1 \end{bmatrix}$$

donde A_0 es una matriz $m_0 x m_0$; A_i, es una columna de la matriz D_i, $i=1,\ldots,p$; y V, una matriz $p x m_0$. La matriz A_0 está formada por columnas de las matrices D_i, $i=0,\ldots,p$, y las columnas de V son vectores con a lo sumo un elemento positivo e igual a uno. Nótese que si una columna de A_0 pertenece a la matriz D_i $(i=1,\ldots,p)$ la correspondiente columna de V será un vector con un solo elemento unitario en la posición i.

Dada la importancia del concepto de matriz base en un problema de programación lineal, es necesario considerar las siguientes preguntas:

a) ¿Se puede calcular B^{-1} en forma simple y eficiente?

b) ¿Se puede actualizar fácilmente B^{-1} cuando se tiene un cambio de vectores en la base?

Las siguientes secciones responden a estas preguntas.

2 El proceso de inversión de la base

Considérese la matriz

$$P = \begin{bmatrix} I_0 & A_1 & A_2 & \cdots & A_p \\ & 1 & & & \\ & & 1 & & \\ & & & & \\ & & & & 1 \end{bmatrix}$$

donde I_0 es una matriz identidad $m_0 x m_0$ y A_i, $i=1,\ldots,$ p son vectores columna que se han definido en la matriz base B. Nótese que P es no singular y que su inversa es dada como

$$P^{-1} = \begin{bmatrix} I_0 & -A_1 & -A_2 & \cdots & A_p \\ & 1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix}$$

Usando esta propiedad se puede establecer una relación entre las matrices P y B que permitan determinar en forma simple la inversa de B. Considérese la relación B = PQ donde, la matriz Q queda determinada en forma única debido a la no singularidad de B y P, específicamente $Q = P^{-1}B$ ó bien,

$$Q = \begin{bmatrix} B_T & 0 \\ V & I \end{bmatrix}$$

donde $B_T = A_0 - \begin{bmatrix} A_1, \ldots, A_p \end{bmatrix} V$ es una matriz $m_0 x m_0$, denominada la matriz de trabajo y V, la matriz definida en B, llamada la matriz lateral. Una propiedad importante de B_T es

Proposición 3.2.1 La matriz de trabajo es no singular.

Prueba. Det B_T = Det Q = Det$(P^{-1}B)$ = (Det P^{-1}) (Det B)$\neq 0$.

Es conveniente puntualizar que el cálculo de la inversa de Q se reduce casi exclusivamente al cálculo de B_T^{-1}, pues,

$$Q^{-1} = \begin{bmatrix} B_T^{-1} & 0 \\ -VB_T^{-1} & I \end{bmatrix}$$

Como consecuencia de esta discusión se concluye que la inversa B es sencilla de determinar si se conocen las matrices angulares P y Q.

3.3 El proceso de actualización de la inversa de la base.

Supóngase que hemos expresado la matriz base B del problema con cotas superiores generalizadas en la forma.

$$B = P\,Q$$

donde P y Q son matrices angulares no singulares cuyas inversas son sencillas de determinar. Supóngase que deseamos cambiar la base B por otra, que se denotará B^*, la cual difiere de la primera en un solo vector columna. La nueva base se expresa en términos de matrices angulares P^* y Q^* como

$$B^* = P^*\,Q^*$$

Aquí conviene determinar si existe alguna relación entre P y P^* ó Q y Q^*, que permita evaluar en forma simple la inversa de B^*. Primeramente se recuerda que es sencillo calcular la matriz elemental E tal que

$$B^{*-1} = E\,B^{-1}$$

esto es, el trabajo de inversión de B puede ser usado en el de B^{*-1}. Asimismo, es sencillo determinar la matriz E_p, tal que

$$P^{*-1} = E_p P^{-1}$$

En relación con las matrices Q y Q^* ó más bien respecto a las matrices de trabajo B_T y B_T^*, se tiene que sus inversas están relacionadas por medio de una fórmula que usa ciertas submatrices de las matrices elementales E y E_p, esto es,

Teorema 3.3.1 Sean E y E_p las matrices elementales que actualizan las inversas de las matrices B y P, respectivamente, esto es, $B^{*-1} = EB^{-1}$ y $P^{*-1} = E_p P^{-1}$. Considérese las matrices partidas

$$E = \begin{bmatrix} E_0 & E_1 \\ E_2 & E_3 \end{bmatrix} \quad ; \quad E_p = \begin{bmatrix} E_p^0 & E_p^1 \\ E_p^2 & E_p^3 \end{bmatrix}$$

donde E_0 y E_p^0 son matrices de orden $m_0 x m_0$ y el resto de las matrices son tales que conforman con la multiplicación de B^{-1} y P^{-1} respectivamente. Supóngase $E_1=0$ ó $E_p^2 = 0$. Entonces,

$$B_T^{*-1} = (E_0 - E_1 V) B_T^{-1} (E_p^0)^{-1}$$

Prueba. Sea $B^* = P^* \cdot Q^*$ y $B = P \cdot Q$ Entonces,

$$Q^{*-1} P^{*-1} = B^{*-1} = E B^{-1} = EQ^{-1}P^{-1}$$

De donde, usando el hecho que $E_p^{-1} = P^{-1}P^*$, se tiene

$$Q^{*-1} = E Q^{-1}P^{-1}P^* = E Q^{-1}E_p^{-1}$$

que en forma explícita es equivalente a

$$\begin{bmatrix} B_T^{*-1} & 0 \\ -VB_T^{*-1} & I \end{bmatrix} = \begin{bmatrix} E_0 & E_1 \\ E_2 & E_3 \end{bmatrix} \begin{bmatrix} B_T^{-1} & 0 \\ -VB_T^{-1} & I \end{bmatrix} \begin{bmatrix} \tilde{E}_p^0 & \tilde{E}_p^1 \\ \tilde{E}_p^2 & \tilde{E}_p^3 \end{bmatrix}$$

donde la matriz E_p^{-1} se ha partido conformablemente y sus submatrices se denotan por $\tilde{E}_p^i$, i=0, 1,2,3. Entonces B_T^{*-1} es

$$B_T^{*-1} = \begin{bmatrix} E_0, & E_1 \end{bmatrix} \begin{bmatrix} B_T^{-1} & 0 \\ VB_T^{-1} & I \end{bmatrix} \begin{bmatrix} \tilde{E}_p^0 \\ \tilde{E}_p^2 \end{bmatrix}$$

$$= \begin{bmatrix} E_0 - E_1V \end{bmatrix} B_T^{-1} \tilde{E}_p^0 + E_1\tilde{E}_p^2$$

Sin embargo, E_p es una matriz elemental de la forma

$$E_p = \begin{bmatrix} I_0 & E_p^1 \\ 0 & E_p^3 \end{bmatrix} \quad \text{ó} \quad E_p = \begin{bmatrix} E_p^0 & 0 \\ E_p^2 & I \end{bmatrix}$$

cuyas inversas, respectivamente son dadas por

$$E_p^{-1} = \begin{bmatrix} I_0 & -E_p^1 \cdot (E_p^3)^{-1} \\ 0 & (E_p^3)^{-1} \end{bmatrix} \quad \text{ó} \quad E_p^{-1} = \begin{bmatrix} (E_p^0)^{-1} & 0 \\ -E_p^2 \cdot (E_p^0)^{-1} & I \end{bmatrix}$$

Sin embargo, en cualquiera de estos casos se cumple

$$\tilde{E}_p^0 = (E_p^0)^{-1} \qquad \text{y} \qquad \tilde{E}_p^2 = -E_p^2 (E_p^0)^{-1}$$

de donde, $B_T^{*-1} = (E_0 - E_1V)B_I^{-1}(E_p^0)^{-1} - E_1E_p^2(E_p^0)^{-1}$.

Si $E_1 = 0$ ó $E_p^2 = 0$, el teorema queda demostrado.

Como aplicación de este resultado considérese el siguiente caso de actualización de la inversa de la base. Supóngase que se dispone de una base en la forma factorizada B = PQ donde P y Q son matrices angulares. Supóngase que la nueva base B* difiere de B en un vector columna el cual se encuentra asociado a la base de trabajo, esto es, un vector colocado en las primeras m_0 columnas de B. Sea la forma factorizada $B^* = P^* Q^*$, donde P^* y Q^* son matrices angulares a determinar. Se observa que en este caso P = P^* pues no hay cambio en los vectores - base asociados con las restricciones independientes, de donde la matriz elemental E_p que actualiza la inversa de P es - la identidad. Por otra parte, la matriz elemental E que actualiza la inversa de B, esto es, $B^{*-1} = E\, B^{-1}$, es de la forma

$$E = \begin{bmatrix} E_0 & 0 \\ E_2 & I \end{bmatrix}$$

pues el elemento pivote esta en una hilera de la base de - trabajo, aplicando el teorema 3.3.1 con $E_1 = 0$ se concluye que

$$B_T^{*-1} = E_0 B_T^{-1}$$

Usando esta inversa se puede calcular en forma simple la matriz inversa de Q^* y se obtiene la inversa de B^* pues las inversas de P^* y Q^* son conocidas.

Se considera ahora el caso en que la nueva matriz base B^* se obtiene de B mediante el reemplazo de una columna que no pertenece a la base de trabajo, esto es, una columna asociada a una restricción independiente. Sea $\underline{a}$ la columna de la base B que está asociada a la restricción r. Sea $\underline{d}$ el vector que reemplaza a la columna $\underline{a}$ de B para obtener B^*. Para obtener la nueva matriz B^* en la misma forma angular de la matriz B se puede proceder a reemplazar el vector $\underline{d}$ por $\underline{a}$ de la siguiente manera:

En la matriz B se intercambia la columna $\underline{a}$ por una columna asociada a la base de trabajo. A continuación se reemplaza el vector $\underline{a}$ por $\underline{d}$ usando los resultados del teorema 3.3.1. Las condiciones para efectuar el intercambio del vector $\underline{a}$ por un vector de las primeras m_0 columnas de B, esto es, un vector asociado a la base de trabajo, es que el vector hilera V_r de la matriz lateral V sea diferente de cero. La necesidad de esta condición, así como la correspondiente fórmula de actualización de la inversa de la base de trabajo, se establecen en el teorema 3.3.2

En el caso de que en la matriz B el vector $\underline{a}$ columna no pueda intercambiarse por un vector asociado a la base de trabajo, esto es, $V_r=0$, se puede reemplazar directamente el vector $\underline{a}$ por $\underline{d}$. En este caso es sencillo verificar que la base de trabajo y la matriz lateral V no cambian; pues no hay pivotes en las columnas de la matriz de trabajo.

Teorema 3.3.2 Sea a una columna en la matriz base B asociada con la r-ésima restricción independiente. Sea V_r el correspondiente vector hilera de la matriz lateral V que se supone diferente de cero. Entonces, la columna a puede intercambiarse por cualquier columna j de la base de trabajo cuyo componente en V_r es diferente de cero. Asimismo la nueva base de trabajo tiene como inversa

$$B_T^{*-1} = E_R B_T^{-1}$$

donde

$$E_R = \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & -V_r & & \\ & & & 1 & \\ & & & & \ddots \\ & & & & & 1 \end{bmatrix}$$

con V_r en la posición del vector j de la base de trabajo.

Prueba. De la identidad $Q = P^{-1}B$, donde

$$Q = \begin{bmatrix} B_T & 0 \\ V & I \end{bmatrix}$$

se tiene que un vector de la r-ésima restricción independiente puede intercambiarse por una columna de la base de trabajo si la hilera V_r de la matriz V es diferente de cero; esto es, equivalente a la suposición del teorema. Se observa que el intercambio de vectores en la base B corresponde a una simple

permutación, esto es, la matriz E tal que $B^{*-1} = E\,B^{-1}$ satisface $E^{-1} = E$. Asimismo, la matriz E_p tal que $P^{*-1} = E_p P^{-1}$ es de la forma

$$E_p = \begin{bmatrix} I & E_N^1 \\ 0 & E_N^3 \end{bmatrix}$$

pues se reemplaza un vector de la restricción convexa r. Aplicando el teorema 3.3.1 con $E_N^2 = 0$ se tiene

$$B_T^{*-1} = (E_0 - E_1 V) B_T^{-1}$$

pues en $E_N^0 = I$. Sin embargo, la forma explícita de E es

$$E = \begin{bmatrix} E_0 & E_1 \\ E_2 & E_3 \end{bmatrix} = \begin{bmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ & & & & 1 & & & \\ & & & & & 1 & & \\ & & & & & & 1 & \\ & & & & & & & 1 \end{bmatrix}$$

Nótese que $E_1 V$ es una matriz de ceros excepto por su hilera j que es igual a V_r. De aquí se tiene que $E_0 - E_1 V = E_R$

Como resumen de esta sección se puede establecer:

El proceso de actualización de la inversa de la base

0. Sea a la columna que entra a la base

1. Sea d la columna que sale de la base. Si d forma parte de la base de trabajo, continúese; de otra manera ir a 3.

2. Reemplácese d por a en la matriz de trabajo y actualícese su inversa (teorema 3.3.1). También actualícese la matriz lateral V y se termina.

3. Verifíquese si la columna a asociada con la restricción convexa r puede intercambiarse con una columna de la base de trabajo. Esto es posible cuando la hilera V_r de la matriz lateral es diferente a cero. Si hay intercambio de vectores, actualícese la matriz de trabajo y su inversa (teorema 3.3.2). También actualícese la matriz lateral V y regrese a 2. De otra manera, continúese.

4 Reemplácese d por a directamente. El vector d estará asociado a la restricción convexa r necesariamente; pues forma parte de la base asociada con esta restricción. Las matrices lateral y de trabajo no cambian en este caso. Se termina.

4 El método simplex revisado con factorización de la base.

El método simplex revisado que utiliza la técnica de factorización de la base es:

Supóngase que B es una base del problema lineal y que la solución $x = (x_B, 0) = (B^{-1}b, 0)$ es una solución básica factible.

1. Calcúlese el vector de multiplicadores simplex $\lambda = C_B B^{-1}$, donde C_B es el vector de costos en la base. Calcúlese el vector de costos relativos $\bar{C}_R = C_R - \lambda R$, donde R son las columnas no básicas. Si $\bar{C}_R \geq 0$ la solución básica actual es óptima; de otra manera continúese

2. Determínese el vector a_j que entra a la base. Calcúlese el vector $y = B^{-1}a_j$ que expresa el vector a_j en términos de los vectores que forman la base.

3. Determínese el vector que sale de la base usando la regla estándar de no violación de restricciones de no negatividad de las variables de decisión. Específicamente si $\underset{\sim}{b} = B^{-1}b$ determine el índice k tal que

$$\underset{\sim}{b}_k / y_k = \min \left\{ \underset{\sim}{b}_i / y_i \ ; \quad \text{si} \quad y_i > 0 \right\}$$

donde $\underset{\sim}{b}_i$ (y_i) denota el coeficiente i-esimo del vector $\underset{\sim}{b}(y)$. Finalmente, aplíquese el proceso de actualización de la base y regrésese nuevamente al paso 1.

En la práctica no siempre se dispone de una solución factible básica del problema lineal. En tal caso se puede obtener una o bien demostrar que no existe usando:

El proceso de obtención de una base factible básica.

a. Determínese una solución factible básica de la restricción

$$\mathbb{1}_i x_i = 0 \qquad x_i \geq 0$$

donde $i=1,\ldots,p$. Si alguna restricción no tiene solución factible básica se termina: el problema original no tiene solución factible básica. De otra manera sean x_i^E, $i=1,\ldots,p$, las correspondientes soluciones factibles básicas.

b. Hágase $x_0=0$ y $q=\sum_{i=1}^{n} D_i x_i^B - b$. Defínase la matriz diagonal $\underset{\sim}{D}$ de orden $m_0 x m_0$ tal que $d_{ii}=-1$ si $q_i \geq 0$; y, $d_{ii}=1$ si $q_i<0$.

Resuélvase el problema con cotas superiores generalizadas.

$$\min \underset{\sim}{z} = w_1+w_2+\ldots+w_p$$

$$\underset{\sim}{D}w + \sum_{i=0}^{p} D_i x_i = b$$

$$\mathbb{1}_i \cdot x_i = 1 \quad i=1,2,\ldots,p.$$

$$w \geq 0 \; ; \; x_i \geq 0 \quad i=0,\ldots,p.$$

usando el método descrito anteriormente. Nótese que este problema tiene una solución factible básica, específicamente, $(w,0, x_1^B,\ldots,x_p^B)$. Si la solución óptima z^* es cero, la base asociada con esta solución es una base factible básica del problema original. De otra manera se concluye que no existe base factible básica.

CAPITULO IV

EL PROBLEMA LINEAL ANGULAR CON RESTRICCIONES DE LIGA

4.1 Definición

Considérese el problema lineal

$$\text{minimice } z = c_0x_0 + c_1x_1 + c_2x_2 + \ldots + c_px_p$$

$$D_0x_0 + D_1x_1 + D_2x_2 + \ldots + D_px_p = b_0$$

$$E_1x_1 = b_1$$

$$E_2x_2 = b_2$$

$$\ldots$$

$$E_px_p = b_p$$

$$x_0 \geq 0 \; ; \; x_1 \geq 0 \; ; \; \ldots \; ; \; x_p \geq 0$$

Donde D_i matriz $m_0 x n_i$; E_i, matriz $m_i x n_i$; b_i, vector columna de m_i elementos; c_i, vector hilera de n_i elementos; y x_i, vector - columna de n_i variables, $i=1,\ldots,p$. Este es el problema lineal angular con restricciones de liga.

En este problema se supone que la **matriz de restricciones** tiene rango completo, esto es, no hay restricciones redundantes. En el análisis y solución de este problema se denota por B_i a cada submatriz de E_i de orden $m_i x m_i$ que es no singular. También se denota por A_i la submatriz de D_i de orden $m_0 x m_i$ asociada con las columnas de B_i, $i=1,\ldots,p$. Por otra parte, se denota por I_i la matriz identidad de orden $m_i x m_i$ $i=0, 1, \ldots, p$.

Considérese la forma general de las bases asociadas a las restricciones del problema anterior. Si B es una base, se puede reordenar las columnas de la misma para tener

$$B = \begin{bmatrix} (A_{00}, & A_{01}, & A_{02} \cdots & A_{0p}) & A_1, & A_2, & \cdots, & A_p \\ (0 & E_{11} & 0 \cdots & 0) & B_1 & & & \\ (0 & 0 & E_{22} \cdots & 0) & & B_2 & & \\ \multicolumn{8}{c}{\text{---------- ---------- ---------- ---------- ----------}} \\ (0 & 0 & 0 & E_{pp}) & & & & B_p \end{bmatrix}$$

donde A_{0q} es una matriz formada por columnas de D_q, $q=0,1,\ldots,p$; las matrices E_{ii} y B_i, donde B_i es no singular, están formadas por columnas de E_i; y, A_i es una submatriz de D_i, $i=1,\ldots,p$. Las columnas de las matrices A_i y B_i están relacionadas, pues forman parte de las columnas asociadas con las i-ésimas restricciones del problema anterior.

Es sencillo verificar que la suposición de rango completo de la matriz de restricciones del problema original implica que cada una de las submatrices de las restricciones independientes, esto es, la matriz E_i, es de rango completo. Esto a su vez implica que cada matriz E_i tiene una submatriz no singular B_i, que es la que, en particular, aparece en la matriz base.

Los procesos de inversión y actualización de la base de este problema son tratados en las siguientes secciones.

4.2 El proceso de inversión de la base

Considérese la matriz angular

$$P = \begin{bmatrix} I_0 & A_1 & A_2 & \cdots & A_p \\ & B_1 & & & \\ & & B_2 & & \\ & & & \ddots & \\ & & & & B_p \end{bmatrix}$$

donde I_0 es una matriz de orden $m_0 x m_0$ y A_i, B_i, $i=1, \ldots, p$, -- son las matrices definidas en la matriz base B. Se observa que P es una matriz no singular, pues las submatrices que forman la diagonal son no singulares. En particular, se tiene que

$$P^{-1} = \begin{bmatrix} I_0 & -\bar{A}_1 & -\bar{A}_2 & \cdots & -\bar{A}_p \\ & B_1^{-1} & & & \\ & & B_2^{-1} & & \\ & & & \ddots & \\ & & & & B_p^{-1} \end{bmatrix}$$

donde $\bar{A}_i = A_i B_i^{-1}$. Esta propiedad motiva a considerar una relación entre las matrices P y B que permita determinar en forma simple la inversa de B. Considérese la relación $B = P\,Q$ donde Q queda determinada pues $Q = P^{-1}B$ ó bien, explícitamente,

$$Q = \begin{bmatrix} B_T & 0 & 0 & \cdots & 0 \\ V_1 & I_1 & & & \\ V_2 & & I_2 & & \\ \vdots & & & \ddots & \\ V_P & & & & I_P \end{bmatrix}$$

donde las matrices B_T y V_i, $i=1,\ldots, p$ son dadas como

$$B_T = (A_{00}, A_{01}, \ldots, A_{0p}) - \sum_{i=1}^{p} A_i B_i^{-1} \quad (0,\ldots,E_{ii},\ldots 0)$$

$$V_i = B_i^{-1} \quad (0,\ldots, E_{ii},\ldots, 0) \qquad i = 1, \ldots, p.$$

La matriz B_T se denomina matriz de trabajo y es sencillo demostrar que es una matriz no singular. La matriz V, formada por las submatrices $V_1, V_2, \ldots, V_p$, es denominada matriz lateral. Una forma compacta de Q y su inversa es

$$Q = \begin{bmatrix} B_T & 0 \\ V & I \end{bmatrix} \qquad Q^{-1} = \begin{bmatrix} B_T^{-1} & 0 \\ -VB_T^{-1} & I \end{bmatrix}$$

Consecuentemente se puede determinar la inversa de B en forma simple usando el producto $B = PQ$ donde P y Q son matrices angulares cuyas inversas son sencillas de calcular.

4.3 El proceso de actualización de la base

Supóngase que la matriz base B del problema lineal angular con restricciones de liga está en la forma

$$B = P\,Q$$

donde P y Q son matrices angulares no singulares. Supóngase que cambiamos un vector $\underline{a}$ de la base B por un vector $\underline{d}$ para obtener una nueva base; que se denota por B^*, la cual difiere de la primera en un solo vector columna. La matriz B^* puede expresarse en términos de matrices angulares no singulares P^* y Q^* como

$$B^* = P^*Q^*$$

Con el propósito de determinar las nuevas matrices P^* y Q^* es conveniente analizar los siguientes cambios de vectores.

Caso i. El vector $\underline{a}$ forma parte de la base de trabajo.
En este caso es sencillo verificar que $P^*=P$ pues no hay cambio en las bases asociadas con los bloques de restricciones independientes. Por otra parte, la matriz Q^* tiene una matriz de trabajo dado por

$$B_T^{*-1} = E_0 B_T^{-1}$$

debido al teorema siguiente, donde se sabe que $E_1 = 0$, pues el pivoteo se tiene en una columna asociada con la base de trabajo y, $E_p^0 = I_0$, pues la matriz elemental E_p que actualiza P es la identidad

Teorema 4.3.1 Sean E y E_p las matrices elementales que actualizan las inversas de las matrices B y P, respectivamente, esto es, $B^{*-1} = EB^{-1}$ y $P^{*-1} = E_pP^{-1}$. Considérese las particiones

$$E = \begin{bmatrix} E_0 & E_1 \\ E_2 & E_3 \end{bmatrix} \quad ; \quad E_p = \begin{bmatrix} E_p^0 & E_p^1 \\ E_p^2 & E_p^3 \end{bmatrix}$$

donde E_0 y E_p^0 son matrices de orden $m_0 x m_0$ y el resto de las matrices son tales que conforman con la multiplicación de B^{-1} y P^{-1}, respectivamente. Si $E_1 = 0$ ó $E_p = 0$, entonces

$$B_T^{*-1} = (E_0 - E_1V)B_T^{-1} (E_p^0)^{-1}$$

Prueba. Ver teorema 3.3.1

En el caso i la determinación de Q^{*-1} es sencilla pues

$$Q^{*-1} = \begin{bmatrix} B^{*-1} & 0 \\ -V^*B_T^{*-1} & I \end{bmatrix}$$

Caso ii. La columna a forma parte de la base de un block i.

En este caso conviene, si es posible, intercambiar a por un vector asociado a la base de trabajo. Las condiciones analíticas de este intercambio son:

Teorema 4.3.2 Sea a una columna en la base asociada con la hilera r del block de restricciones i. Sea V_r el correspondiente vector hilera de la matriz lateral V, el cual se supone diferente de cero. Entonces, la columna a puede intercambiarse por cualquier columna j de la base de trabajo cuyo componente en V_r sea diferente de cero, para formar una nueva base del block i y una nueva base de trabajo cuya inversa es

$$B_T^{*-1} = E_R B_T^{-1}$$

donde

$$E_R = \begin{bmatrix} 1 & & & \\ & \ddots & & \\ & & V_r & \\ & & & \ddots \\ & & & & 1 \end{bmatrix}$$

con V_r en la posición del vector j de la base de trabajo.

Prueba. Excepto por detalles, es la misma que la del teorema 3.3.2

Si en el caso ii hay intercambio de vectores es necesario también actualizar la matriz lateral V. Si no hay intercambio de vectores se reemplaza el vector d por a directamente y se actualiza la inversa del block i junto con la matriz lateral V.

Como resumen de la discusión anterior se puede establecer:

El proceso de actualización de la inversa de la base

0. Sea a la columna que entra a la base

1. Sea d la columna que sale de la base. Si d forma parte de la base de trabajo, se continúa; de otra manera ir a 3.

2. Se reemplaza d por a en la matriz de trabajo y se actualiza su correspondiente inversa (teorema 4.3.1). Se actualiza la matriz lateral V y se termina.

3. Se verifica si la columna d puede intercambiarse por una de las que forman la matriz de trabajo. Esto es posible cuando la hilera r del block asociado con la columna básica d tiene un elemento diferente de cero en V, esto es, V_r diferente de cero. Si hay intercambio de vectores se actualiza la matriz de trabajo, su inversa (teorema 4.3.2) y la matriz lateral V. Se regresa a 2; de otra manera continúese

4. Se reemplaza d por a en la base del block correspondiente, pues d y a están en el mismo block. Se actualiza la inversa del block y la matriz lateral V. La matriz de trabajo no cambia. Se termina.

4 El método simplex revisado con factorización de la base.

Una propiedad fundamental de los problemas lineales es que sus correspondientes soluciones óptimas son puntos extremos del conjunto convexo formado por los vectores que satisfacen las restricciones. Estos puntos extremos son un número finito y su caracterización analítica es equivalente al concepto de soluciones factibles básicas. Consecuencia de esta propiedad es que la búsqueda de soluciones óptimas del problema lineal puede realizarse jerarquizando las soluciones factibles básicas de acuerdo a los valores de la función objetivo. A este respecto, los métodos de solución del problema lineal son simplemente métodos constructivos de bases factibles básicas, sin repetición, cuyo valor de la función objetivo converge al valor óptimo.

Los diferentes métodos de programación lineal tienen dos aspectos que los caracterizan: (i) la selección del par de vectores que entra y sale de la base y (ii) la actualización de la inversa de la base. La primera característica se denomina estrategia y la última proceso de actualización de la base. En los métodos tipo simplex, la selección del vector que entra a la base se obtiene por el criterio de costos relativos más negativo y la selección del vector que sale es dada por la regla estándar de no violación de las restricciones de no negatividad. Sin embargo, la actualización de la inversa de la nueva base se obtiene por medio del método de pivoteo.

En el caso de los métodos de programación lineal con descomposición existe una variedad de estrategias, las primales - (del tipo simplex), las duales, las de cálculo parcial de costos reducidos y otras. Asimismo, se tienen diferentes procesos de actualización de la inversa de la base, por ejemplo la usual del método simplex, la del simplex revisado, la de factorización de la base y otras más.

El método simplex revisado con factorización de la base aplicado al problema de esta sección es el mismo que el desarrollado para el caso del problema con cotas superiores generalizadas. Sin embargo, es importante establecer en detalle algunas de las ventajas computacionales que se tiene cuando se - efectua la selección del vector que entra y del que sale de la base El análisis de esta selección consiste de los siguientes pasos:

a. La transformación hacia atrás
b. La transformación hacia adelante
c. La determinación del vector que entra a la base
d. La determinación del vector que sale de la base

En el desarrollo de este análisis se supone que la base B del problema lineal angular con restricciones de liga ha sido expresado en su forma producto, esto es,

$$B = P\,Q$$

donde P y Q son matrices angulares no singulares.

4.5 La transformación hacia atrás

Considérese la operación

$$\lambda B = C_B$$

donde λ representa el vector de multiplicadores simplex ó precios sombra y C_B es el vector de costos asociados con las variables que estan en la base. Esta operación se denomina transformación hacia atrás y es una de las que más tiempo consumen en la aplicación de los métodos comerciales de la programación lineal. Usando la forma factorizada $B = PQ$ donde P y Q son matrices angulares se tiene que,

$$\lambda = C_B B^{-1} = C_B Q^{-1} P^{-1}$$

Sin embargo, debido a la forma independiente como se manejan las matrices Q^{-1} y P^{-1} es conveniente calcular λ usando

$$\text{i.} \quad \underset{\sim}{C} = C_B Q^{-1}$$

$$\text{ii.} \quad \lambda = \underset{\sim}{C} P^{-1}$$

Sin embargo, si los vectores C_B, $\underset{\sim}{C}$ y λ se han partido en forma tal que corresponden a las restricciones de liga y a cada uno de los bloques de restricciones independientes, por ejemplo, $C_B = (C_0, C_1, \ldots, C_p)$ y $\lambda = (\lambda_0, \ldots, \lambda_p)$, entonces se puede observar que la forma explícita de la operación i, es equivalente a

$$(\underset{\sim}{C}_0, \underset{\sim}{C}_1, \ldots, \underset{\sim}{C}_p) = (C_0, C_1, \ldots, C_p) \begin{bmatrix} B_T^{-1} & & & \\ -V_1 & I_1 & & \\ -V_2 & & I_2 & \\ \vdots & & & \ddots \\ -V_p & & & I_p \end{bmatrix}$$

donde $\bar{V}_i = V_i B_T^{-1}$ $i=1,\ldots,p$. Consecuentemente se tiene que

$$\underset{\sim}{C}_0 = C_0 B_T^{-1} - \sum_{i=1}^{p} C_i V_i B_T^{-1} \quad ; \quad \underset{\sim}{C}_i = C_i \quad i = 1, \ldots, p.$$

Por otra parte, la operación ii puede expresarse como $\lambda P = \underset{\sim}{C}$ cuya forma explícita, en términos de ecuaciones es

$$\lambda_0 = \underset{\sim}{C}_0 \quad ; \quad \lambda_0 A_i + \lambda_i B_i = \underset{\sim}{C}_i \quad i = 1, \ldots, p$$

ó bien, despejando ecuaciones, se encuentra que

$$\lambda_0 = \underset{\sim}{C}_0 \quad ; \quad \lambda_i = (\underset{\sim}{C}_i - \underset{\sim}{C}_0 A_i) B_i^{-1} \quad i = 1, \ldots, p.$$

Un caso importante de las operaciones i y ii es:

Proposición 4.5.1 Supóngase que $C_B=(C_0,C_1,\ldots,C_p)$, satisface $C_i=0$, $i=1,\ldots,p$. Entonces, la operación $\lambda=C_B B^{-1}$ es dada por

$$\lambda_0 = C_0 B_T^{-1}$$

$$\lambda_i = -\lambda_0 A_i B_i^{-1} \qquad I = 1, \ldots, p.$$

Prueba. Se sigue de la discusión de esta sección.

4.6 La transformación hacia adelante

Se considera ahora la operación

$$By = a$$

donde a es un vector columna de la forma

$$a' = \left[a_0', 0, \ldots, a_i', 0, \ldots, 0 \right]$$

esto es, los únicos componentes que pueden ser diferentes de cero son aquéllos que están asociados con las hileras de las restricciones de liga y con, a lo sumo, un block de restricciones independientes. Esta operación se denomina la transformación hacia adelante, y es necesaria para determinar el vector que sale de la base en cada iteración del simplex.

Usando la forma producto de la matriz B, se tiene que

$$y = B^{-1}a = Q^{-1}p^{-1}a$$

donde P y Q son matrices angulares. Explícitamente,

$$y = \left[\begin{array}{cc} B_T^{-1} & \\ -\bar{V}_1 & I_1 \\ \hline -\bar{V}_i & I_i \\ \hline -V_p & I_p \end{array}\right] \left[\begin{array}{ccc} I_0 & & -\bar{A}_i \\ & I_1 & \\ \hline & & \bar{B}_i^{-1} \\ \hline & & I_p \end{array}\right] \left[\begin{array}{c} a_0 \\ 0 \\ \hline a_i \\ \hline 0 \end{array}\right]$$

donde $\bar{V}_i = V_i B_T^{-1}$ y $\bar{A}_i B_i^{-1}$ $\quad i=1, \ldots, p.$

Consecuentemente se tiene

$$y = \begin{bmatrix} B_T^{-1} & & & \\ -\bar{V}_1 & I_1 & & \\ \hline -\bar{V}_i & & I_i & \\ \hline -\bar{V}_p & & & I_p \end{bmatrix} \begin{bmatrix} a_0 & -\bar{A}_i a_i \\ \hline 0 & \\ \hline B_i^{-1} a_i & \\ \hline 0 & \end{bmatrix}$$

$$= \begin{bmatrix} B_T^{-1} H_i \\ \hline -\bar{V}_1 H_i \\ \hline -\bar{V}_i H_i + B_i^{-1} a_i \\ \hline -V_p H_i \end{bmatrix}$$

donde $H_i = a_0 - \bar{A}_i a_i$. De aquí se observa que la operación $By = a$ puede efectuarse en forma sencilla si se conoce la forma producto $B = PQ$, donde P y Q son matrices angulares. Finalmente, se puntualiza que si el vector y es no positivo para alguna columna a no básica, entonces el problema lineal original es no acotado.

4.7 Determinación del vector que entra a la base

Sea λ el vector de variables duales o multiplicadores simplex. Usando λ se puede determinar el vector columna que entra a la base si se calcula el vector de costos relativos.

$$\bar{C}_R = C_R - \lambda R$$

donde R y C_R representan la matriz y vector de costos de las variables no básicas. Si $\bar{C}_R \geq 0$ la solución factible básica que se dispone es óptima. De otra manera, se selecciona una columna con coeficiente de costos relativos menor que cero.

4.8 Determinación del vector que sale de la base

Sea $a' = \left[a'_0, 0, \ldots, a'_i, \ldots, 0\right]$ el vector que entra a la base. Este vector expresado en términos de la base B, es $y=B^{-1}a$. Nótese que la forma explícita de y cuando B se maneja en la forma producto $B = PQ$, se estableció al analizar la transformación hacia adelante. Consecuentemente, si b es el vector de recursos del problema lineal original, la selección del vector que sale de la base se hace comparando el vector y con $\bar{b} = B^{-1}b$ de acuerdo al criterio del método simplex.

Finalmente, se puede establecer que el método simplex revisado - con factorización de la base del problema lineal con cotas superiores generalizadas se aplica también al problema analizado. El proceso de obtención de una base inicial es también similar.

4.9 EJEMPLO. Considérese el problema lineal:

$$\text{maximice } Z = x_1 + 3x_2 + x_3 + x_4 + 3x_5 + 2x_6 + x_7 + 2x_8 + 3x_9$$

s.a

$$\begin{aligned}
4x_1 + 4x_2 + x_3 + 2x_5 + 4x_6 + 2x_7 + x_8 + 2x_9 &= 22 \\
2x_2 + x_4 + 4x_5 + 3x_6 + x_7 &= 10 \\
x_3 + x_4 + x_5 &= 8 \\
x_6 + x_7 + x_8 &= 6 \\
x_7 + x_9 &= 4
\end{aligned}$$

$$x_i \geq 0 \ , \ i=1,\ldots,9$$

cuyo tableau de las restricciones es:

a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	b
4	4	1	0	2	4	2	1	2	22
0	2	0	1	4	3	1	0	0	10
0	0	1	1	1	0	0	0	0	8
0	0	0	0	0	1	1	1	0	6
0	0	0	0	0	0	1	0	1	4

Se puede observar fácilmente que este problema es un problema lineal angular con restricciones de liga, por lo que se puede escribir como:

$$\text{maximice } Z = C_0x_0 + C_1x_1 + C_2x_2$$

s.a.

$$D_0x_0 + D_1x_1 + D_2x_2 = b_0$$

$$E_1x_1 = b_1$$

$$E_2x_2 = b_2$$

$$x_0 \geq 0 \ , \ x_1 \geq 0 \ , \ x_2 \geq 0$$

donde

$$C_0 = [1,3] \ ; \ C_1 = [1,1,3] \ ; \ C_2 = [2,1,2,3]$$

$$x_0 = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \ ; \ x_1 = \begin{bmatrix} x_3 \\ x_4 \\ x_5 \end{bmatrix} \ ; \ x_2 = \begin{bmatrix} x_6 \\ x_7 \\ x_8 \\ x_9 \end{bmatrix}$$

$$E_1 = [1,1,1] \ ; \ E_2 = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

$$D_0 = \begin{bmatrix} 4 & 4 \\ 0 & 2 \end{bmatrix} \ ; \ D_1 = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 4 \end{bmatrix} \ ; \ D_2 = \begin{bmatrix} 4 & 2 & 1 & 2 \\ 3 & 1 & 0 & 0 \end{bmatrix}$$

$$b_0 = \begin{bmatrix} 22 \\ 10 \end{bmatrix} \ ; \ b_1 = [8] \ ; \ b_2 = \begin{bmatrix} 6 \\ 4 \end{bmatrix}$$

Una base para este problema es : $B = [a_3, a_4, a_5, a_8, a_9]$, esto es,

$$B = \begin{bmatrix} 1 & 0 & 2 & 1 & 2 \\ 0 & 1 & 4 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

donde las matrices A_i, B_i, i=1,2 son fácilmente identificables. Asimismo, la correspondiente matriz P y su inversa son:

$$P = \begin{bmatrix} 1 & 0 & 2 & 1 & 2 \\ 0 & 1 & 4 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad P^{-1} = \begin{bmatrix} 1 & 0 & -2 & -1 & -2 \\ 0 & 1 & -4 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Usando la propiedad de que PQ = B se tiene que la matriz Q y su inversa son, respectivamente:

$$Q = \begin{bmatrix} -1 & -2 & 0 & 0 & 0 \\ -4 & -3 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad Q^{-1} = \begin{bmatrix} 3/5 & -2/5 & 0 & 0 & 0 \\ -4/5 & 1/5 & 0 & 0 & 0 \\ 1/5 & 1/5 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Consecuentemente, la inversa de la base, $B^{-1} = Q^{-1}P^{-1}$ será:

$$B^{-1} = \begin{bmatrix} 3/5 & -2/5 & 2/5 & -3/5 & -6/5 \\ -4/5 & 1/5 & 4/5 & 4/5 & 8/5 \\ 1/5 & 1/5 & -1/5 & -1/5 & -2/5 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Nótese que en este caso la matriz de trabajo y su inversa son:

$$B_T = \begin{bmatrix} -1 & -2 \\ -4 & -3 \end{bmatrix} \qquad B_T^{-1} = \begin{bmatrix} 3/5 & -2/5 \\ -4/5 & 1/5 \end{bmatrix}$$

es fácil observar que se obtienen los mismos valores para B_T y V, aplicando los resultados obtenidos en la sección 4.1, esto es,

$B_T = (A_{00}, A_{01}, \ldots, A_{01}) - \sum_{i=1}^{p} A_i B_i^{-1}(0,\ldots,E_{ii},\ldots,0)$ donde

para nuestro caso p=2. Entonces, resulta:

$$B_T = A_{01} - A_1B_1^{-1}E_{11} - A_2B_2^{-1}E_{22} =$$

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 2 \\ 4 \end{bmatrix} [1] [1\ ,\ 1] - \begin{bmatrix} 1 & 2 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} [0,\ 0] = \begin{bmatrix} -1 & -2 \\ -4 & -3 \end{bmatrix}$$

y por otra parte, se tiene que $V_i = B_i^{-1}(0,\ldots,E_{ii},\ldots,0)$, i=1,...,p, quedando:

$$V_1 = [1] \quad [1,\ 1] = [1,\ 1]$$

$$V_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Por lo que

$$V = \begin{bmatrix} 1 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Iteración 1. $B = [a_3, a_4, a_5, a_8, a_9]$

$C_B = [1, 1, 3, 2, 3]$; $\lambda = C_B B^{-1} = [2/5, 2/5, 3/5, 8/5, 11/5]$

Vectores no básicos: a_1, a_2, a_6 y a_7

$\lambda a_1 = 8/5 \qquad C_1 - \lambda a_1 = 1 - 8/5 = -3/5$

$\lambda a_2 = 12/5 \qquad C_2 - \lambda a_2 = 3 - 12/5 = 3/5$

$\lambda a_6 = 22/5 \qquad C_6 = \lambda a_6 = 2 - 22/5 = -12/5$

$\lambda a_7 = 5 \qquad C_7 = \lambda a_7 = 1 - 5 = -4$

De donde, el vector que entra a la base es a_2.

Se determinará ahora el vector que sale de la base. Sea:

$$B^{-1}a_2 = \begin{bmatrix} 8/5 \\ -14/5 \\ 6/5 \\ 0 \\ 0 \end{bmatrix} ; \quad B^{-1}b = \begin{bmatrix} 4 \\ 2 \\ 2 \\ 6 \\ 4 \end{bmatrix}$$

$$\min \{4/8/5 , - , 2/6/5 , - , -\} = 5/3$$

consecuentemente, sale de la base el vector a_5.

Proceso de actualización de la base.

Con el propósito de mantener la forma producto es necesario efectuar dos operaciones que quedan resumidas como:

a. $B = [a_3, a_4, a_5, a_8, a_9]$ a $B^* = [a_3, a_5, a_4, a_8, a_9]$

b. $B = [a_3, a_5, a_4, a_8, a_9]$ a $B^* = [a_3, a_2, a_4, a_8, a_9]$

Operación a. $B = [a_3, a_1, a_5, a_8, a_9]$ a $B^* = [a_3, a_5, a_4, a_8, a_9]$

Las nuevas matrices P^* y P^{*-1} son:

$$P^* = \begin{bmatrix} 1 & 2 & 0 & 1 & 2 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad P^{*-1} = \begin{bmatrix} 1 & 0 & 0 & -1 & -2 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Asimismo, las matrices B^* y $Q^* = P^{*-1}B^*$ son dadas por

$$B^* = \begin{bmatrix} 1 & 2 & 0 & 1 & 2 \\ 0 & 4 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad ; \quad Q^* = \begin{bmatrix} 1 & 2 & 0 & 0 & 0 \\ -1 & 3 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Nótese que la matriz de trabajo y su inversa son

$$B_T^* = \begin{bmatrix} 1 & 2 \\ -1 & 3 \end{bmatrix} \quad ; \quad B_1^{*-1} = \begin{bmatrix} 3/5 & -2/5 \\ 1/5 & 1/5 \end{bmatrix}$$

Aquí es conveniente puntualizar que B_T^{*-1} puede calcularse en forma directa de la fórmula del teorema 4.2, esto es

$$B_T^{*-1} = \begin{bmatrix} 1 & 0 \\ -V_i & \end{bmatrix} \begin{bmatrix} B_T^{-1} \\ \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -1 & -1 \end{bmatrix} \begin{bmatrix} 3/5 & -2/5 \\ -4/5 & 1/5 \end{bmatrix} = \begin{bmatrix} 3/5 & -2/5 \\ 1/5 & 1/5 \end{bmatrix}$$

Se puede observar que para obtener la inversa de B_T^* en esta forma, no es necesario evaluar explícitamente B_T^*. Asimismo,

conociendo la inversa de B_T^*, se pueden evaluar las inversas de Q* y B*.

$$Q^{*-1} = \begin{bmatrix} 3/5 & -2/5 & 0 & 0 & 0 \\ 1/5 & 1/5 & 0 & 0 & 0 \\ -4/5 & 1/5 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} ; \quad B^{*-1} = \begin{bmatrix} 3/5 & -2/5 & 2/5 & -3/5 & -6/5 \\ 1/5 & 1/5 & -1/5 & -1/5 & -2/5 \\ -4/5 & 1/5 & 4/5 & 4/5 & 8/5 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Operación b. $B = [a_3, a_5, a_4, a_8, a_9]$ a $B^* = [a_3, a_2, a_4, a_8, a_9]$

Las nuevas matrices P^* y P^{*-1} son las mismas matrices que en la operación a, pues no hay cambio en las bases de los bloques. Por otra parte, las matrices B^* y $Q^* = P^{*-1}B^*$ son:

$$B^* = \begin{bmatrix} 1 & 4 & 0 & 1 & 2 \\ 0 & 2 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} ; \quad Q^* = \begin{bmatrix} 1 & 4 & 0 & 0 & 0 \\ -1 & 2 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

De aquí se observa que la matriz de trabajo y su inversa son

$$B_T^* = \begin{bmatrix} 1 & 4 \\ -1 & 2 \end{bmatrix} ; \quad B_T^{*-1} = \begin{bmatrix} 1/3 & -2/3 \\ 1/6 & 1/6 \end{bmatrix}$$

Nuevamente se puntualiza que la inversa de la matriz base B_T^* puede determinarse por medio del teorema 4.1. Específicamente, calculemos las matrices elementales E y E_p, donde primero

necesitamos determinar

$$B^{-1}a_2 = \begin{bmatrix} 3/5 & -2/5 & 2/5 & -3/5 & -6/5 \\ 1/5 & 1/5 & -1/5 & -1/5 & -2/5 \\ -4/5 & 1/5 & 4/5 & 4/5 & 8/5 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 2 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 8/5 \\ 6/5 \\ -14/5 \\ 0 \\ 0 \end{bmatrix}$$

por lo que el vector eta valdrá

$$\begin{bmatrix} -8/5/6/5 \\ 1/6/5 \\ 14/5/6/5 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -4/3 \\ 5/6 \\ 7/3 \\ 0 \\ 0 \end{bmatrix}$$

De donde se tiene que E y E_p son dadas como

$$E = \left[\begin{array}{cc} E_0 & E_1 \\ \hline E_2 & E_3 \end{array}\right] = \left[\begin{array}{cc|ccc} 1 & -4/3 & 0 & 0 & 0 \\ 0 & 5/6 & 0 & 0 & 0 \\ \hline 0 & 7/3 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

$$E_p = \left[\begin{array}{cc} E_p^0 & E_p^1 \\ \hline E_p^2 & E_p^3 \end{array}\right] = \left[\begin{array}{cc|ccc} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

Aplicando la fórmula del teorema 4.1, se tiene que

$$B_T^{*-1} = [E_0 - E_1 \; V] \; B_T^{-1}(E_p^0)^{-1} = E_0 B_T^{-1}$$

pues $E_1 = 0$ y $E_p^0 = I$. Consecuentemente,

$$B_T^{*-1} = \begin{bmatrix} 1 & -4/3 \\ 0 & 5/6 \end{bmatrix} \begin{bmatrix} 3/5 & -2/5 \\ 1/5 & 1/5 \end{bmatrix} = \begin{bmatrix} 1/3 & -2/3 \\ 1/6 & 1/6 \end{bmatrix}$$

que es exactamente la matriz especificada con anterioridad y obtenida directamente. Nótese, sin embargo, que por medio del teorema 4.1 se obtiene B_T^{*-1}, sin construir explícitamente B_T^*. De la información obtenida de B_T^{*-1}, se puede construir Q^{*-1}, esto es,

$$-VB_T^{*-1} = -\begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1/3 & -2/3 \\ 1/6 & 1/6 \end{bmatrix} = \begin{bmatrix} -1/3 & 2/3 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

lo que implica

$$Q^{*-1} = \begin{bmatrix} 1/3 & -2/3 & 0 & 0 & 0 \\ 1/6 & 1/6 & 0 & 0 & 0 \\ -1/3 & 2/3 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Consecuentemente, $B^{*-1} = Q^{*-1}P^{*-1}$ es dado por

$$B^{*-1} = \begin{bmatrix} 1/3 & -2/3 & 2/3 & -1/3 & -2/3 \\ 1/6 & 1/6 & -1/6 & -1/6 & -1/3 \\ -1/3 & 2/3 & 1/3 & 1/3 & 2/3 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Iteración 2. $B = [a_3, a_2, a_4, a_8, a_9]$

$C_B = [1, 3, 1, 2, 3]$; $\lambda = C_B B^{-1} = [1/2, 1/2, 1/2, 3/2, 2]$

Vectores no básicos: a_1, a_5, a_6 y a_7

$\lambda a_1 = 2$ $\quad C_1 - \lambda a_1 = -1$

$\lambda a_5 = 7/2$ $\quad C_5 = \lambda a_5 = -1/2$

$\lambda a_6 = 5$ $\quad C_6 - \lambda a_6 = -3$

$\lambda a_7 = 5$ $\quad C_7 = \lambda a_7 = -4$

Consecuentemente, la base actual es óptima. Entonces,

$$B^{-1}b = \begin{bmatrix} 4/3 \\ 5/3 \\ 20/3 \\ 6 \\ 4 \end{bmatrix}$$

De donde la solución óptima es

$Z^* = 37$; $x_1^* = 0$; $x_2^* = 5/3$; $x_3^* = 4/3$; $x_4^* = 20/3$

$x_5^* = 0$; $x_6^* = 0$; $x_7^* = 0$; $x_8^* = 6$; $x_9^* = 4$

APENDICE A

En este apéndice se establecen algunos conceptos, resultados e implicaciones básicas de convexidad en la teoría de la optimización.

DEFINICION A1.- Si x, y son elementos del espacio euclidiano R^n, se dice que la línea que une a x con y es el conjunto de elementos $\{Z \in R^n : Z = \alpha x + (1-\alpha)y,\ 0 \leq \alpha \leq 1\}$. Un conjunto C contenido en R^n es convexo si dados cualquier x, y $\in$ C la línea que los une está en C.

Como ejemplo de un conjunto convexo, se tiene el conjunto S definido por

$$S = \{x : Ax \geq b,\ x \geq 0\}$$

es decir, el conjunto de soluciones no negativas del sistema $Ax \geq b$, donde A es una matriz de m renglones y n columnas de rango m, x es un vector columna de n componentes y b es un vector columna de n componentes. Para demostrar que se trata de un conjunto convexo, se debe exhibir que para cualquier par de puntos del conjunto, el segmento de recta que los une también está en el conjunto.

Sean x_1, x_2 dos puntos de S y α en $]0,1[$, entonces

$$Ax_1 \geq b\ ;\ x_1 \geq 0 \quad : \quad Ax_2 \geq b\ ;\ x_2 \geq 0$$

$$\alpha Ax_1 \geq \alpha b\ ;\ \alpha x_1 \geq 0 \qquad (1-\alpha)Ax_2 \geq (1-\alpha)b\ ;\ (1-\alpha)x_2 \geq 0$$

sumando ambas desigualdades y agrupando se obtiene

$$A(\alpha x_1 + (1-\alpha)x_2) \geq b \qquad \alpha x_1 + (1-\alpha)x_2 \geq 0$$

De donde S es convexo.

Una de las propiedades elementales de conjuntos convexos es:

Proposición A.1 Sea {An} una colección de conjuntos convexos en R^n.

Entonces $C = \bigcap_n An$ es un conjunto convexo.

Prueba. Sean x, y ε C. Entonces x,y ε An para cada n. Usando la convexidad de An se tiene que $Z = \alpha x + (1-\alpha)$ y ε An donde $0 \leq \alpha \leq 1$.

Por lo tanto, ZεC y se concluye que C es convexo.

DEFINICION A.2.- Se dice que Z es una combinación convexa de los m elementos $x_1, x_2, \ldots, x_m$ del espacio R^n, si

$$Z = \sum_{i=1}^{m} \alpha_i x_i \text{ donde } \sum_{i=1}^{m} \alpha_i = 1 \quad y \quad \alpha_i \geq 0, \ i=1,\ldots,m$$

Este concepto nos permite obtener una caracterización alternativa y común de conjunto convexo.

Proposición A.2. Sea S un conjunto contenido en el espacio R^n. Entonces S es convexo, sí y sólo si contiene todas las combinaciones convexas finitas de sus elementos.

Prueba. Sólo es necesario probar que si S es convexo, entonces contiene todas las combinaciones convexas finitas de sus elementos. Por definición, esto es cierto si m=2. Usando inducción, supongase que esto se cumple para cualquier m-1 elementos. En el caso de m elementos, sean $x_i \varepsilon S$, $\alpha_i \geq 0$,

$i=1,\ldots,m$ y $\sum_{i=1}^{m} \alpha_i = 1$ y para simplificar $1>\alpha_m$, entonces

$$X = \sum_{i=1}^{m-1} \alpha_i x_i + \alpha_m x_m = (1-\alpha\ )\ y + \alpha_m x_m \quad \text{donde}$$

$Y = \sum_{i=1}^{m-1} (\alpha_i/1-\alpha_m)x_i$. Sin embargo $y\varepsilon S$ por inducción .

De donde $x\varepsilon S$ y la prueba termina.

En el caso de que un conjunto dado no sea convexo es conveniente asociar a éste, un cierto conjunto convexo llamado el contorno convexo.

DEFINICION A.3. Sea S un conjunto contenido en el espacio R^n. El contorno convexo de S, denotado por C o S es el conjunto convexo obtenido de la intersección de todos los conjuntos convexos que contienen a S.

Proposición A.3 Sea S un conjunto en R^n. Entonces, C o S consiste de todas las combinaciones convexas finitas de elementos de S.

Prueba. Es obvio que $S \subset C$ o S. Sea P el conjunto de todas las combinaciones convexas finitas de elementos de S, esto es,

$$P = \{x: x = \sum_{i=1}^{m} \alpha_i x_i\ ,\ x_i \varepsilon S,\ \alpha_i \geq 0,\ i=1,\ldots,m,\ \sum_{i=1}^{m} \alpha_i = 1,\ m \text{ finito}\}$$

Se observa fácilmente que P es un conjunto convexo y que $S \subset P$. Por otra parte, de la proposición A.2 se obtiene que $P \subset C$oS y de la definición A.3 se implica que $P \supset C$oS, lo cual demuestra que P=CoS y la prueba termina.

Cabe mencionar que el contorno convexo de m puntos es un conjunto cerrado.

DEFINICION A.4 Un conjunto K contenido en el espacio R^n, es un cono si $\alpha x \in K$ cuando $x \in K$ y $\alpha \geq 0$.

La idea fundamental de un cono es que, si un punto pertenece al cono, entonces la línea que parte del origen y pasa por el punto también pertenece al cono. Nótese que un cono no es necesariamente un conjunto convexo, sin embargo, si un cono es un conjunto convexo, se denomina cono convexo. Como ejemplo de cono convexo se tiene el conjunto $K = \{x \in R^n : Ax \geq 0\}$, donde se verifica fácilmente que K es un cono convexo.

Las caras de K son conos convexos de menor dimensión, cuando la dimensión de estas caras es igual a uno, se denominan rayos extremos.

Sea K un cono convexo en R^n. Entonces

$$K^* = \{y \in R^n : xy \leq 0 \text{ para todo } x \in K\}$$

es un cono, denominado el cono dual de K.

El conjunto convexo más importante en la teoría de optimización es el hiperplano. Establecer el concepto de hiperplano es sencillo si se generaliza las propiedades geométricas de la línea y el plano en los espacios de dos y tres dimensiones, respectivamente.

DEFINICION A.5 Un conjunto V en el espacio R^n es una variedad lineal si dados x, $y \in V$ se tiene que $\alpha x + (1-\alpha)y \in V$ para toda $\alpha \in R$.

Se observa que esta definición está relacionada con la de conjunto convexo. Esto es, en la variedad lineal (también llamada conjunto afín) V dados dos puntos, no sólo la línea que los une está en V, sino toda la línea que pasa por estos puntos.

Es claro que dada una variedad lineal en R^n se puede hablar de dimensión. Por ejemplo, una línea recta es una variedad lineal de dimensión 1. En general, la dimensión de una variedad lineal en R^n se puede determinar por medio de la traslación de ésta al origen, lo que resulta en un subespacio lineal cuya dimensión es fácil de calcular. Así, una interpretación geométrica de variedad lineal es que ésta es una traslación de un subespacio.

DEFINICION A.6 Un conjunto H es un hiperplano en R^n si H es una variedad lineal de dimensión n-1.

El hiperplano de un espacio es, simplemente, una variedad lineal que separa al espacio en dos partes. Así, en el espacio de dos dimensiones, la línea recta separa los puntos del espacio en dos partes y lo mismo sucede con el plano en un espacio de tres dimensiones.

Proposición A.4 Sea a un vector hilera de n componentes, y sea b un escalar. Entonces el conjunto

$$H = \{x \in R^n \mid ax = b\}$$

es un hiperplano en R^n.

Prueba. Es fácil verificar que H es un conjunto afín o variedad lineal. Selecciónese x_0 y H y fórmese el conjunto $M = H-x_0$, esto es

$$M = \{x\varepsilon R^n : a(x-x_0)=0\} = \{y\varepsilon R^n ; ay=0\}$$

Recíprocamente, si $y\varepsilon R^n$ es tal que ay=0, esto es, $y\varepsilon M$ se tiene que $x_0 + y\varepsilon H$. En este caso, M está constituido por todos los vectores que son otorgonales al vector a y por lo tanto M tiene dimensión n-1 y lo mismo es cierto de H.

Lo interesante de la proposición es que el resultado recíproco es cierto, esto es

Proposición A.5 Sea H un hiperplano en R^n. Entonces, existe un vector a diferente de cero, perteneciente a R^n y un escalar b tal que

$$H = \{x\varepsilon R^n : ax=b\}$$

Prueba. Selecciónese un elemento $x_0\varepsilon H$ y fórmese el conjunto $M = H-x_0$. Puesto que H tiene dimensión n-1, el subespacio M tiene dimensión n-1. Sea a R^n un vector ortogonal a M. Entonces, $M = \{x\varepsilon R^n ; ax=0\}$. Hágase $ax_0 = b$ y obsérvese que para cualquier $x_1\varepsilon H$ tenemos $x_1-x_0\varepsilon M$. Equivalentemente $ax_1-ax_0=0$. Por lo tanto, $ax_1 = b$ y H está contenido en el conjunto $\{x\varepsilon R^n : ax=b\}$. Finalmente, H tiene dimensión n-1 y lo mismo es cierto de $\{x\varepsilon R^n ; ax=b\}$ (proposición A.4). Por lo tanto, se puede concluir que $H = \{x\varepsilon R^n ; ax=b\}$ y la prueba termina.

En resumen, la combinación de las proposiciones A.4 y A.5 establecen que un hiperplano es el conjunto de soluciones de una ecuación lineal.

DEFINICION A.7 Sea $H = \{x : ax=b\}$ un hiperplano en R^n. Entonces los conjuntos convexos dados como

$$H^+ = \{x : ax \geq b\} \text{ y } H^- = \{x : ax \leq b\}$$

se denominan los semiespacios cerrados positivo y negativo asociados con H, respectivamente. Los semiespacios abiertos positivo y negativo, respectivamente son:

$$\overline{H}^+ = \{x : ax>b\} \quad \text{y} \quad \overline{H}^- = \{x : ax<b\}$$

DEFINICION A.8 Un conjunto que consiste de la intersección de un número finito de semiespacios cerrados se denomina politope convexo. Asimismo, si un politope es no vacío y acotado se denomina poliedro.

Un caso especial de un poliedro convexo es

DEFINICION A.9 El contorno convexo de cualquier conjunto de n+1 puntos de R^n, los cuales no están en un mismo hiperplano de R^n se denomina simplex.

Se observa que si los n+1 puntos no están en un mismo hiperplano, n puntos deben ser linealmente independientes, puesto que en otro caso todos los puntos caerían en un mismo hiperplano que pasa por el origen.

A continuación se presenta el concepto de punto extremo y se establece su caracterización analítica para el caso de politopes convexos formados por las soluciones de un sistema de ecuaciones lineales. Esta caracterización permite analizar, teóricamente, el problema de programación lineal.

Geométricamente el concepto de punto extremo es obvio. Por ejemplo, en un espacio de dos dimensiones, un cuadrado y un triángulo tienen como puntos extremos sus vértices y en un círculo todos los puntos de la frontera son puntos extremos. Analíticamente el concepto de punto extremo se puede establecer por la siguiente

DEFINICION A.10 Un elemento x del conjunto convexo S es un punto extremo si para toda y,z de S y $0<\alpha<1$, se tiene que $x = \alpha y+(1-\alpha)Z$ implica x=y=Z.

Para establecer la equivalencia de puntos extremos y soluciones básicas, es conveniente recordar (Capítulo II, sección 2.1) que para el sistema de ecuaciones Ax=b, donde A es una matriz mxn, b es un vector columna de m componentes y x es un vector columna de n componentes, se tiene que si B es una submatriz de A de orden mxm no singular y suponiendo que los n-m componentes del vector x no asociados a las columnas de B se igualan a cero, entonces la solución del conjunto de ecuaciones resultante se denomina una solución básica con respecto a la base B y las componentes de x asociadas a las columnas de B se denominan variables básicas, siendo la idea de

la solución básica del sistema de ecuaciones Ax=b el que podemos escribir $A = [B,R]$, donde B es no singular, quedando entonces el sistema $BX_B = b$ donde el vector $X = [X_B, 0] = [B^{-1}b, 0]$ es la solución deseada.

TEOREMA A.1 Sea A una matriz mxn de rango m y b un vector columna de m componentes. Sea K el politope convexo que consiste de todos los vectores x que satisfacen el sistema $Ax=b, x\geq 0$. Entonces, un vector x es un punto extremo de K si y sólo si $x\geq 0$ es una solución básica de Ax=b.

Prueba. Supóngase que $X=(x_1, \ldots, x_m, 0, \ldots, 0)\geq 0$ es una solución básica de Ax=b. Entonces

$$a_1x_1 + a_2x_2 + \ldots + a_mx_m = b$$

donde $a_1, \ldots, a_m$ son los primeros vectores columna de A, los cuales son linealmente independientes. Supóngase que X no es punto extremo. Entonces, existen elementos distintos $b, Z \in K$ tales que $X=\alpha y+(1-\alpha)Z$ para algún $0<\alpha<1$ y $y\geq 0$, $z\geq 0$, se tiene que los últimos n-m componentes de estos vectores deben ser cero. Por otra parte, se tiene

$$a_1y_1 + a_2y_2 + \ldots + a_my_m = b$$

$$a_1z_1 + a_2z_2 + \ldots + a_mz_m = b$$

donde los vectores $a_1, \ldots, am$ son linealmente independientes, lo cual implica que x = y = z, lo cual es una contradicción, por lo tanto se concluye que x es punto extremo de K.

Recíprocamente, supóngase que X es un punto extremo de K. Por simplificar se supone que los primeros k componentes de X son diferentes de cero, luego se tiene

$$a_1x_1 + a_2x_2 + \dots + a_kx_k = b$$

donde $x_i > 0$, $i=1,\dots,k$, $x_j = 0$, $j = k+1,\dots,n$. Supóngase que $a_1,\dots,a_k$ son vectores linealmente dependientes, por lo tanto existirá una combinación lineal no trivial de estos vectores tal que

$$a_1y_1 + \dots + a_ky_k = 0$$

Defínase el vector $y= (y_1,y_2,\dots,y_k,0,\dots,0)$ en R^n. Usando el hecho que $x_i>0$, $i=1,\dots,k$ es posible encontrar un $\varepsilon>0$ tal que

$$x + \varepsilon y \geq 0 \quad y \quad x - \varepsilon y \geq 0$$

de donde $X = \frac{1}{2}(X + \varepsilon y)+ \frac{1}{2}(X\cdot\varepsilon y)$, esto es, X se puede expresar como una combinación convexa de dos elementos distintos en K, lo cual constituye una contradicción pues X es punto extremo y por lo tanto los vectores $a_1,a_2,\dots,a_k$ son linealmente independientes. Nótese que $k\leq m$, si $k = m$ es claro que $X > 0$ y es una solución básica de $Ax=b$. Si $k<m$ es posible encontrar $m - k$ vectores linealmente independientes del conjunto de $n-m$ vectores restantes y X sigue siendo una solución básica, en este caso, degenerada. Esto termina la prueba.

Es conveniente puntualizar que en el teorema anterior, el número de soluciones básicas del sistema lineal $Ax=b$ es finito

pues es menor que

$$\binom{n}{m} = \frac{n!}{m!\,(n-m)!}$$

correspondiente al número de formas distintas de seleccionar m de las n columnas de la matriz A. Consecuentemente, el número de puntos extremos de politopes convexos formados por las soluciones del sistema $Ax=b$, $x \geq 0$ es finito. También se enfatiza que el teorema A.1 puede ser aplicado en la caracterización de los puntos extremos de otros politopes convexos, como es el caso del politope convexo formado por las soluciones del sistema $Ax \leq b$, $x \geq 0$.

Proposición A.6 Considérese el sistema lineal

$$\begin{aligned} Ax &\leq b \qquad (P) \\ x &\geq 0 \end{aligned}$$

donde A es una matriz mxn, b un vector columna de m componentes y x un vector columna de n componentes. La forma estándar de (P) es

$$\begin{aligned} Ax + Iy &= b \\ x \geq 0 \;,\; y &\geq 0 \end{aligned} \qquad (P')$$

donde I es la matriz identidad de orden nxm y y es un vector columna de m componentes. Entonces, existe una correspondencia uno a uno entre los puntos extremos del politope (P) en R^n y los del politope (P') en R^{n+m}.

Prueba. Sean $(\bar{x}, \bar{y})$ un punto extremo de (P'). Supóngase que $\bar{x}$ no es punto extremo de (P). Entonces existen elementos dis-

tintos x_1, x_2 en R^n que satisfacen (P), tales que $x = \alpha x_1 + (1-\alpha)x_2$ para algún $0<\alpha<1$. Sin embargo,

$$Y_1 = b-Ax_1 \geq 0 \ , \ Y_2 = b-Ax_2 \geq 0$$

de donde se concluye que $y = b-A\bar{x} = \alpha y_1 + (1-\alpha)y_2$, por lo que $(\bar{x},\bar{y})$ no es punto extremo de (P') lo cual es una contradicción. Por lo tanto, $\bar{x}$ es punto extremo de (P). Además,si $\bar{x}$ es punto extremo de (P), entonces $(\bar{x},\bar{y})$ donde $\bar{Y} = b-A\bar{x}$ es punto extremo de (P').

TEOREMA A.2 Supóngase que la función lineal $Z=Cx$ tiene un mínimo en el politope convexo $K = \{x\in R^n : Ax=b, x \geq 0\}$. Entonces, el mínimo se adquiere en un punto extremo de K.

Prueba. Sean $x^1, \ldots, x^k$ puntos extremos de K. Entonces cada X en K puede expresarse como

$$X = \sum_{i=1}^{k} \alpha_i x^i \ ; \ \sum_{i=1}^{k} \alpha_i = 1 \ ; \ \alpha_i \geq 0 \ , \ i=1,\ldots,k$$

Sea $\bar{Z} = \min \{Cx^i \ , \ i=1,\ldots,k\}$. Entonces se tiene

$Cx = \sum_{i=1}^{k} \alpha_i \ cx^i \geq (\sum_{i=1}^{k} \alpha_i)\bar{Z} = \bar{Z}$ y la prueba termina.

APENDICE B

B.1 Introducción

El programa GUB es un método de programación lineal diseñado para resolver problemas lineales angulares con restricciones y variables de liga. Específicamente, problemas de la forma

$$\text{minimice } z = c_0x_0+c_1x_1+\ldots+c_px_p+dy$$

$$D_0x_0+D_1x_1+D_2x_2+\ldots+D_px_p+H_0y = b_0$$

$$E_1x_1 \qquad\qquad +H_1y = b_1$$

$$F_2x_2 \qquad\qquad +H_2y = b_2$$

$$\ldots$$

$$F_px_p+H_py = b_p$$

$$x_0 \geq 0 \; ; \; x_1 \geq 0 \; ; \; \ldots \; ; \; x_p \geq 0 \; ; \; y \geq 0$$

Este programa es un método de solución simplex revisado que utiliza la técnica de factorización de la base, cuyo propósito es manejar y actualizar las inversas de las bases.

B.2 Descripción del programa

El programa G-GUB fue escrito y desarrollado por Winkler(ref.34),usando las facilidades técnicas que ofrece el sistema de computadoras IBM disponible en la Universidad de Stanford. Las subrutinas de programación lineal de este programa han sido adaptadas del código Fortran LPM-1, desarrollado por Tomlin (ref.32). El programa consiste de un programa principal, un subprograma - - BLOCKDATA y 22 subrutinas, cuyos nombres se dan en la Tabla 1.

El programa G-GUB tiene un aspecto importante relacionado con la flexibilidad de uso de diferentes estrategias de selección del par de vectores que entra y sale de la base en las iteraciones del método simplex revisado. La idea y ventaja de disponer de varias estrategias, es seleccionar la más adecuada - para los problemas particulares que se resuelvan, pues, la operación de selección del par de vectores que entra y sale de la base es lo que más tiempo consume en el desarrollo de los programas de programación lineal. Las estrategias disponibles y - su descripción se dan en la Tabla 2. Las estrategias incluyen el caso de no emplear la forma factorizada de la base. Asimismo, se incluyen los casos de considerar costos relativos de - todos ó una sola parte de los vectores no básicos al seleccionar el vector que entra a la base. Las estrategias, también incluyen el caso de formación de una base global inicial a partir de las bases factibles (u óptimas) asociadas con cada uno de los subproblemas.

El programa D-GUP puede considerarse formado por tres etapas, las cuales corresponden a cada una de las llamadas de subrutinas que hace el programa principal. La primera etapa consistirá pues en la llamada de la subrutina INDATA y es donde se efectúa el método simplex revisado por cada subproblema - en forma individual. La segunda etapa coincide con la llamada que hace el programa principal de la subrutina NORMAL y en esta parte del programa se realiza la inversión total, esto es, se efectúa el método simplex revisado tomando los subproblemas en forma global. La tercera y última etapa del - programa está constituida por la subrutina UNRAVL y en ella se calculan los precios sombra y se escriben los resultados finales.

TABLA 1

Subprogramas que componen G-GUB

BLKDAT	NORMAL
BTRAN	PACK
CHANGE	PRICE
CHSOL	UNPACK
CHUZR	UNRAVL
FORMC	UPBETA
FTRAN	UPNVRS
INDATA	UPVEC
INPUT1	VECTOR
INVERT	WRETA
INVRSE	XBASIS
ITEROP	

TABLA 2

Descripción de estrategias

Estrategia	Representación de la base	Fase I	Costos relativos *
-1	Forma factorizada	Factibilización en cada bloque	Parcial
0	Forma factorizada	Optimalidad en cada bloque	Parcial
1	Forma factorizada	Optimalidad en cada bloque	Total
2	Forma factorizada	No factibilidad	Parcial
3	Forma factorizada	No factibilidad	Total
4	Forma factorizada	----------	-------
5	Normal (LPM1)	No factibilidad	Total

* Selección del vector no básico que entra a la base considerando todos los vectores posibles o una sola parte de ellos.

B.2.1 Descripción de las subrutinas

SUBRUTINA BTRAN

Propósito:

La subrutina BTRAN se utiliza en las tres etapas del programa G-GUB. Esta subrutina ejecuta la transformación hacia atrás, esto es, calcula el vector de precios sombra. En el caso de que el vector eta sea nulo, esto es, NETA=0, la subrutina BTRAN no se efectúa.

SUBRUTINA CHANGE (SLUP)

Propósito:

Verificar si los archivos actuales que contienen a A(.), IA(.), NB(1340) corresponden a los del actual subproblema. En caso contrario, actualizarlos proporcionándole a la subrutina VECTOR los parámetros necesarios. Es llamada por la suburtina NORMAL e INVRSE (en la segunda etapa del programa principal) y por la subrutina UNRAVL (en la tercera y última etapa del programa principal).

Parámetros:

LSUP es el número del subproblema en cuestión.

SUBRUTINA CHSOL(KINST)

Propósito:

La subrutina CHSOL (KINST) desempeña tres funciones ó fases diferentes, dependiendo del valor del parámetro KINST. La primera de ellas consiste en verificar la exactitud de la inversa de la base (por medio de los arreglos YA(IA(J),1) y YA(I,KPRICE)

en la primera etapa del programa G-GUB, esto es, cuando la - inversión de la base se hace en cada subproblema de manera - individual. La segunda función de CHSOL se realiza en la segunda etapa del programa G-GUB (inversión total) y obtiene la variable ERMAX (máximo error por fila) que sirve para verificar la exactitud de la inversa de la base. La tercera función que desempeña CHSOL también se lleva a cabo en la segunda etapa del programa G-GUB y en ella calcula ERMAX y COND (imprime estos resultados), variables que representan el máximo error por fila y el número aproximado de bases condicionales (elementos de la inversa de la base que provocan error). Esto lo hace antes de empezar con las iteraciones finales del método simplex revisado.

Parámetro:

KINST puede tomar los valores de 1 a 4 inclusive. El valor de 1 corresponde a la segunda fase de CHSOL, cuando KINST se - iguala a 2, CHSOL efectúa la tercera fase y para los valores 3 y 4 de KINST, CHSOL desarrolla la primera fase.

SUBRUTINA CHUZR (KVEC, NTYPE, KENTRY)

Propósito:

La subrutina CHUZR elige hilera pivote para el método simplex revisado.

Parámetros:

KVEC es la columna para la cual se hace el análisis y se asigna en la subrutina NORMAL. NTYPE se calcula en CHUZR misma tomando los valores 1, 2 (en la estrategia cero) y 3, dependiendo de los valores de las variables DRART, DRMIN y DRMAX, las

cuales son calculadas en CHUZR misma. KENTRY sólo toma los valores 2 ó 1 dependiendo si la columna indicada por KVEC es pivote ó no respectivamente y es asignada en la subrutina NORMAL.

SUBRUTINA FORMC

Propósito:

La subrutina FORMC, forma el vector de la función objetivo para el método simplex revisado y se usa en conjunto con la subrutina PRICE.

SUBRUTINA FTRAN (KPAR,KVEC)

Propósito:

La subrutina FTRAN realiza la transformación hacia adelante de las columnas de la matriz por la inversa de la base, necesaria para determinar el vector que sale de la base en cada iteración. Esta subrutina se utiliza en las dos primeras etapas del programa G-GUB.

Parámetros:

KPAR determina los vectores eta mediante los cuales se actualiza la columna KVEC.

SUBRUTINA INDATA

Propósito:

INDATA es la primera de las tres subrutinas que componen el programa principal (primera etapa del programa G-GUB), además es en la única etapa del programa donde se usa. Tiene por objeto recoger la información dada al programa G-GUB por medio

de las tarjetas de datos y gobierna por medio de la subrutina NORMAL la ejecución del método simplex revisado, para cada uno de los subproblemas del problema.

SUBRUTINA INPUT1

Propósito:

Leer todas las tarjetas de datos, en formato MPS, correspondientes al problema, esto es, lee todas las tarjetas de datos a excepción de la primera (leída en INDATA). Verifica si los nombres de las columnas, renglones y lados derechos están repetidos y en éste caso marca error. Calcula la densidad del problema imprimiendo sus estadísticas; almacena los nombres de las variables no restringidas, los coeficientes distintos de cero en el vector A, en el vector LA guarda el número de índice del vector A, donde empieza cada nueva columna, en el vector IA almacena el número del renglón del subproblema en cuestión, al cual está asociado el coeficiente y cuenta las columnas incluido el lado derecho del subproblema, imprimiendo la información correspondiente al nombre del problema, filas, columnas, lado derecho, columnas estructurales, número de elementos no nulos. La subrutina INPUT1 es llamada por la subrutina INDATA, tantas veces como número de subproblemas (incluído el subproblema cero) tenga el problema y sólo es llamada en la primera etapa del programa G-GUB.

SUBRUTINA INVERT

Propósito:

La subrutina INVERT determina la inversa de la base por medio del método de descomposición LU y es utilizada en las dos primeras etapas del programa G-GUB. En ella se imprimen las variables ERMAX (máximo error de hilera), NETA (número de componentes del vector eta), NELEM (número de columnas estructuradas), NBNONZ (número de elementos diferentes de cero).

SUBRUTINA INVRSE

Propósito:

La subrutina INVRSE sólo es utilizada al inicio de la segunda etapa del programa G-GUB, esto es, cuando se realiza la inversión total, la cual realiza ayudándose de la subrutina INVERT.

SUBRUTINA ITEROP

Propósito:

Escribe los títulos y resultados de las iteraciones del método simplex revisado en las dos primeras etapas del programa G-GUB, donde la primera etapa corresponde al proceso efectuado, tomando cada subproblema individualmente o sea a la primera llamada de subrutina del programa principal (SUBRUTINA INDATA) y la segunda etapa será aquélla en la cual se hace la inversión total y corresponde a la segunda llamada de subrutina del programa principal (SUBRUTINA NORMAL).

Parámetros:

a) KPAR.- toma únicamente los valores cero y uno, dependiendo si ITEROP va a escribir títulos o iteraciones respectivamente.

b) KVEC.- indica la columna que ha sido pivote para el método simplex revisado.

c) LSTAT

SUBRUTINA NORMAL

Propósito:

La subrutina normal dirige la ejecución del método simplex revisado. Se utiliza en la primera etapa del programa G-GUB y constituye la segunda etapa del mismo.

SUBRUTINA PACK(KVEC,IIFROM)

Propósito:

La subrutina PACK auxilia para determinar la salida de un vector de la base de trabajo y también en el cambio de los valores X y Y debido a la permutación entre filas y es llamada sólo por la subrutina UPNVRS durante la segunda etapa del programa G-GUB.

Parámetros:

KVEC es la columna pivote e IIFROM representa al subproblema al cual corresponde KVEC, esto es, IIFROM es igual a JFROM, calculado en la subrutina PRICE.

SUBRUTINA PRICE

Propósito:

Calcular los costos relativos y elegir la columna pivote para el método simplex revisado. Se utiliza en las tres etapas del programa G-GUB.

SUBRUTINA UNPACK(IV,KVEC)

Propósito:

La subrutina UNPACK expande las columnas de la matriz comprimida insertando ceros en los sitios indicados y se usa en las tres fases del programa G-GUB.

Parámetros:

IV es el argumento del arreglo LA(.), el cual indica el índice del arreglo A(.), que es la matriz comprimida, a partir del cual están guardados los coeficientes distintos de cero de la variable correspondiente a la columna KVEC.

SUBRUTINA UNRAVL

Propósito:

UNRAVL es la tercera y última subrutina que compone el programa principal (tercera etapa del programa G-GUB) y es en la única etapa del programa donde se utiliza. UNRAVL es la subrutina que escribe el encabezado correspondiente y la información final que proporciona el programa G-GUB donde aparecen los valores finales de las variables básicas y de las duales ó precios sombra, así como las restricciones a las cuales están asociadas y sus respectivos lados derechos.

SUBRUTINA UPBETA (KVEC)

Propósito:

La subrutina UPBETA actualiza el lado derecho y se utiliza en las dos primeras etapas del programa G-GUB.

Parámetros:

KVEC es la columna pivote.

SUBRUTINA UPNVRS(KVEC,KOUT)

Propósito:

En la subrutina UPNVRS se efectúan:

a) Salida del vector en la base de trabajo, en la segunda etapa del programa G-GUB, y se ayuda de las subrutinas PACK(KVEC,JFROM(KVEC)) y WRETA(KWVEC).

b) Salida del vector en un subproblema, sólo en la segunda etapa del programa.

c) Cambio del vector que sale con un vector en la base de trabajo y escritura de la fila eta, sólo en la segunda etapa del programa y se ayuda de la subrutina WRETA(LSUB).

d) Cambio de los valores de los vectores X y Y debidas a la permutación de filas, también sólo se realiza en la segunda etapa del programa e intervienen las subrutinas PACK (KVEC,JFROM(KVEC)) y WRETA(KWVEC).

e) Actualización de las columnas en la base de trabajo, también sólo en la segunda etapa del programa, y se ayuda de las subrutinas UNPACK(JH(.), KPRICE) y PACK (KPRICE, JTEMP(4)).

f) Actualización de los vectores que entran, vaciado del

vector cambiado y acumulación de información para el vaciado de la base del subproblema, también sólo en la segunda etapa del programa.

g) Verifica si hay otro vector que introducir a la base, sólo en la segunda etapa del programa y se auxilia de las subrutinas FORMC y BTRAN.

h) Actualización de la base y verifica si hay otro vector que mejore la presente base. Es la única parte de la subrutina UPNVRS que se utiliza en las primera y segunda etapa del programa G-GUB, y se auxilia de las subrutinas WRETA(KVEC), FORMC y BTRAN(3,K).

Parámetros:

KVEC es el vector columna que va a intervenir en la subrutina y KOUT se calcula en UPNVRS misma y es usado en la subrutina NORMAL.

SUBRUTINA UPVEC(KVEC,KINST)

Propósito:

Realizar todos los cambios necesarios una vez hecha la transformación hacia adelante, de las columnas de la matriz por la inversa de la base. Consta de dos fases, de acuerdo al valor del parámetro KINST.

Parámetros:

KVEC es la columna pivote y KINST sólo puede tomar el valor 1 ó 2 y es asignado en la subrutina NORMAL.

SUBRUTINA VECTOR (KS,AX, IAX, NBX, KI)

Propósito:

Leer y escribir en los archivos que contienen los arreglos A(.), IA(.), NB(1340) correspondientes a cada uno de los sub problemas del problema.

Parámetros:

a) KS toma únicamente los valores 1 ó 2. (Escribir ó leer).

b) AX es el arreglo A(.)

c) IAX representa el arreglo IA(.)

d) NBX corresponde al arreglo NB(1340)

e) KL es el número de palabras en los registros A(.), IA(.).

SUBRUTINA WRETA(KVEC)

Propósito:

La subrutina WRETA forma nuevos vectores eta para el cálculo de la forma producto de la base inversa y se utiliza en las dos primeras etapas del programa G-GUB.

Parámetros:

KVEC, es la columna a la cual corresponde el nuevo vector eta.

SUBRUTINA XBASIS(KRU,KOUT)

Propósito:

La subrutina XBASIS se utiliza en las segunda y tercera etapa del programa G-GUB, esto es, se utiliza en las iteraciones del simplex revisado, tomando los subproblemas en forma global y en el cálculo final de los precios sombra. Para ISUB=0 ó ITSINV=0, la subrutina XBASIS no se efectúa, y ésta llama a las subrutinas UNPACK, FTRAN(1,KWVEC) y WRETA(KWVEC).

Parámetros:

El parámetro KRU, en el análisis de la estrategia cero, sólo toma el valor de dos y sirve para evaluar la variable LRU que hace las veces de centinela o bandera y el parámetro KOUT es valuado en XBASIS misma y que junto con la subrutina UPNVRS son las únicas que modifican su valor, el cual es utilizado en la subrutina NORMAL.

B.3 Entrada de datos

En el programa G-GUB la entrada de datos es semejante a los sistemas comerciales APEX y MPS. En particular, la estrategia 5 usa el mismo formato de lectura, el cual se caracteriza, primeramente, por la lectura de los nombres de cada una de las restricciones del problema lineal. A continuación se especifican los nombres de las variables incluyendo únicamente aquéllas restricciones en donde esta variable tiene - coeficiente distinto de cero. Finalmente, se leen los datos derechos si estos son diferentes de cero.

En el caso de las estrategias -1 a 4 del programa G-GUB, la lectura de datos se efectúa por medio de bloques, con el objeto de aprovechar la estructura del problema lineal. Cada bloque contiene una sección de renglones, otra de columnas y otra de lados derechos. Este bloque de hecho caracteriza a un subproblema independiente, ligado a los demás únicamente por las variables y restricciones de liga.

Entrada de Datos

1. La primera tarjeta proporciona información general acerca del problema en cuestión. Se leen 10 datos con formato 10I4, cuya descripción es:

LMAX	Número de subproblemas en el problema.
IOBJ	Número de hilera usado para la función objetivo.
JUSRHS	Número del vector usado como vector de recursos (RHS).
KINP	Unidad de lectura a utilizar.
INVFRQ	Frecuencia de verificación de precisión de la inversa de la matriz base del problema global.
INVFRW	Frecuencia de verificación de precisión de las inversas de la base de los subproblemas individuales.
LPNT	Indice indicando la posición del primer elemento distinto de cero de los vectores eta en el arreglo A (LPNT = 2500).
NSTR	Número de estrategia.
NULT	Número de vectores usados para determinar el que entra a la base en función de los costos relativos ($0 \leq NULT \leq KMULT$).
NPP	Factorización anidada (NPP = 0).

A continuación se tienen las tarjetas que proporcionan la información correspondiente a renglones y columnas del problema de programación lineal que se desea resolver. La estructura de la entrada de datos se modifica dependiendo de la estrategia que se utilice. En el caso de las estrategias -1,0,1,2,3, 4, tenemos los siguientes tipos de tarjetas:

2. En las primeras 4 columnas se escribe la palabra NAME y de la 15 a la 24 el nombre del problema maestro (subproblema en cuestión).

3. En las primeras 4 columnas se escribe la palabra ROWS para indicar que se van a leer las características de los renglones correspondiente a las restricciones de liga del problema maestro. (restricciones del subproblema en cuestión).

4. En la columna 2 ó 3 se escribe el tipo de restricción que se tiene. Mediante una N se indica que se trata de una restricción libre como es el caso de la función objetivo Una L indica que la restricción es de menor o igual; una G indica que la restricción es de mayor o igual, y una E indica que se trata de una igualdad estricta. En las columnas 5 a 14 se escribe el nombre de la restricción en cuestión. Nótese que se van a tener tantas tarjetas como restricciones de liga. (restricciones del subproblema en cuestión).

5. En las primeras 7 columnas se escribe la palabra COLUMNS para indicar que se va a leer la sección de columnas asociadas con el problema maestro. (asociadas con el subproblema en cuestión).

6. En las columnas 5 a 14 se escribe el nombre de cada una de las columnas cuyos datos se van a leer. En las columnas 15 a 24 se escribe el nombre de la restricción para la cual - la variable en cuestión tiene coeficiente distinto de cero. En las columnas 25 a 36 se escribe el valor numérico de este coeficiente con formato F12.4. En las columnas 40 a 49 se escribe el nombre de otra restricción para la cual la - variable en cuestión tenga coeficiente distinto de cero, y en las columnas 50 a 61 el valor numérico de este coeficiente con formato F12.4. En el caso del problema maestro estas variables están asociadas unicamente con restricciones de - liga. (En el caso de los subproblemas estas variables están asociadas tanto a las restricciones de liga como a las restricciones propias del subproblema).

7. En las primeras tres columnas se escribe la palabra RHS para indicar que se va a leer el vector de recursos disponibles correspondientes a las restricciones de liga del problema maestro(restricciones del subproblema en cuestión).

8. En las columnas 5 a 14 se escribe el nombre asociado con el vector de lados derechos del problema. En las columnas 15 a 24 se escribe el nombre de una restricción para la cual el lado derecho sea distinto de cero y en las columnas 25 a 36 el valor numérico de este coeficiente. En las columnas 40 a 49 se escribe el nombre de otra restricción y en las columnas 50 a 61 el correspondiente valor del lado derecho. El formato utilizado para valores numéricos es F12.4.

9. Siguiente tarjeta de datos: En las columnas 1 a 6 se escribe la palabra ENDATA para indicar que ya se terminó la entrada de datos del primer bloque del problema (subproblema en cuestión).

A continuación se repiten los últimos 8 conjuntos de tarjetas descritas anteriormente tantas veces como subproblemas haya en el problema global. Es conveniente hacer notar que en cada subproblema se deben incluir no sólo los coeficientes contenidos en el subproblema mismo, sino además aquéllos correspondientes a las restricciones de liga asociadas con las variables del subproblema en cuestión.

En el caso de la estrategia 5 se utiliza una estructura de entrada de datos como si fuese un solo subproblema en el caso de las estrategias -1 a 4. Específicamente, en la sección de ROWS se especifican todos los renglones de la matriz con su correspondiente tipo de restricción (N,L,G ó E). Asimismo, en la sección de COLUMNS, se especifican todos los coeficientes del problema en cuestión. Finalmente, en la sección de RHS se especifican todos los dados derechos. En esta estrategia no se utiliza la técnica de factorización de la base.

Este tipo de formato se conoce como formato APEX o MPS. En la práctica es común encontrar problemas lineales con estructura semejante a la manejada por G-GUB, cuyos datos se encuentran en formato APEX.

A continuación se muestra la codificación del ejemplo de la sección 4.9.

COMISION DEL PLAN NACIONAL HIDRAULICO

SARH

PROGRAMA *TESIS*

DIRECCION DEL CENTRO DE INFORMACION

CODIFICACION FORTRAN

FECHA ______

HOJA Nº *1*

PROGRAMADOR ______

C. PARA COMENTARIO

INSTRUCCION NUMERO | Ø = CERO, Ø = ALFABETICO O | 1 = UNO, I = ALFABETICO I | INSTRUCCION | 2 = DOS, Z = ALFABETICO Z | U = VOCAL U, V = CONSONANTE V | IDENTIFICACION

```
   2   1   1  5   70  30 1800  0  13  01
NAME          TESIS
RØWS
 N   ØBJ
 E   R1
 E   R2
CØLUMNS
    X1        ØBJ       -1.         R1        4.
    X2        ØBJ       -13.        R1        4.
    X2        R2        2.
RHS
    LIMIT     R1        22.         R2        10.
ENDATA
NAME          SUB1
RØWS
 E   R3
CØLUMNS
    X3        ØBJ       -1.         R1        1.
    X3        R3        1.
    X4        ØBJ       -1.         R2        1.
    X4        R3        1.
    X5        ØBJ       -13.        R1        2.
    X5        R2        4.          R3        1.
RHS
```

94

COMISION DEL
PLAN
NACIONAL
HIDRAULICO

SARH

PROGRAMA TESIS

DIRECCION DEL CENTRO DE INFORMACION

FECHA

HOJA Nº 2

C PARA COMENTARIO

CODIFICACION FORTRAN

PROGRAMADOR

INSTRUCCION

```
      LIMIT     R13       81.
ENDATA
NAME          SUB2
ROWS
 E    R14
 E    R15
COLUMNS
      X6        ØBJ       -12.      R11       4.
      X6        R12       3.        R14       1.
      X7        ØBJ       -11.      R11       2.
      X7        R12       1.        R14       1.
      X7        R15       11.
      X8        ØBJ       -12.      R11       1.
      X8        R14       1.
      X9        ØBJ       -13.      R11       2.
      X9        R15       11.
RHS
      LIMIT     R14       61.       R15       4.
ENDATA
```

B.4 Modificaciones y pruebas

El programa G-GUB fue escrito originalmente en lenguaje FORTRAN para máquinas IBM. Sin embargo, debido a las diferencias propias del manejo de archivos de los sistemas IBM y - CDC, éste último disponible en SARH, fue necesario hacer diversas modificaciones para el desarrollo y uso de este programa. En particular, se modificó el uso de archivos de acceso directo, así como la utilización de equivalencias debido a la formación de palabras en sistemas CDC.

Las modificaciones técnicas descritas anteriormente fueron - probadas por medio de tres problemas pequeños, los cuales - fueron resueltos usando cada una de las estrategias de G-GUB. Estos problemas también fueron resueltos utilizando APEX III para comparar tiempos de ejecución y comprobar resultados - (ver Tabla 3).

Es conveniente mencionar que originalmente la capacidad del problema G-GUB estaba demasiado restringida para los usos - que se pretende dar al programa en CPNH. Consecuentemente, - con el propósito de proceder a aumentar la capacidad del programa para resolver problemas mayores, se consideró necesario llevar a cabo ciertas modificaciones. Estas se describen a continuación. Cabe puntualizar que las modificaciones están ligadas por los problemas que ellas mismas fueron generando.

B.4.1 Eliminación de Doble Precisión

Una vez que se corrió el programa original se procedió a eliminar la doble precisión. La razón de esto son las limitaciones de memoria y las características de formación de palabras en el sistema CDC. También se modificó la entrada de datos del programa, en particular, la lectura de nombres de variables y restricciones, ya que en la versión original se utilizaban dos palabras para almacenar un sólo nombre, y esto no es necesario en sistemas CDC.

Con esta nueva versión de G-GUB, versión 2, se corrieron los problemas de prueba que se tenían, para verificar resultados y comparar tiempos de ejecución (ver tabla 3). Además, se hicieron pruebas con otro problema que también consistía de dos subproblemas, como los anteriores, pero cuya dimensión total es de 57 restricciones y 30 variables. Este problema es una parte del denominado problema NOROESTE.

El primer problema que surgió de esta prueba fue la limitación del número de registros de acceso directo. Consecuentemente, el número original de registros (80) se modificó a 200. Los nuevos resultados obtenidos mediante G-GUB fueron comparados con APEX, observando que únicamente las estrategias 0 y 5 estaban proporcionando resultados correctos (ver tabla 4).

B.4.2 Modificación de Dimensiones para considerar mayor número de costos relativos de los vectores no básicos.

En la versión original de G-GUB se tenía prevista la consideración de un máximo de tres costos relativos. Dado que únicamente se habían corrido problemas pequeños (del orden de 10 restricciones por 10 variables), no había sido necesario incrementar la cota máxima, pero como ahora se deseaba implementar el programa con problemas "medianos", fue necesario llevar a cabo - ciertas modificaciones que se resumen como:

Valor original	Valor modificado
KMULT = 3	KMULT = 10
KPRICE = 4	KPRICE = 11
KLAST = 8	KLAST = 21

Asimismo, se modificaron las dimensiones de los arreglos DCMIN, JCOLP y JFROM, de 3 a 10 en todos los casos. La dimensión de la matriz YA se modificó de 115 x 9 a 115 x 16, usando los cálculos indicados en los comentarios del programa principal de G-GUB. Además, fue necesario cambiar las proposiciones EQUIVALENCE en las subrutinas UPVEC, CHSOL, UPBETA, UPNVRS, NORMAL. Aquí conviene mencionar que la proposición original era: EQUIVALENCE (Y(1),YA(461)), (YTEM(1),YA(921)), y la modificada es EQUIVALENCE (Y(1),YA(1266), (YTEMP(1), YA(2416)).

La versión anterior de G-GUB, versión 3, se probó con los 3 problemas de prueba originales y además con la misma parte del problema NOROESTE que se consideró en la versión 2. Los resultados obtenidos fueron comparados con APEX, y las estrategias 0, 1, 3, 4 y 5 proporcionaron resultados correctos, más no así las restantes (-1 y 2). (ver tabla 3 y 4).

Con la versión 3 de G-GUB se decidió hacer algunas pruebas con el problema NOROESTE completo que consiste de 11 subproblemas y un total de 191 restricciones y 153 variables. Sin embargo, no se obtuvieron resultados correctos. En este momento se observó la posibilidad de que el programa G-GUB tuviera limitaciones en cuanto a los valores de los coeficientes de las variables, y arbitrariamente se redujeron los lados derechos del problema NOROESTE con dos subproblemas a números mucho menores que los originales. Con los datos anteriores, se corrió la versión 3 de G-GUB, se comparó con APEX III, y se observó que todos los resultados obtenidos fueron correctos (ver tablas 4 y 5).

B.4.3 Modificación de coeficientes máximos permitidos en G-GUB

Con base en los resultados obtenidos en el inciso anterior se empezaron a buscar las posibles restricciones para los valores de los coeficientes y se detectaron algunos límites superiores permitidos en la subrutina CHUZR. Estos valores eran de 10^9 y 10^{10} y fueron modificados a 10^{13} y 10^{11} respectivamente. De esta manera se generó la versión 4 de G-GUB con la que se corrieron los problemas NOROESTE con dos subproblemas y NOROESTE completo. En ambos casos se obtuvieron resultados idénticos a los de APEX-III para las estrategias -1, 0, 1, 2, 3 y 4. Sin embargo, la estrategia 5 no corrió con el problema NOROESTE completo, lo cual se debe a que esta estrategia funciona íntegramente en memoria, y las dimensiones de las variables en G-GUB no fueron suficientes para el tamaño del problema.

B.4.4. Modificación de dimensiones para considerar el número mayor de restricciones en el problema original.

En la versión original de G-GUB se tienen especificadas dimensiones para resolver problemas hasta con un máximo de 345 restricciones totales, por lo que fue necesario efectuar algunos cambios en variables y en dimensiones de arreglos para incrementar el número de restricciones totales permitidas. De acuerdo a las indicaciones establecidas por los comentarios al principio del programa, las modificaciones realizadas fueron las siguientes:

a. En el subprograma BLOCKDATA, el valor de la variable NROWT se modificó de 345 a 1035 y el de la variable KLAST de 15 a 21.

b. La dimensión de las variables X, Y, LROW y LNROW se modificó de 345 a 1035.

c. La dimensión de la matriz YA se modificó de (115,9) a (115,22).

d. Las proposiciones de EQUIVALENCE en las subrutinas UPVEC, CHSOL, UPBETA, UPNVRS y NORMAL se modificaron de: (Y(1), YA(1266)), (YTEMP(1), YA(1726)) a (Y(1), YA(1266)), (YTEMP(1), YA(2416))

Las modificaciones anteriores permiten resolver problemas hasta con un total de 1035 restricciones, pero incluyendo restricciones y variables de liga adicionales. Los resultados obtenidos se muestran en la tabla 6.

RESULTADOS PROGRAMA G-GUB

TABLA 3

PROBLEMA	NUMERO DE SUBPROBLEMAS	ESTRATEGIA	KULT	INVFAC	LPNT	VERSION 1 (doble precisión) TIEMPO	VERSION 1 NUM ITER	VERSION 1 FUNCION OBJETIVO	VERSION 2 (precisión sencilla) TIEMPO	VERSION 2 NUM ITER	VERSION 2 FUNCION OBJETIVO	VERSION 3 (10 vectores de costos relativos) TIEMPO	VERSION 3 NUM ITER	VERSION 3 FUNCION OBJETIVO	VERSION 4 (modificación de un subproblema) TIEMPO	VERSION 4 NUM ITER	VERSION 4 FUNCION OBJETIVO
LEON	2	-1	3	30	800	0.73	3	36.66667	0.67	3	36.66667						
		0	3			0.71	1		0.68	1							
		1	3			0.67	1		0.66	1							
		2	3			0.84	6		0.77	6							
		3	3			0.86	5		0.75	5							
		4	3			0.84	5		0.76	5							
		5	3	7	▽	0.28	5										
		-1	3	100	2500	0.86	3		0.67	3							
		0	3			0.79	1		0.66	1							
		1	3			0.82	1		0.66	1							
		2	3			0.97	6		0.78	6							
		3	3			0.98	5		0.78	5							
		4	3			0.95	5		0.76	5							
▽	▽	5	3	▽	▽			▽	0.25		▽						

RESULTADOS APEX-III:

Valor de la Función Objetivo : 36.66667 TIEMPO: 1.013

NOTA: Todos los tiemnos de proceso están dados en segundos.

RESULTADOS PROGRAMA G-GUB

TABLA 3
(Continuación)

PROBLEMA	NUEVOS RENGLONES	ESTRATEGIA	NULT	INVFRN	LPNT	VERSION 1 (doble precisión) TIEMPO	NUM ITER	FUNCION OBJETIVO	VERSION 2 (precisión sencilla) TIEMPO	NUM ITER	FUNCION OBJETIVO	VERSION 3 (10 vectores de costos relativos) TIEMPO	NUM ITER	FUNCION OBJETIVO	VERSION 4 (modificación de lim. superiores) TIEMPO	NUM ITER	FUNCION OBJETIVO
ALEX	4	-1	3	30	80	0.80	4	40.00000	0.70	4	40.00000						
		0				0.83	4		0.70	4							
		1				0.80	4		0.73	4							
		2				0.58	1		0.55	1							
		3				0.62	1		0.55	1							
		4				0.65	1		0.52	1							
		5				0.22	1										
		-1		100	250	0.89	4		0.78	4							
		0				0.98	4		0.83	4							
		1				0.96	4		0.84	4							
		2				0.68	1		0.63	1							
		3				0.68	1		0.64	1							
		4				0.69	1		0.63	1							
		5							0.19								

RESULTADOS APEX-III

Valor de la Función Objetivo = 40.00000 TIEMPO: 0.907

RESULTADOS PROGRAMA G-GUB

TABLA 3
(Continuación)

PROBLEMA	IMPRIMIR ITERACIONES	ESTRATEGIA	NULT	INVERW	LPNT	VERSION 1 (doble precisión) TIEMPO	VERSION 1 NUM ITER	VERSION 1 FUNCION OBJETIVO	VERSION 2 (precisión sencilla) TIEMPO	VERSION 2 NUM ITER	VERSION 2 FUNCION OBJETIVO	VERSION 3 (10 vectores de costos relativos) TIEMPO	VERSION 3 NUM ITER	VERSION 3 FUNCION OBJETIVO	VERSION 4 (modificación de lim. superiores) TIEMPO	VERSION 4 NUM ITER	VERSION 4 FUNCION OBJETIVO
TESIS	2	-1	3	30	800	0.99	5	37.00000	0.90	5	37.00000						
		0				0.92	4		0.88	4							
		1				0.98	5		1.03	5							
		2				1.00	8		1.08	8							
		3				1.21	8		1.20	8							
		4				1.24	8		1.05	8							
		5		▽	▽	0.39	8		0.34	8							
		-1		100	2500	1.10	5		0.91	5							
		0				1.05	4		0.91	4							
		1				1.12	5		0.90	5							
		2				1.31	8		1.07	8							
		3				1.41	8		1.15	8							
		4				1.47	8		1.24	8							
▽	▽	5	▽	▽	▽			▽			▽						

RESULTADOS APEX-III

Valor de la Función Objetivo : 37.00000 TIEMPO 1.0337

RESULTADOS PROGRAMA G-GUB

TABLA 4

PROBLEMA	NUMERO DE SUB PROBLEMAS	ESTRATEGIA	MULT	INVFRW	LPNT	VERSION 1 (doble precisión)			VERSION 2 (precisión sencilla)			VERSION 3 (10 vectores de costos relativos)			VERSION 4 (modificación de lim. superiores)		
						TIEMPO	NUM ITER	FUNCION OBJETIVO	TIEMPO	NUM ITER	FUNCION OBJETIVO	TIEMPO	NUM ITER	FUNCION OBJETIVO	TIEMPO	NUM ITER	FUNCION OBJETIVO
[illegible]	3	-1	3	30	800	9.10	42	7 581 594.4	6.09	42	7 581 594.4						
		0				6.14	26	10 570 913.9	4.50	26	10 570 913.9						
		1				6.03	21	8 444 542.3	4.41	21	8 444 542.3						
		2				2.77	5	5 307 911.9	2.20	5	5 307 911.9						
		3				9.49	40	8 554 573.3	6.41	40	8 554 573.3						
		4				9.49	40	8 554 573.3	6.39	40	8 554 573.3						
		5		↓	↓	5.60	38	10 570 913.9	3.31	38	10 570 913.9						
		-1		100	2500	10.13	42	7 581 594.4	6.48	42	7 581 594.4	3.89	18	10 570 913.9			
		0				6.68	26	10 570 913.9	4.70	26	10 570 913.9	5.75	32	10 570 913.9			
		1				6.48	21	8 444 542.3	4.59	21	8 444 542.3	5.10	27	10 570 913.9			
		2				2.92	5	5 307 911.9	2.29	5	5 307 911.9	2.30	5	5 307 911.5			
		3				10.08	39	8 554 573.3	6.57	39	8 554 573.3	6.71	39	8 554 573.3			
		4				10.01	39	8 554 573.3	6.59	39	8 554 573.3	6.73	39	8 554 573.3			
↓	↓	5	↓	↓	↓	5.11	38	10 570 913.9				3.09	27	10 570 913.9			

RESULTADOS APEX-III

Valor de la Función Objetivo 10 570 913.928 TIEMPO. 5.572

NOTA: La diferencia en los valores de la función objetivo se debe a que no todas las estrategias funcionaron correctamente para la versión de G-GUB considerada.

RESULTADOS PROGRAMA G-GUB

TABLA 4
(Continuación)

PROBLEMA	NUMERO SUB-PROBLEMAS	ESTRATEGIA	MULT	INVERW	LPNT	VERSION 1 (doble precisión)			VERSION 2 (precisión sencilla)			VERSION 3 (10 vectores de costos relativos)			VERSION 4 (modificación de lim superiores)		
						TIEMPO	NUM ITER	FUNCION OBJETIVO	TIEMPO	NUM ITER	FUNCION OBJETIVO	TIEMPO	NUM ITER	FUNCION OBJETIVO	TIEMPO	NUM ITER	FUNCION OBJETIVO
NOROESTE	2	-1	5	15	1500							5.05		10 570 913.9	5.09	31	10 570 913.9
		0										3.98		10 570 913.9	4.12	24	10 570 913.9
		1										4.48		10 570 913.9	4.52	23	10 570 913.9
		2										6.03		10 570 913.9	6.42	44	10 570 913.9
		3										5.24		10 570 913.9	5.75	34	10 570 913.9
		4										5.23		10 570 913.9	5.79	34	10 570 913.9
		5													3.01	27	10 570 913.9
		-1	10	15	1500							4.39	22	6 877 535.4	5.13	31	10 570 913.9
		0										4.33	24	10 570 913.9	4.23	24	10 570 913.9
		1										4.64	23	10 570 913.9	4.63	23	10 570 913.9
		2										5.01	31	7 140 520.0	6.45	44	10 570 913.9
		3										6.02	34	10 570 913.9	5.96	34	10 570 913.9
		4										5.98	34	10 570 913.9	5.99	34	10 570 913.9
		5										3.05		10 570 913.9	3.02	27	10 570 913.9

RESULTADOS PROGRAMA G-GUB

TABLA 4
(Continuación)

PROBLEMA	NUM. DE SUB-PROBLEMAS	ESTRATEGIA	KULT	INVFRW	LPNT	VERSION 1 (doble precisión)			VERSION 2 (precisión sencilla)			VERSION 3 (10 vectores de costos relativos)			VERSION 4 (modificación de lim. superiores)		
						TIEMPO	NUM ITER	FUNCION OBJETIVO	TIEMPO	NUM ITER	FUNCION OBJETIVO	TIEMPO	NUM ITER	FUNCION OBJETIVO	TIEMPO	NUM ITER	FUNCION OBJETIVO
NOROESTE	2	-1	10	30	800							3.97	19	6 877 535.4			
		0										4.11	24	10 570 913.9			
		1										4.75	25	10 570 913.9			
		2										5.03	31	7 140 520.0			
		3										5.96	34	10 570 913.9			
		4										6.01	34	10 570 913.9			
		5	↓	↓	↓							2.97	27	10 570 913.9	2.96	27	10 570 913.9
		-1	10	100	2500							4.43	19	6 877 535.4			
		0										4.37	24	10 570 913.9			
		1										4.95	25	10 570 913.9			
		2										5.20	31	7 140 520.0			
		3										6.04	34	10 570 913.9			
		4										6.10	34	10 570 913.9			
↓	↓	5	↓	↓	↓							2.98	27	10 570 913.9	3.00	27	10 570 913.9

RESULTADOS PROGRAMA G-GUB

TABLA 5

PROBLEMA	NUMERO DE PROBLEMAS	ESTRATEGIA	NULT	INVFRW	LPNT	VERSION 1 (doble precisión)			VERSION 2 (precisión sencillo)			VERSION 3 (10 vectores de costos relativos)			VERSION 4 (modificación de lim. superiores)		
						TIEMPO	NUM ITER	FUNCION OBJETIVO	TIEMPO	NUM ITER	FUNCION OBJETIVO	TIEMPO	NUM ITER	FUNCION OBJETIVO	TIEMPO	NUM ITER	FUNCION OBJETIVO
NGROSS1	11	-1	10	100	2500	67.30	130	8 728 214.3				30.81	115	15 899 302.4	46.57	183	19 020 383.9
		0				57.15	151	8 321 686.4				79.59	297	16 084 956.6	80.51	297	16 084 956.6
		1				55.63	105	14 467 476.9				28.92	76	12 346 859.5	38.61	107	19 020 383.9
		2				89.02	126	9 083 885.3				55.71	233	19 020 383.9	54.14	233	19 020 383.9
		3				89.06		11 446 499.9				49.08	128	19 020 383.9	46.30	128	19 020 383.9
		4				89.57	78	11 446 499.9				49.07	128	19 020 383.9	48.98	128	19 020 383.9
		5	↓	↓	↓												
		-1	10	15	1500							52.43	221	19 020 383.9	53.46	221	19 020 383.9
		0										40.22	167	19 020 383.9	40.52	167	19 020 383.9
		1										43.55	126	19 020 383.9	44.09	126	19 020 383.9
		2													54.14	233	19 020 383.9
		3													46.30	128	19 020 383.9
		4													48.98	128	19 020 383.9
↓	↓	5	↓	↓	↓												

RESULTADOS APEX-III

Valor de la Función Objetivo : 19 020 383.995 TIEMPO: 27.943

RESULTADOS PROGRAMA G-GUB

TABLA 5
(Continuación)

PROBLEMA	NUMERO SUB PROBLEMAS	ESTRATEGIA	NULT	INVFRW	LMT	VERSION 1 (doble precisión)			VERSION 2 (precisión sencilla)			VERSION 3 (10 vectores de costos relativos)			VERSION 4 (multiplicacion de lim. superiores)		
						TIEMPO	NUM ITER	FUNCION OBJETIVO	TIEMPO	NUM ITER	FUNCION OBJETIVO	TIEMPO	NUM ITER	FUNCION OBJETIVO	TIEMPO	NUM ITER	FUNCION OBJETIVO
NOROESTE	11	-1	10	100	1500	52.96	96	8 587 021.0				81.01	115	15 899 302.4			
		0										79.40	297	16 084 956.6			
		1										29.34	76	12 346 859.5			
		2										55.79	233	19 020 383.9			
		3										49.81	128	19 020 383.9			
		4										49.85	128	19 020 383.9			
		5															
		-1	10	30	1500										41.74	176	19 020 383.9
		0													44.86	176	19 020 383.9
		1													37.23	103	19 020 383.9
		2													53.02	237	19 020 383.9
		3													48.32	134	19 020 383.9
		4				46.15	116	16 574 717.3							49.40	134	19 020 383.9
		5															

RESULTADOS PROGRAMA G-GUB

TABLA 6

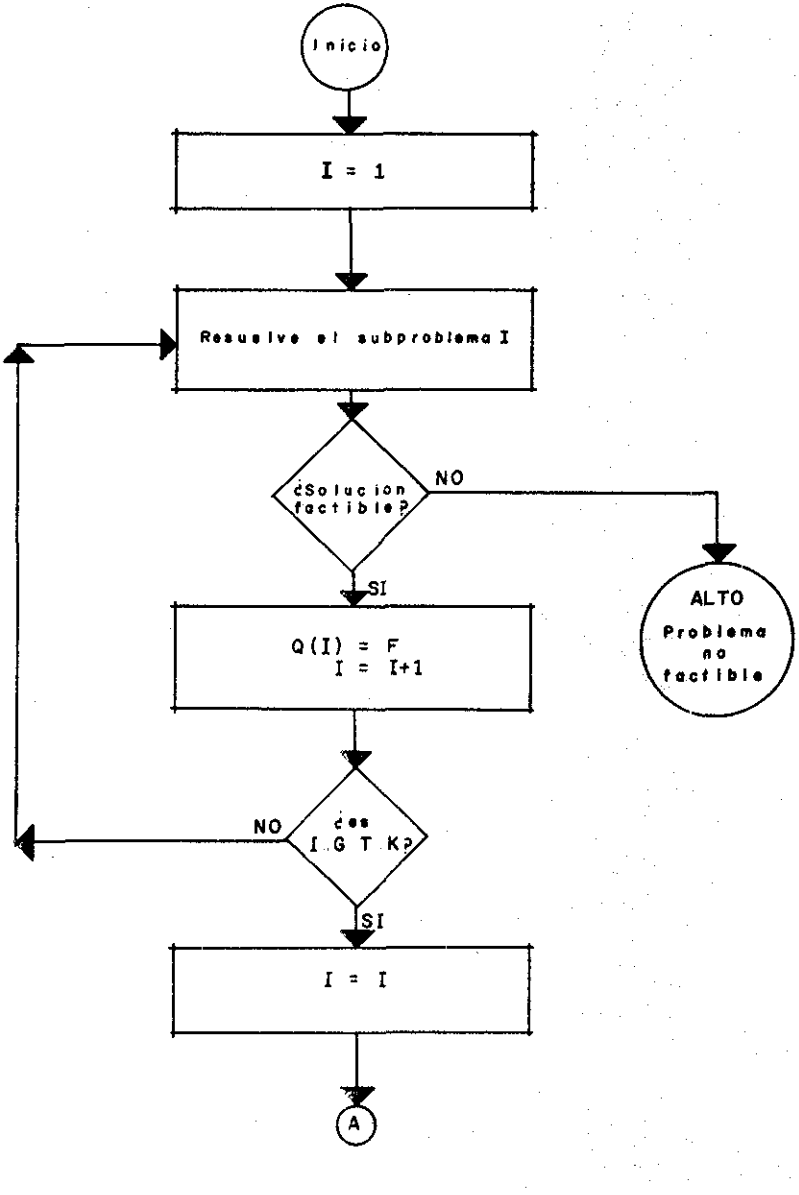
PROBLEMA	NUMERO SUB-PROBLEMAS	ESTRATEGIA	VULT	INFRM	LPNT	VERSION 1 (doble precisión) TIEMPO	NUM ITER	FUNCION OBJETIVO	VERSION 2 (precisión sencilla) TIEMPO	NUM ITER	FUNCION OBJETIVO	VERSION 3 (10 vectores de costos relativos) TIEMPO	NUM ITER	FUNCION OBJETIVO	VERSION 4 (modificación de las superiores) TIEMPO	NUM ITER	FUNCION OBJETIVO
Noroeste*	11	-1	10	30	200	10071	295	8 598 410.5									
		0				0004	244	8 588 408.1									
		1				12404	246	8 588 408.1									
		2				11005	254	8 588 408.1									
		3				12171	279	8 588 408.1									
		4				12137	279	8 588 408.1									
Noroeste	11	0	10	30	150	4571	176	19 020 383.9									

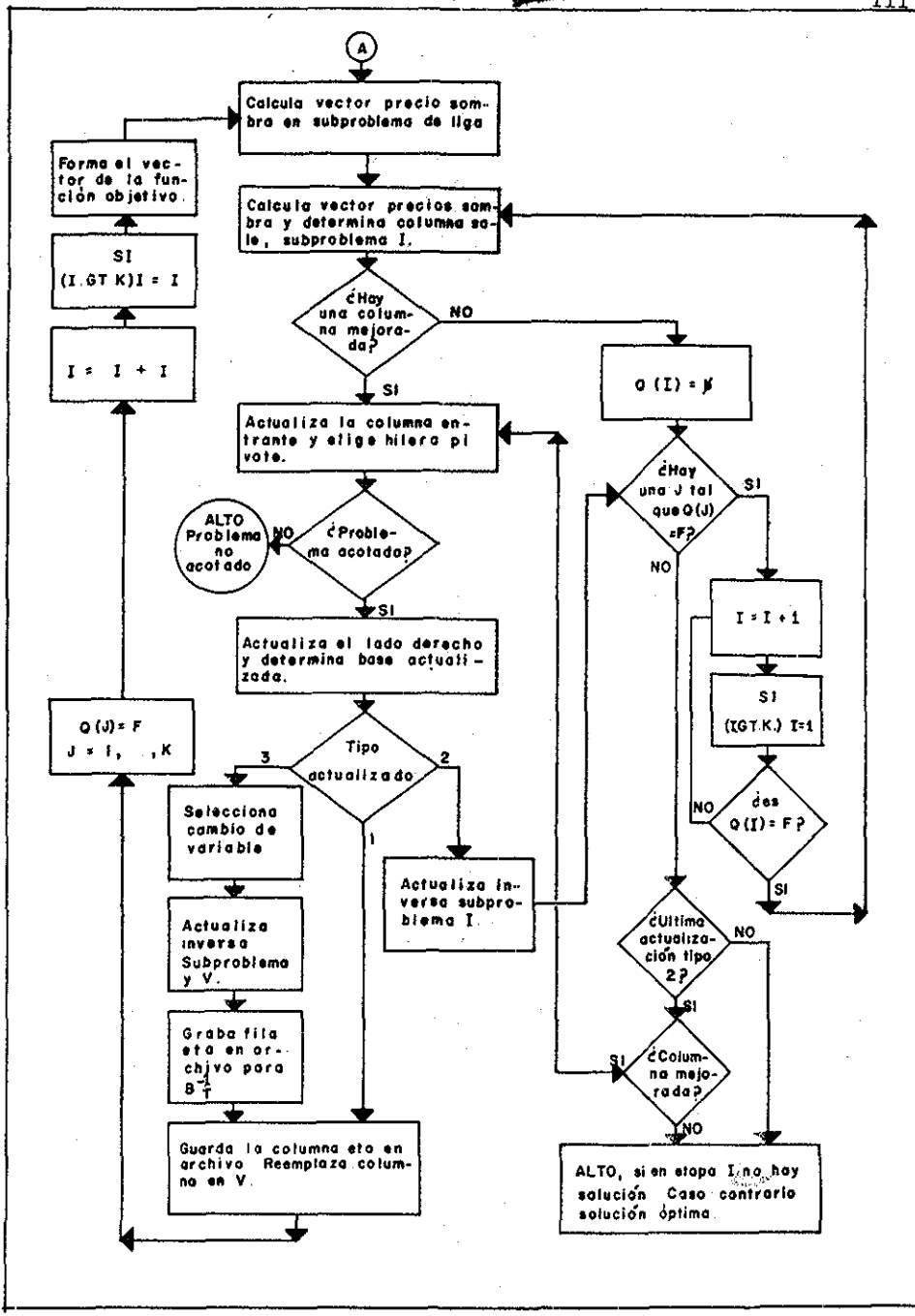
RESULTADOS APEX-III

Valor de la Función objetivo. 8 588 408.177 Tiempo. 52.299

* En este problema no se incluyeron variables y restricciones de liga.

B5 Diagrama de bloques del programa G-GUB.



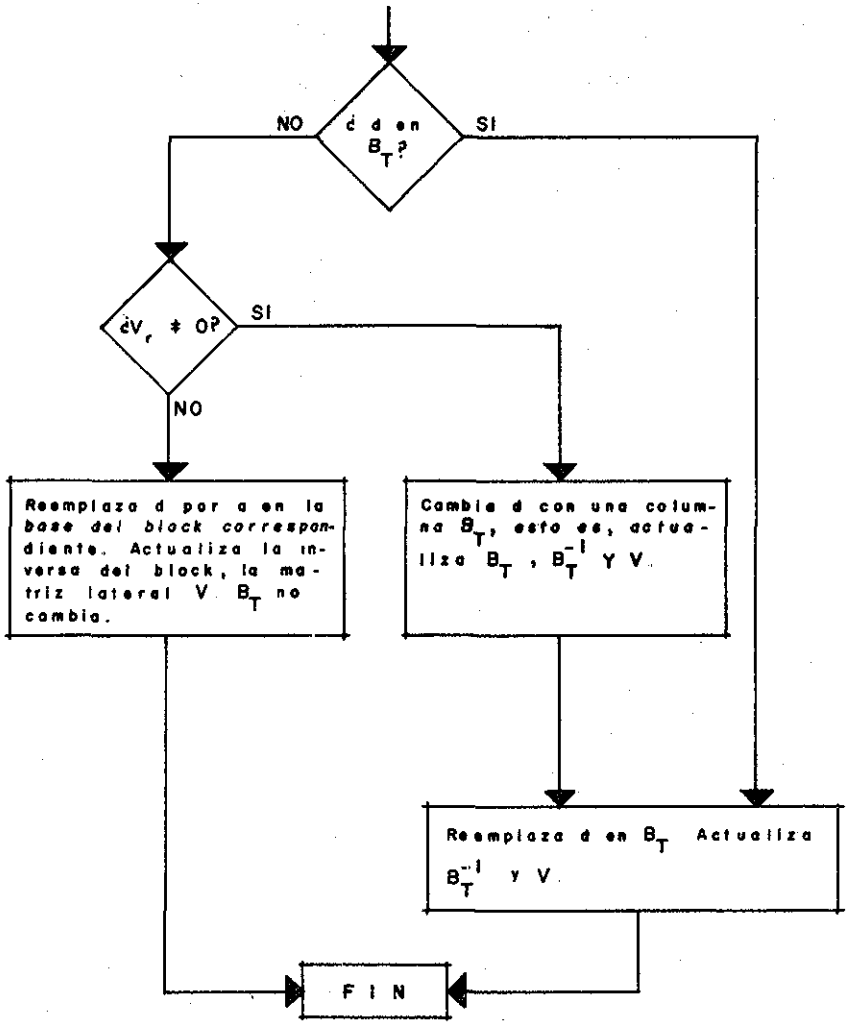


B.5.1 Diagrama de bloques del proceso de actualización de la base.

a. columna que entra a la base.

d. columna que sale de la base.

r. índice de la hilera del block asociado con la columna básica d.



REFERENCIAS

1. Balas, E., (1966) "An Infeasibility-Pricing Descomposition Method for Linear Programs", Operations Research, Vol. 14, No. 5, 847-873.

2. Baumol, W., y Fabian, T. (1964). "Descomposition Pricing for Descentralization and External Economies", Management Science, 11, 1-32

3. Beale, E., (1975). "The Current Algorithm Scope of Mathematical Programming Systems", Mathematical Programming Study 4, 1-11.

4. Beale, E.M.L., (1970) "Advanced Algorithmic Features for General Mathematical Programming Systems", in J. Abadie (ed.), Integer and Nonlinear Programming, North-Holland Publishing Company, London.

5. Beale, E.M.L., (1963) "The Simplex Method Using Pseudo-Basic Variables for Structured Linear Programs", in R.L. Graves and P. Wolfe (eds.), Recent Advances in Mathematical Programming, McGraw-Hill. New York.

6. Bennett, J.M., (1966) "An Approach to Some Structured Linear Programming Problems", Operations Research, Vol. 14, No. 4, 636-645.

7. Brearley, A., Mitra, G., y Williams H. (1975) "Analysis of the Mathematical Programming Problems prior to applying the Simplex Method", Mathematical Programming, 8, 54-83.

8. Crowder, H., y Hatting J. (1975). "Partially normalized Pivot Selection in Linear Programming" Mathematical Programming Study, 4, 12-45.

9. Dantzig, G.B y Van Slyke R.M. (1967) "Generalized Upper Bounded Techniques for Linear Programming", Journal of Computer and System Sciences. Vol. 1, No. 3, 213-226.

10. Forrest, J.J.H. and J.A. Tomlin, (1972) "Updating Triangular Factors of the Basis to Maintain Sparsity in the Product Form Simplex Method, Math. Prog. 2.

11. Fuentes, M.S. (1977). Apuntes de Clase (Teoría y Técnicas de Optimización I). División de Estudios Superiores de la Facultad de Ingeniería, UNAM.

12. Geoffrion, A.M., (1967) "Elements of Large-Scale Mathematical Programming", Management Science, 16, 652-691.

13. Grigoriadis, N.D., (1973) "Unified Pivoting Procedures for Large Structured Linear Systems and Programs", in D.M. Himmelblau (ed.), Descomposition of Large-Scale Problems, North-Holland.

14. Grigoriades, M.D y Ritter K. (1969) "A Descomposition Method for Structured Linear and Nonlinear Programs" , Journal of Computer and System Sciences, Vol. 3, No. 4, 335-360.

15. Hartman, J.K. and L.S. Lasdon, (1970) "A Generalized Upper Bounding Method for Doubly Coupled Linear Programs", Naval Logistic Research Quartely, 411-429.

16. Jennergren, P., (1973) "Price Schedules Descomposition Algorithm for Linear Programming Problems", Econometrica, 41, 965-980.

17. Knowles, T.W., (1973) "An Artificial-Variables Elimination Method for Block-Diagonal Programming Problems", Operations Research, Vol.21, No. 3 pp. 712-727.

18. Lasdon, L.S., (1970) Optimization Theory for Large Systems, Macmillan.

19. Lasdon, L., (1973) "Descomposition in Resource Allocation", in D.M. Himmelblau (ed.) Descomposition of Large Scale Problems, North-Holland, 207-233.

20. Lawrence Jr. J., (1976) "A Dual Descomposition Algorithm", Mathematical Programming 11, 117-193.

21. Muller-Merbach H., (1965) "Das Verfahrender direkten Dekomposition in der Linearen Planungs rechnung", Ablauf-und-Planungs forchung, 6, 306-322.

22. Ohse, D. (1973) "A Dual Descomposition Method for Block Diagonal Linear Programs", Zeitschrift for Operation Research, Vol. 17, 55-67.

23. Olsen Paul. (1975). "A synthesis of compact inverse methods for Block-angular linear programming", Mathematical Programming Study 4, 58-74.

24. Orchard-Hays, W., (1968) Advanced Linear Programming Computing Techniques, McGraw-Hill Book Company, New York.

25. Orchard-Hays, W., (1973) "Practical Problems en LP Descomposition", in D.M. Himmelblau, (ed.) Descomposition of Large Scale Problems, North-Holland.

26. Orchard-Hays, W., (1975) "Factoring LP Block Angular Basis", Mathematical Programming Studies 4, 75-92.

27. Rosen, J.B., (1964) Primal Partition Programming for Block Diagonal Matrices", Numerische Mathemetik, Vol. 6 pp. 250-260.

28. Sakarovitch, M. y Saigal R. (1967) "An extension of Generalized Upper Bounding Techniques for Structured Linear Programming Problems", SIAM. Journal on Applied Mathematics, Vol. 15 No. 4, 906-914.

29. Simmonard M. (1962). "Programación Lineal" Dunod, Paris.

30. Tomlin, J.A., (1974) "On Pricing and Backward Transformation in Linear Programming, Mathematical Programming, Vol. 6, No. 1 pp. 42-47.

31. Tomlin. J., (1975) "An Accuracy test for Updating Triangular Factors", Mathematical Programming Study 4, 142-145.

32. Tomlin J.A., "User Guide for LPM-1" System Optimization Laboratory. Department of Operation Research. Stanford University.

33. Varaiya, P., (1966) Descomposition of Large Scale Systems', Siam Journal of Control, 17, 173-179.

34. Winkler C., (1974) "Basis Factorization for Block-Angular Linear Programs: Unified Theory of Partioning and Descomposition using the simplex Method. Technical Report SOL-74-19. Department of Operation Research Stanford University, Stanford.

```
      PROGRAM GGUB(INPUT,OUTPUT,TAPE5=INPUT, TAPE6=OUTPUT, TAPE8,
     1TAPE9)
      WRITE(6,10)
   10 FORMAT(1H1)
C*  THE NAME G-GUB STANDS FOR 'GENERAL-GENERALIZED UPPER BOUNDING TECHNI
C*********************************************************************
C*********************************************************************
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
C                              VERSION 0                              C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
C     COMMENTS ON DIMENSIONS OF ARRAYS                                C
C                                                                     C
C      NRMAX   =   MAXIMUM NUMBER OF ROWS FOR ANY SUBPROBLEM .        C
C                  (INCLUDING COMMON ROWS IN EACH SUBPROBLEM)         C
C      NROWT   =   NUMBER OF ROWS FOR THE WHOLE PROBLEM               C
C      NEMAX   =   MAXIMUM NUMBER OF ELEMENTS IN  E  ARRAY            C
C      NAMAX   =   MAXIMUM NUMBER OF ELEMENTS IN  A  ARRAY            C
C      NTMAX   =   MAXIMUM NUMBER OF ETA VECTORS                      C
C      NLAMAX  =   MAXIMUM NUMBER OF COLUMNS (INCLUDING ARTIFICALS    C
C                  AND SLACKS) ON ANY SUBPROBLEM                      C
C      KMULT   =   NUMBER OF VECTORS FOR MULTIPLE PRICING             C
C                                                                     C
C      RELATIONS :                                                    C
C                     KPRICE   =  KMULT + 1                           C
C                     KNVEC    =  KMULT + 2                           C
C                     KLAST    =  KNVEC + (NROWT + NRMAX - 1)/NRMAX   C
C                                                                     C
C      DIMENSION  OF                                                  C
C                                                                     C
C      DCHIN , DCOLP , JFROM   IS   KMULT                             C
C      X  ,  Y  ,LROW , LNROW  IS   NROWT                             C
C      YA       IS  (NRMAX,(KLAST+1))                                 C
C      IB       IS  (10 + NTMAX + 2*(NLAMAX + NRMAX))                 C
C      OTHERS AS DEFINED ABOVE                                        C
C                                                                     C
C      WHEN CHANGING DIMENSIONS HAVE TO CHANGE EQUIVALENCE STATEMENT   C
C     FOR  Y  AND  YTEMP  IN ALL. ALSO EQUIVALENCE STATEMENT IN INVERT C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      CALL INDATA
      CALL NORMAL
      CALL NORAML
      STOP
      END
```

SYMBOLIC REFERENCE MAP (R=1)

```
      BLOCK DATA
      IMPLICIT INTEGER (O)
C
C
      COMMON/BLOCK/ZTOLZE,ZTOLPI,ZTOSST,ZTOLRJ,KMULT,KPRICE,KWVEC,KLAST,
     1OBL,OA,OB,OC,OE,OF,OG,OI,OL,OM,ON,OO,OR,OU,NRMAX,NTMAX,NEMAX,NA
     2MAX,NLMAX,NROUT
C
      DATA ZTOLZE/1.E-3/,ZTOLPI/1.E-6/,ZTOSST/1.E-5/,ZTOLRJ/1.E-5/
      DATA   KMULT/10/, KPRICE/11/, KWVEC/12/, KLAST/21/
      DATA OBL/1H /,OA/1HA/,OB/1HB/,OC/1HC/,OE/1HE/,OF/1HF/,OG/1HG/,
     1     OI/1HI/,OL/1HL/,OM/1HM/,ON/1HN/,OO/1HO/,OR/1HR/,
     2     OU/1HU/,NRMAX/115/,NTMAX/500/,NEMAX/4000/,NAMAX/4000/,
     3     NLMAX/350/, NROUT/108/
      END
```

```
      SUBROUTINE INPUT1
C
      IMPLICIT INTEGER (O)
      REAL KITIM,KJTIM,KITINV,KJTINV
C
      COMMON /BLOCK/ZTOLZE,ZTOLPV,ZTCOST,ZTOLRJ,KMULT,KPRICE,KWVEC,KLAST,
     1OBL,OA,OB,OC,OE,OF,OG,OH,OI,OL,OM,ON,OO,OR,OU,NRMAX,NTMAX,NEMAX,NA
     2MAX,NLMAX,IROIT
C
      COMMON /INDSK/ A(4000),IA(4000),LE(602),LA(250),INBAS(250),
     1 RH(115),ISTYPE(115),NROW,NCOL,JRHS,NETA,NELEM,NPR1,NPR2,NPR3
C
      COMMON YA(115,22), X(1035), DCMIN(10), DSUM, DPROD, DY,
     1DE, DP, DELTA, ERMAX,
     1EEMAX,XMAX,COND,SUMINF,SUM,TIMER,KOLA(4,15),KITIM,KJTIM,KITINV,
     2KJTINV,KNAME(10),KTEMP(10),KD,KCASE,KINP,ITEMP(115),JBAS(5,80),
     3ICOLS(250),ICOL(250),LROW(1035),LNROW(1035),NTROW(16),MSTATU(15),
     4 ITEMP(15), ISTAT,IOBJ,IROWP,IVIN,IVOUT,IFFEZ,NLELEM,NLETA,NGELEM,
     5NGETA,JUELEM,JUETA,JUSRHS,LSUB,LMAX,LMAXP1,NTOT,NOLO,NCOLO,NROWO,
     6LCORE, JCOLP(10), JFROM(10), ICOLP, ITSINV, ITSIWB,
     6ITCNT, INVFR, INVFRW,
     7IRTEMP,IRV,JTEMP(14),LS(15),NTIME,MULT,NUETAS(15),ITSITP,ISTRAT
C
      DIMENSION KNAM(250)
      EQUIVALENCE (KNAM(1),YA(461))
C
      CALL SECOND(KITIM)
    5 READ(KINP,700) N1,N2,N3,N4,(KNAME(I),I=1,2),ATEMP1,KNAME(3),ATEMP2
  700 FORMAT(4A1,A10,   A10,   F12.4,3X,A10,    F12.4)
      IF(N1 .NE. OBL)GO TO 50
      IF(L)210,320,410
   50 IF(N1 .EQ. OD)GO TO 100
      IF(N1 .EQ. OR .AND. N2 .EQ. OO)L= - 1
      IF(N1 .EQ. OR .AND. N2 .EQ. OO)GO TO 150
      IF(N1 .EQ. OC)GO TO 300
      IF(N1 .EQ. OB .AND. N2 .EQ. OA)L=1
      IF(N1 .EQ. OB .AND. N2 .EQ. OA)GO TO 150
      IF(N1 .EQ. OR .AND. N2 .EQ. OH)GO TO 500
      GO TO 500
  100 KTEMP(1) = KNAME(2)
  150 WRITE(6,710) N1,N2,N3,N4,KNAME(1),KNAME(2)
  710 FORMAT(3X,4A1,A10,   A10)
      GO TO 5
C
  210 IROW = IROW + 1
      KNAM(IROW) = KNAME(1)
C
C         TEST ROW TYPE
C
      IF(N2 .EQ. OL .OR. N3 .EQ. OL)GO TO 220
      IF(N2 .EQ. OE .OR. N3 .EQ. OE)GO TO 230
      IF(N2 .EQ. OG .OR. N3 .EQ. OG)GO TO 240
      IF(N2 .EQ. ON .OR. N3 .EQ. ON)GO TO 250
      GO TO 230
C
  220 ISTYPE(IROW) = 1
      GO TO 255
```

```
      230 ISTYPE(IRO) = -1
          GO TO 255
      240 ISTYPE(IRO) = 1
          A(IROW) = -1.
          GO TO 250
      250 ISTYPE(IRO) = 0
      255 A(IROW) = 1.
      260 IA(IRO) = IRO
          LA(IRO) = IRO
          JA(IRO) = IRO
          INDAS(IROW) = IROW
          GO TO 5
C
C               MATRIX ELEMENTS
C
      300 IF(LS) .GT. 0 .OR. LMAX .EQ. 0)GO TO 310
          IROW = IRO
      310 ICOL = IRO
          L = 0
          NELEM = IRO
          KCS1 = 1
          GO TO 130
C
      320 J = 2
          K = 4
          IF(KNAME(1) .EQ. KCS1) GO TO 328
C
C                 TEST FOR SPLIT VECTOR
C
          DO 325 I = 1,ICOL
          IF(KNAME(1) .NE. KNAM(I)) GO TO 325
          WRITE(6,8250) KNAME(1)
 8250 FORMAT(1H0,'SPLIT VECTOR ',A10)
      325 CONTINUE
C
          ICOL = ICOL + 1
          KCS1 = KNAME(1)
          KNAM(ICOL) = KCS1
          LA(ICOL) = NELEM + 1
          INDAS(ICOL) = 0
C
C               TEST FOR ROW MATCH
C
      328 IF ( ABS(ATEMP1) .LE. ZTOLZE) GO TO 335
      330 DO 34) I = 1,IROW
          IF(KNAME(J) .NE. KNAM(I)) GO TO 340
          NELEM = NELEM + 1
          IA(NELEM) = I
          A(NELEM) = ATEMP1
      335 IF(K .GT. 5)) TO 5
          IF( ABS(ATEMP2) .LE. ZTOLZE)G) TO 5
          J = 3
          K = 6
          ATEMP1 = ATEMP2
          GO TO 330
      340 CONTINUE
          WRITE(6,3)))  KNAME(J)
```

```
 8300 FORMAT(13H0NO MATCH FOR ROW ,A10)
      STOP
C
C         BASIS CARDS
C
  410 DO 420 I = 1, NCOL
      IF(KNAME(1) .NE. KNAM(I)) GO TO 420
      IBVEC = I
      GO TO 425
  420 CONTINUE
      WRITE(6,8400) KNAME(1)
 8400 FORMAT(21H0NO MATCH FOR VECTOR ,A10)
      GO TO 5
C
  425 DO 430 I = 1, NROW
      IF(KNAME(2) .NE. KNAM(I)) GO TO 430
      IBROW = I
      GO TO 440
  430 CONTINUE
      WRITE(6,8300) KNAME(2)
      GO TO 5
C
  440 IH(IBROW) = IBVEC
      INBAS(IBROW) = 0
      INBAS(IBVEC) = IBROW
      GO TO 5
C
C         RHS
C
  500 IVIN = NCOL
      L = 0
      GO TO 150
C
C         END OF INPUT
C
  600 LA(NCOL+1) = NELEM + 1
      JRHS = IVIN + JUSRHS
      NUMRHS = NCOL - IVIN
      LE(1) = NELEM + 1
      NETA = 0
      IF(LSUB .GT. 0 .AND. LMAXPI .GE. 2)GO TO 615
      DO 605 I=1,NCOL
      INCOL(I) = I
  605 ICOLS(I) = 0
      NCOLO = NCOL
      NOLO = NCOL
  615 ICOLP = NCOL
      NCOL = IVIN
      NSCOL = NCOL - NROW
  620 NELEM = LA(IVIN+1) - 1 - NROW
      RELEM = NELEM
      RDENS = 1.
      IF(NSCOL .EQ. 0)GO TO 625
      RDENS = RELEM/(NROW*NSCOL)
  625 WRITE(6,710) N1,N2,N3,N4,KNAME(1)
      WRITE(6,8500) NROW,NSCOL,NELEM,RDENS
 8500 FORMAT(//1X,18HPROBLEM STATISTICS/1H0,I5,5H ROWS/1H0,I5,
```

```
     1104 STRUCTURAL COLUMNS /1H0,I5,18H NON-ZERO ELEMENTS/
     211,4HDENSITY = ,F7.5)
      CALL SECOND(KJTIM)
      TIMER = KJTIM- KITIM
      WRITE(6,3000)KTEMP(1),TIMER
 3000 FORMAT(/,1X,*INPUT TIME SUBPROBLEM *,A10,* = *,F8.2,* SECONDS*,//)
      RETURN
      END
```

```
      SUBROUTINE FTRAN(KPAR,KVEC)
C
      IMPLICIT INTEGER (O)
      REAL KITIN,KJTIM,KITINV,KJTINV
C
      COMMON/BLOCK/ZTOLZE,ZTOLPV,ZTCOST,ZTOLRJ,KMULT,KPRICE,KWVEC,KLAST,
     1OBL,OA,OB,OC,OE,OF,OG,OH,OI,OL,OM,ON,OO,OR,OU,NRMAX,NTMAX,NEMAX,NA
     2MAX,NLMAX,NROWT
C
      COMMON/INDISK/ A(4000),IA(4000),LE(602),LA(250),INRAS(250),
     1JR(115),ISTYPE(115),NROW,NCOL,JRHS,NETA,NELEM,NPR1,NPR2,NPR3
C
      COMMON YA(115,32), X(1035), DCHIN(10), DSUM, DPROD, DY,
     1DE, DP, DELTA, ERMAX,
     1ZEMAX,KMAX,COLD,SUMINF,SUM,TIMER,KOLA(4,15),KITIM,KJTIM,KITINV,
     2KJTINV,KNAME(10),KTEMP(10),KO,KCASE,KINP,ITEMP(115),JRAS(5,80),
     3ICOLS(250),ICOL(250),LROW(1035),LNROW(1035),NTROW(16),MSTAT(15),
     4ITEMP(15),ISTAT,IOBJ,IROWP,IVIN,IVOUT,IFFEZ,NLELEM,NLETA,NELEM,
     5NETA,NELEM,NETA,JUSRHS,LSUB,LMAX,LMAXP1,NTOT,NOLO,NCOLO,NROWO,
     6LCORE, JCOLP(1)), JFROM(10), ICOLP, ITSINV, ITSIMP,
     6ITCNT, INVFRQ, INVFRW,
     7IRTEMP,INV, JTEMP(14),LS(15),NTIME,MULT,NUETAS(15),ITSITP,ISTRAT
C
      CALL SECOND(KITINV)
      ILE = NETA
      IF(KPAR - 2)100,110,120
  100 IFE = 1
      GO TO 200
  110 IFE = ILETA + 1
      GO TO 200
  120 IFE = NETA
  200 IF (IFE .GT. ILE) GO TO 9000
C
      DO 1000 IK = IFE,ILE
      LL = LE(IK)
      KK = LE(IK+1) - 1
      IF(IA(LL) .LT. 0)GO TO 600
      IF( ABS(YA(IA(LL),KVEC)) .LE. ZTOLZE)GO TO 1000
      DY = YA(IA(LL),KVEC)/A(LL)
      YA(IA(LL),KVEC) = DY
      IF (KK .EQ. LL) GO TO 1000
      LL = LL + 1
      DO 500 J = LL,KK
      YA(IA(J),KVEC) = YA(IA(J),KVEC) - A(J)*DY
  500 CONTINUE
      GO TO 1000
C
  600 IPIV = -IA(LL)
      DSUM = A(LL)*YA(IPIV,KVEC)
      IF(KK .LE. LL)GO TO 900
      LL = LL + 1
      DO 800 J=LL,KK
      DSUM = DSUM + A(J)*YA(IA(J),KVEC)
  800 CONTINUE
  900 YA(IPIV,KVEC) = DSUM
 1000 CONTINUE
 9000 CALL SECOND(KJTIM)
```

```
      C
            KJTINV = KJTINV + KJTIM - KITINV
            RETURN
            END
```

```
      SUBROUTINE BTRAN
C
      IMPLICIT INTEGER (O)
      REAL KITIM,KJTIM,KITINV,KJTINV
C
      COMMON /OLD/OK/ZTOLZE,ZTOLPV,ZTCOST,ZTOLRJ,KMULT,KPRICE,KWVEC,KLAST,
     1OBL,OA,OB,OC,OE,OF,OG,OH,OI,OL,OM,ON,OO,OR,OU,NRMAX,NTMAX,NEMAX,NA
     2MAX,NLMAX,IT)/T
C
      COMMON /INDISK/ A(4000),IA(4000),LE(602),LA(250),INRAS(250),
     1JN(115),ISTYPE(115),NROW,NCOL,JRHS,NETA,NELEM,NPR1,NPR2,NPR3
C
      COMMON YA(115,2), X(1035), DCMIN(10), DSUM, DPROD, DY,
     1DE, DP, DELTA, ERMAX,
     1EEMAX,KMAX,COND,SUMINF,SUM,TIMER,KOLA(4,15),KITIM,KJTIM,KITINV,
     2KJTINV,KJAGE(11),KTEMP(10),KD,KCASE,KINP,ITEMP(115),JBAS(5,80),
     3ICOLS(250),ICOL(250),LROW(1035),LNROW(1035),NTROW(16),MSTATU(15),
     4ITEMP(15),ISTAT,IOBJ,IROWP,IVIN,IVOUT,IFFEZ,NLELEM,NLETA,NGELEM,
     5JGETA,JUELE1,JUETA,JUSRHS,LSUB,LMAX,LMAXP1,NTOT,NOLO,NCOLO,NROWO,
     6LCORE, JCOLP(10), JFROM(10), ICOLP, ITSINV, ITSIWR,
     7ITCNT, ITVFRQ, INVFRM,
     7IRTEMP,INY,JTEMP(14),LS(15),NTIME,NULT,JUETAS(15),ITSITP,ISTRAT
C
      IF (NETA .LE. 0) GO TO 9000
      CALL SECOND(KITINV)
C
      DO 100 I = 1,NETA
      IK = NETA - I + 1
      LL = LE(IK)
      KK = LE(IK+1) - 1
      IPIV = IA(LL)
      IF(IPIV .LE. 0)GO TO 700
      DP = A(LL)
      DY = YA(IPIV,KPRICE)
      DSUM = 0.
      IF (KK .LE. LL) GO TO 600
      LL = LL + 1
      DO 50 J = LL,KK
      DSUM = DSUM + A(J)*YA(IA(J),KPRICE)
  500 CONTINUE
C
  600 YA(IPIV,KPRICE) = (DY - DSUM)/DP
      GO TO 1000
  700 IPIV = -IA(LL)
      DY = YA(IPIV,KPRICE)
      YA(IPIV,KPRICE) = A(LL)*DY
      IF(KK .LE. LL)GO TO 1000
      LL = LL + 1
      DO 80 J=LL,KK
      YA(IA(J),KPRICE) = YA(IA(J),KPRICE) + A(J)*DY
  800 CONTINUE
 1000 CONTINUE
C
      CALL SECOND(KJTIM)
      KJTINV = KJTINV + KJTIM - KITINV
 9000 RETURN
      END
```

```
      SUBROUTINE MRETA(KVEC)
C
      IMPLICIT INTEGER (O)
      REAL KITIM,KJTIM,KITINV,KJTINV
C
      COMMON/ABL/DK,ZTOLZE,ZTOLPV,ZTCOST,ZTOLRJ,KMULT,KPRICE,KWVEC,KLAST,
     1ABL,OA,OB,OC,OE,OF,OG,OH,OI,OL,OM,ON,OO,OR,OU,NRMAX,NTMAX,NEMAX,MA
     2MAX,NLMAX,IRJOUT
C
      COMMON/IBIG/A(4000),IA(4000),LE(602),LA(250),INRAS(250),
     1LL(115),ISTORE(115),NROW,NCOL,JRHS,NETA,NELEM,NPR1,NPR2,NPR3
C
      COMMON YA(115,22), X(1035), DCMIN(10), DSUM, DPROD, DY,
     1DE, DP, DELTA, ERMAX,
     1CEMAX,CMAX,CJD,SUMINF,SUM,TIMER,KOLA(4,15),KITIM,KJTIM,KITINV,
     2KJTINV,KOA(1),KTEMP(1),KD,KCASE,KINP,ITEMP(115),JRAS(5,80),
     3ICOLS(250),IJCOL(250),LROW(1035),LNROW(1035),NTROW(16),MSTATU(15),
     4ITEMP(15),ISTAT,IOBJ,IROWP,IVI,IVOUT,IFFFZ,NLELEM,NLETA,NELEM,
     5JBETA,JELEM,JJETA,JUSRHS,LSU3,LMAX,LMAXP1,NTOT,NOLO,NCOLO,NROWO,
     6LCORE, JCOLP(1), JFROW(10), ICOLP, ITSINV, ITSIWB,
     5ITCNT, IVVER(1), INVERW,
     7IRTEMP,IIZ,JTEMP(14),LS(15),NTIME,MULT,NUETAS(15),ITSITP,ISTRAT
C
      JELEM = JELEM + 1
      IA(NELEM) = IROWP
      IF(JTEMP(3) .EQ. IROWP)IA(NELEM) = - IROWP
      A(NELEM) = YA(IROWP,KVEC)
      ZTOLWR = 0.01* ABS(YA(IROWP,KVEC))
      IF(ZTOLWR .GT. ZTOLZE)ZTOLWR = ZTOLZE
C
      DO 1000 I = 1,NROW
      IF (I .EQ. IROWP) GO TO 1000
      IF ( ABS(YA(I,KVEC)) .LE. ZTOLWR) GO TO 1000
      JELEM = JELEM + 1
      IA(NELEM) = I
      A(NELEM) = YA(I,KVEC)
 1000 CONTINUE
C
      JETA = JETA + 1
      LE(NETA+1) = JELEM + 1
      RETURN
      END
```

```
      SUBROUTINE INVERT
      IMPLICIT INTEGER (A)
      REAL KITIM,KJTIM,KITINV,KJTINV
C
      COMMON/BLOCK/ZTOLZE,ZTOLPV,ZTCOST,ZTOLRJ,KMULT,KPRICE,KWVEC,KLAST,
     1OBL,OA,OB,OC,OE,OF,OG,OH,OI,OL,OM,ON,OO,OR,OU,NRMAX,NTMAX,NEMAX,MA
     2MAX,NLAMAX,NROUT
C
      COMMON/INDISK/ A(4000),IA(4000),LE(602),LA(250),INBAS(250),
     1JH(115),ISTYPE(115),NROW,NCOL,JRHS,NETA,NELEM,NPR1,NPR2,NPR3
C
      COMMON YA(115,22), X(1035), DCMIN(10), DSUM, DPROD, DY,
     1DE, DP, DELTA, ERMAX,
     1EEMAX,XMAX,COND,SUMINF,SUM,TIMER,KOLA(4,15),KITIM,KJTIM,KITINV,
     2KJTINV,KMAME(1),KTEMP(10),KO,KCASE,KINP,ITEMP(115),JBAS(5,80),
     3ICOLS(250),INCOL(250),LROW(1035),LNROW(1035),NTROW(16),MSTATU(15),
     4ITEMP(15),IST\T,IOBJ,IBOMP,IVIN,IVOUT,IFREZ,NLELEM,NLETA,NGELEM,
     5IGETA,NUELEM,NUETA,JUSRNS,LSUB,LMAX,LMAXP1,NTOT,NOLO,NCOLO,NROWO,
     5LCORE, JCOLP(1), JFROW(10), ICOLP, ITSINV, ITSIWR,
     6ITOUT, IUVER), INVERN,
     7IRTEMP,INF,ITEMP(14),LS(15),NTIME,MULT,NUETAS(15),ITSITP,ISTRAT
C
      DIMENSION IRE3(115),LIRE3(115),JVRE3(115)
C
C           SET PARAMETERS
C
      NELEMC = LE(NETA+1) - LE(1)
      WRITE(6,2))LS(1),NETA,NELEMC
   20 FORMAT(/,3X,*INVERSION SUBPROBLEM *,I4,15X,*OLD : *,I4,* ETAS ,*,
     1I6,* ELEMENTS*)
      CALL SECOND(KJTIM)
      DFAC = 0.05
   30 NETA = 0
      NLETA = 0
      NGETA = 0
      NUETA = 0
      NELEM = LE(1) - 1
      NLELEM = 0
      NGELEM = 0
      NUELEM = 0
      NABOVE = 0
      LR1 = 1
      KR1 = 0
      LR4 = NROW + 1
      KR4 = NROW
C
C        PUT SLACKS AND ARTIFICIALS IN PART 4 AND REST IN PART 1
C
      DO 10 I = 1,NROW
      IF (JH(I) .GT. NROW) GO TO 50
      LR4 = LR4 - 1
      IREG(LR4) = JH(I)
      JVREG(LR4) = JH(I)
      GO TO 90
   50 KR1 = KR1 + 1
      JVREG(KR1) = JH(I)
   90 LIREG(I) = -1
```

```
      JH(I) = 0
  100 CONTINUE
C
      KR3 = LR4 -1
      LR3 = LR4
C
      DO 200 I = LR4,KR4
      LHREG(IREG(I)) = 0
      JH(IREG(I)) = IREG(I)
      INBAS(IREG(I)) = IREG(I)
  200 CONTINUE
C
C             PULL OUT VECTORS BELOW BUMP AND GET ROW COUNTS
C
      NJOBJT = KR4 - LR4 + 1
      IF (KR1 .EQ. 0) GO TO 1190
      J = LR1
  210 IV = IVREG(J)
      LL = LA(IV)
      KK = LA(IV+1) - 1
      IRCNT = 0
      DO 22) I = LL,KK
      IF(IA(I) .GT. IROW)GO TO 225
      IBNDNZ = IBNDNZ + 1
      IF (LHREG(IA(I)) .GE. 0) GO TO 220
      IRCNT = IRCNT + 1
      LHREG(IA(I)) = LHREG(IA(I)) - 1
      IRP = IA(I)
  220 CONTINUE
  225 IF (IRCNT - 1) 230,250,300
  230 WRITE(6,3000)IA
 3000 FORMAT(* MATRIX SINGULAR  *,A2)
      INBAS(IV) = 0
      IVREG(J) = IVREG(KR1)
      KR1 = KR1 - 1
      IF (J .GT. KR1) GO TO 310
      GO TO 210
C
  250 IVREG(J) = IVREG(KR1)
      KR1 = KR1 - 1
      LR3 = LR3 - 1
      IVREG(LR3) = IV
      IREG(LR3) = IRP
      LHREG(IRP) = 0
      JH(IRP) = IV
      INBAS(IV) = IRP
      IF (J .GT. KR1) GO TO 310
      GO TO 210
  300 IF (J .GE. KR1) GO TO 310
      J = J+1
      GO TO 210
C                    PULL OUT REMAINING VECTORS ABOVE AND BELOW THE
C                    BUMP AND ESTABLISH MERIT COUNTS OF COLUMNS
C
  310 IVREM = 0
      IF(KR1 .EQ. 0) GO TO 1190
      J = LR1
```

```
      320 IV = JVEC(J)
          LL = LA(IV)
          KK = LA(IV+1) - 1
          IRCNT = 0
          DO 80) I = LL,KK
          IF(IA(I) .GT. IROW)GO TO 805
          IF(LHREG(IA(I)) .NE. -2) GO TO 400
C
C             PUT ABOVE BUMP (PART OF L)
C
          IABOVE = IA(I)/C + 1
          IRUMP = IA(I)
          CALL GIPAC(IV,KPRICE)
          CALL JRETA(KPRICE)
          ILETA = JETA
          JH(IA(I)) = IV
          INBAS(IV) = IA(I)
          JVREG(J) = JVREG(KR1)
          KR1 = KR1 - 1
          NVREM = NVREM + 1
          LHREG(IA(I)) = IV
          GO TO 940
C
      400 IF (LHREG(IA(I)) .GE. 0) GO TO 800
          IRCNT = IRCNT + 1
          IRP = IA(I)
      800 CONTINUE
C
      805 IF (IRCNT - 1) 810,900,1000
      810 WRITE(6,800))))
          INBAS(IV) = 0
          JVREG(J) = JVREG(KR1)
          NVREM = NVREM + 1
          KR1 = KR1 - 1
          IF (J .GT. KR1) GO TO 1010
          GO TO 320
C
C                 PUT VECTOR BELOW BUMP
C
      900 JVREG(J) = JVREG(KR1)
          NVREM = NVREM + 1
          KR1 = KR1 - 1
          LR3 = LR3 - 1
          JVREG(LR3) = IV
          IREG(LR3) = IRP
          LHREG(IRP) = 0
          JH(IRP) = IV
          INBAS(IV) = IRP
C
C           CHANGE ROW COUNTS
C
      940 DO 950 II = LL,KK
          IF(IA(II) .GT. IROW)GO TO 960
          IF (LHREG(IA(II)) .GE. 0) GO TO 950
          LHREG(IA(II)) = LHREG(IA(II)) + 1
      950 CONTINUE
      960 IF (J .GT. KR1) GO TO 1010
```

```
      GO TO 320
 1000 IF (J .GE. KR1) GO TO 1010
      J = J+1
      GO TO 320
 1010 IF(IVREG1 .GT. 0) GO TO 310
C
C                GET MERIT COUNTS
C
 1020 IF (KR1 .EQ. 0) GO TO 1190
      DO 1100 J = LR1,KR1
      LL = LA(JVREG(J))
      KK = LA(JVREG(J)+1) - 1
      IMCNT = 0
      DO 105 I = LL,KK
      IF(IA(I) .GT. IROW)GO TO 1050
      IF (LHREG(IA(I)) .GE. 0) GO TO 1050
      IMCNT = IMCNT - (LHREG(IA(I)) +1)
 1050 CONTINUE
 1060 IREG(J) = IMCNT
 1100 CONTINUE
C
C                  SORT COLUMNS INTO MERIT ORDER
C                  USING SHELL SORT
C
      ISD = 1
 1106 IF (KR1 .LT. 2*ISD) GO TO 1108
      ISD = 2*ISD
      GO TO 1106
 1108 ISD = ISD - 1
C                        END OF INITIALIZATION
 1101 IF (ISD .EQ. 0) GO TO 1107
      ISK = 1
 1102 ISJ = ISK
      ISL = ISK + ISD
      ISY = IREG(ISL)
      ISZ = JVREG(ISL)
 1103 IF (ISY .LT. IREG(ISJ)) GO TO 1104
 1105 ISL = ISJ + ISD
      IREG(ISL) = ISY
      JVREG(ISL) = ISZ
      ISK = ISK + 1
      IF ((ISK + ISD) .LE. KR1) GO TO 1102
      ISD = (ISD - 1) / 2
      GO TO 1101
 1104 ISL = ISJ + ISD
      IREG(ISL) = IREG(ISJ)
      JVREG(ISL) = JVREG(ISJ)
      ISJ = ISJ - ISD
      IF (ISJ .GT. 0) GO TO 1103
      GO TO 1105
 1107 CONTINUE
C                          END OF SORT ROUTINE
C
C          PUT JVT BELOW BUMP ETAS (PART OF U)
C
 1190 ISLCK = 0
      IBELO = 0
```

```
      NELAST = NEMAX
      NTLAST = NTMAX
      LE(NTLAST + 1) = NELAST + 1
C
      LR = LR3
      IF (LR3 .GE. LR4) LR = LR4
      IF (LR .GT. KR4) GO TO 2050
      JK = KR4 + 1
      DO 2000 JJ= LR,KR4
      JK = JK - 1
      NELOUT = NELOUT + 1
      IF (JVREG(JK) .GT. NROW) GO TO 1200
      NBLCK = NBLCK + 1
 1200 LL = LA(JVREG(JK))
      KK = LA(JVREG(JK)+1) - 1
      IF (KK .GT. LL) GO TO 1300
 1250 IF ( ABS(A(LL) - 1.) .LE. ZTOLZE) GO TO 2000
C
 1300 NETA = NETA + 1
      DO 1400 J = LL,KK
      IF(IA(J) .GT. IROW)GO TO 1410
      IF(IA(J) .EQ. IREG(JK))GO TO 1390
      IA(NELAST) = IA(J)
      A(NELAST) = A(J)
      NELAST = NELAST - 1
      NELEM = NELEM + 1
      GO TO 1400
 1390 DP = A(J)
 1400 CONTINUE
 1410 IA(NELAST) = IREG(JK)
      A(NELAST) = DP
      LE(NTLAST) = NELAST
      NELAST = NELAST - 1
      NTLAST = NTLAST - 1
      NELEM = NELEM + 1
 2000 CONTINUE
 2050 IF(KR1 .EQ. 0) GO TO 3600
C
C       DO L-U DECOMPOSITION OF BUMP
C
      DO 3000 J = LR1,KR1
      CALL UNPACK(JVREG(J),KPRICE)
      CALL FTRAN(2,KPRICE)
      ZTOLR = ZTOLZE
 2080 DCMAX = 0.
      IROWP = 0
      IRCMIN = 9999999
      DO 2100 I = 1,NROW
      IF ( ABS(YA(I,KPRICE)) .LT. ZTOLR)GO TO 2100
      IF (LNRCS(I) .GE.0) GO TO 2100
      IF( ABS(YA(I,KPRICE)) .GT. DCMAX)DCMAX =  ABS(YA(I,KPRICE))
      IF (LNRCS(I) .LE. IRCMIN) GO TO 2100
      IRCMIN = LNRCS(I)
      IROWP = I
 2100 CONTINUE
      IF (IROWP .GT. 0) GO TO 2150
      IF(DCMAX .LT. ZTOLZE)GO TO 2125
```

```
      ZTOLR = ZTOLZE*ZTOLZE
      WRITE(6,30000)I
      GO TO 3000
 2125 WRITE(6,30000)I
      INBAS(JVREG(J)) = 0
      GO TO 3000
C
 2150 IF( ABS(YA(IROWP,KPRICE)) .GE. DCMAX*DFAC)GO TO 2155
      ZTOLR = DCMAX*DFAC
      GO TO 2000
 2155 INCR = LREG(IROWP) + 3
C
C              WRITE L AND U ETAS
C
      ZTOLET = 0.001* ABS(YA(IROWP,KPRICE))
      IF(ZTOLET .GT. ZTOLZE)ZTOLET = ZTOLZE
      IF (J .EQ. KR1) GO TO 2160
      NELEM = NELEM + 1
      IA(NELEM) = IROWP
      A(NELEM) = YA(IROWP,KPRICE)
 2160 DO 2300 I = 1,IROW
      IF (I .EQ. IROWP) GO TO 2300
      IF( ABS(YA(I,KPRICE)) .LT. ZTOLET) GO TO 2300
      IF (LREG(I) .GE. 0) GO TO 2200
C
C               L ETA ELEMENTS
C
      NELEM = NELEM + 1
      IA(NELEM) = I
      A(NELEM) = YA(I,KPRICE)
      GO TO 2300
C
C               U ETA ELEMENTS
C
 2200 IA(NELAST) = I
      A(NELAST) = YA(I,KPRICE)
      NELAST = NELAST - 1
      NELEM = NELEM + 1
 2300 CONTINUE
C
      JH(IROWP) = JVREG(J)
      INBAS(JVREG(J)) = IROWP
      NETA = NETA + 1
      IA(NELAST) = IROWP
      IF (J .NE. KR1) GO TO 2330
      A(NELAST) = YA(IROWP,KPRICE)
      GO TO 2340
 2330 A(NELAST) = 1.
      NETA = NETA + 1
      LE(NETA+1) = NELEM + 1
 2340 NELEM = NELEM + 1
      LE(NTLAST) = NELAST
      NELAST = NELAST - 1
      NTLAST = NTLAST - 1
C
C             UPDATE ROW COUNTS
C
```

```
      DO 2350 I = 1, IROW
      IF ( ABS(YA(I,KPRICE)) .LE. ZTOLET) GO TO 2350
      IF (LAREG(I) .GE. 0) GO TO 2350
      LAREG(I) = LAREG(I) - INCR
      IF (LAREG(I) .GE. 0) LAREG(I) = -1
 2350 CONTINUE
      LAREG(IROWP) = 0
 3000 CONTINUE
C
C                SHIFT IE AND E OF U ELEMENTS
C
 3500 NLETA = NETA
      NETA = NLETA + NUETA
      NLELEM = NELEM
      NELEM = NLELEM + NUELEM
      IF (NUELEM .EQ. 0) GO TO 3550
      IF = NEMAX - NUELEM + 1
      INCR = 0
      DO 3510 I = IF,NEMAX
      INCR = INCR + 1
      IA(NLELEM + INCR) = IA(I)
      A(NLELEM + INCR) = A(I)
 3510 CONTINUE
      IDIF = NEMAX - NLELEM - NUELEM
      IF = NTMAX - NUETA + 1
      INCR = 0
      DO 3520 I = IF,NTMAX
      INCR = INCR + 1
      LE(NLETA + INCR) = LE(I) - IDIF
 3520 CONTINUE
      LE(NETA+1) = NELEM + 1
C
C                   INSERT SLACKS FOR DELETED COLUMNS
C
 3550 DO 3600 I = 1,IROW
      IF (JH(I) .NE. 0) GO TO 3600
      JH(I) = I
      IROWP = I
      CALL UNPACK(I,KPRICE)
      CALL FTRAN(1,KPRICE)
      CALL WRETA(KPRICE)
 3600 CONTINUE
C
C        CHECK ACCURACY OF INVERSE
C
      CALL CHSOL(4)
      NELEME = NELEM + 1 - LE(1)
      NSTR = IROW - NSLCK
      WRITE(6,4000)ERMAX,NETA,NELEME,NSTR,NONONZ
 4000 FORMAT(3X,*MAXIMUM ROW ERROR = *,E14.5,6X,*NEW : *,I4,* ETAS ,*,I6
     1,* ELEMENTS*,4X,I4,* STRUCTURAL COLUMNS*,I9,* NON-ZEROES*,/)
      IF(ERMAX .LE. ZTOLRJ)GO TO 5000
      IF(DFAC .GT. 1)GO TO 5000
      DFAC = DFAC*5
      IF(DFAC .GT. 1.)DFAC = 1.
      GO TO 10
 5000 IF(K[illegible] .EQ. 1)GO TO [illegible]
```

```
              DO 5200 J=1,NROW
              X(J) = YA(J,KPRICE)
              JROW(J) = JH(J)
              JROW(J) = 1
              IF(JH(J) .LE. NROW)JROW(J) = ISTYPE(JH(J))
         5200 CONTINUE
         5500 CALL SECOND(KUTIMV)
              KUTIMV = KUTIMV -KUTIM
        C
              RETURN
              END
```

```
SUBROUTINE UNPACK      73/173   OPT=1                    FTN 4.6+460      12
                                                                        136

1                SUBROUTINE UNPACK(IV,KVEC)
           C
                 IMPLICIT INTEGER (G)
                 REAL KITIN,KJTIM,KITINV,KJTINV
5                COMMON YA(115,22), X(1035), DCMIN(10), DSUM, DPROD, DY,
                1DE, DP, DELTA, ERMAX,
                1SEMAX,RHS,COND,SUBIMF,SUM,TIMER,KOLA(4,15),KITIM,KJTIM,KITINV,
                2KJTINV,KUMBE(1),KTEMP(10),KO,KCASE,KINP,ITEMP(115),JBAS(5,80),
                3ICOLS(250),IROL(250),LROW(1035),LIROW(1035),NTROW(16),MSTATU(15),
10               4ITEMP(15),ISTAT,IORJ,IROWP,IVIN,IVOUT,IFEFZ,NLELEM,NLETA,NELEM,
                5IGETA,IUELTA,IUETA,JUSRHS,LSUB,LMAX,LMAXP1,NTOT,NOLO,NCOLO,NROWO,
                6SCORE, JCOLP(1)), JPROW(10), ICOLP, ITSINV, ITSINB,
                6ITCNT, IN/FR), INVERW,
                7IRTEMP,INY,ITEMP(14),LS(15),NTIME,MULT,NUETAS(15),ITSITP,ISTRAT
15         C
                 COMMON/INDISK/ A(4000),IA(4000),LE(602),LA(250),INBAS(250),
                1IA(115),ISTYPE(115),NROW,NCOL,JRHS,NETA,NELEM,NPR1,NPR2,NPR3
           C
                 DO 10) I = 1,NROW
20               YA(I,KVEC) = 0.
             100 CONTINUE
                 LL = LA(IV)
                 KK = LA(IV+1) - 1
                 IF(KK .LT. LL)GO TO 500
25               DO 20) I = LL,KK
                 YA(IA(I),KVEC) = A(I)
             200 CONTINUE
             500 RETURN
                 END
```

```
      SUBROUTINE ITEROP(KPAR,KVEC,LSTAT)
C
      IMPLICIT INTEGER (O)
      REAL KITIM,KUTIM,KITINV,KUTINV
C
      COMMON /MOLDCK/ZTOLZE,ZTOLPV,ZTOCST,ZTOLRJ,KMULT,KPRICE,KWVEC,KLAST,
     1OBL,OA,OB,OC,OE,OF,OG,OH,OI,OL,OM,ON,OO,OR,OU,NRMAX,NTMAX,NEMAX,MA
     2IAX,NLAMAX,NROUT
C
      COMMON/INBOCK/ A(4000),IA(4000),LE(602),LA(250),INBAS(250),
     1JH(115),ISTYPE(115),NROW,NCOL,JRHS,NETA,NELEM,NPR1,NPR2,NPR3
C
      COMMON YA(115,22), X(1035), DCMIN(10), DSUM, DPROD, DY,
     1DE, DP, DELTA, ERMAX,
     1EEMAX, OMAX, ORND, SUMINF,SUM,TIMER,KOLA(4,15),KITIM,KUTIM,KITINV,
     2KUTINV,KINDE(10),KTEMP(10),KD,KCASE,KINP,ITEMP(115),JBAS(5,60),
     3ICOLS(250),INCOL(250),LROW(1035),LNROW(1035),NTROW(16),MSTATU(15),
     4JTEMP(15), ISTAT,IOBJ,IROWP,IVIN,IVOUT,IFEEZ,NLELEM,NLETA,NSELEM,
     5ISETA, NCOLS, NETA,JUSRHS,LSUB,LMAX,LMAXP1,NTOT,NOLO,NCOLO,NROWO,
     6LCORE, JCOLP(10), JFROM(10), ICOLP, ITSINV, ITSIWB,
     8ITCNT, INVFRQ, INVFRW,
     7IRTEMP,IDY,ITEMP(14),LS(15),MTIME,MULT,NUETAS(15),ITSITP,ISTRAT
C
      IF (KPAR .EQ. 0)GO TO 1000
      OBJ = - X(IOBJ)
      IF (MSTAT .EQ. OI .OR. MSTAT .EQ. ON)OBJ = SUMINF
      CALL SECOND(KUTIM)
      TIMER = KUTIM- KITIM
      KDLA = NCOLO - NOLO
      KDIA = LA(NCOLO+1) - LA(NOLO+1)
      LCASE = IRTEMP + 1
      CMIN = DCMIN(KVEC)
      KK = JTEMP(6)
      IF(KCASE .NE. 1)GO TO 50
      KK = LSUB
      JTEMP(2) = LSUB
      LCASE = 0
   50 LL = KK
      IF(KK .EQ. 0)LL = JTEMP(6)
      NELEME = NELEM + 1 - LE(1)
      WRITE(6,90)ITCNT,LSTAT,MSTAT,OBJ,JTEMP(1),JTEMP(2),JTEMP(12),LL,KK
     1,CMIN,NETA,NELEME,TIMER,KDLA,KDIA,LCASE
   90 FORMAT(1H ,I5,1XA2,2XA2,2X,F16.3,2(I8,I6),I4,2X,F16.3,I9,I9,F8.2,3
     1I5)
      DO 100 I=1,12
  100 JTEMP(I) = 0
      GO TO 9000
 1000 WRITE(6,150)
  150 FORMAT(//,* ITCOUNT  STATUS    OBJ.VALUE*,8X,*VECIN FROM *,
     1* VECOUT FROM  IN*,7X,* DJ*,12X,*NETA    NELEM    TIME  CLA   DIA*,
     2* CASE*,//)
 9000 RETURN
      END
```

```
      SUBROUTINE [illegible]
C
      IMPLICIT INTEGER (O)
      REAL KITIM, KJTIM, KITINV, KJTINV, KITIMN, KKITIMN
C
      COMMON /BLOCK/ZTOLZE,ZTOLPV,ZTCOST,ZTOLRJ,KMULT,KPRICE,KWVEC,KLAST,
     1OBL,OA,OB,OC,OE,OF,OG,OH,OI,OL,OM,ON,OO,OR,OU,NPMAX,NTMAX,NEMAX,NA
     2MAX,NLAMA,[illegible]
C
      COMMON/INDISK/ A(4000),IA(4000),LE(602),LA(250),INBAS(250),
     1[illegible](115),ISTYPE(115),NROW,NCOL,JRHS,NETA,NELEM,NPR1,NPR2,NPR3
C
      COMMON /ACCDIR/ BRR1(201),KXTEM(251)
C
      COMMON YA(115,32), X(1035), DCHIN(10), DSUM, DPROD, DY,
     1DE, DP, DELTA, ERMAX,
     1[illegible]MAX,[illegible],SUMINF,SUM,TIMER,KOLA(4,15),KITIM,KJTIM,KITINV,
     2KJTINV,KNAME(10),KTEMP(10),KD,KCASE,KINP,ITEMP(115),JBAS(5,80),
     3ICOLS(250),ICOL(250),LROW(1035),LNROW(1035),NTROW(16),MSTATU(15),
     4ITEMP(15),[illegible]STAT,IOBJ,IROWP,IVIN,IVOUT,IFFEZ,NLELEM,NLETA,NGELEM,
     5NGETA, [illegible], [illegible]ETA,JUSPHS,LSUB,LMAX,LMAXP1,NTOT,NOLO,NCOLO,NROWO,
     6LCORE, JCOLP(10), JFROM(10), ICOLP, ITSINV, ITSIWB,
     6ITCNT, INVFRQ, INVFRW,
     7IRTEMP,INY,ITEMP(14),LS(15),NTIME,MULT,NUETAS(15),ITSITP,ISTRAT
C
      COMMON /ACLOCK/ KITIMN
C
      DIMENSION KNAM(250)
      EQUIVALENCE (KNAM(1),YA(461))
C
      IF(KCASE .EQ. 1)GO TO 60
      LSUB = 0
      CALL CHANGE(LSUB)
      DO 50 I=1,[illegible]
   50 ITEMP(I) = [illegible](I)
   60 DO 5000 I=1,LMAXP1
      IF(KCASE .EQ. 1)GO TO 70
      LSUB = I - 1
      CALL CHANGE(LSUB)
      CALL KBASIS(2,KOUT)
      CALL FORMC
      CALL BTRAN
   70 KD = 2
      IF(I .NE. 1)KD = KOLA(4,LSUB)
      WRITE(6,8300)LSUB
 8300 FORMAT(I3)
      CALL READMS(2,KXTEM, 251,KD)
      KNAME(1)=KXTEM(1)
      DO 85 II=2,251
      KNAM(II-1) = KXTEM(II)
   85 CONTINUE
      IF(I.GE. 2)GO TO 100
      WRITE(6,8000) KNAME(1)
 8000 FORMAT(1H1,*      PROBLEM  *,A10//)
      WRITE(6,8100)
 8100 FORMAT(6H JN(I),1[illegible]X,5HVALUE,9X,9HROW NAMES,9X,5HPI(I),12X,3HRHS//)
  100 KVEC = 2
```

```
      IF(LS(1) .EQ. 0)KVEC = 1
      CALL INPAUX(JRIS,KVEC)
      DO 10 J = 1,PROB
      IF(J .GT. 10) .OR. KCASE .EQ. 1)GO TO 200
      IF(ICOLS(ITEMP(J)) .NE. LS(2))GO TO 1000
      KK = J
      KVEC = 1
      IF(LS(1) .EQ. 0)GO TO 150
      IV = ICOL(ITEMP(J))
      GO TO 300
  150 IV = ITEM(J)
      GO TO 300
  200 IV = PI(J)
      KK = ITR)/(LS(3) + J - NROW
      KVEC = 2
  300 WRITE(6,32) KIAM(IV), X(KK), KIAM(J), YA(J,KPRICE), YA(J,KVEC),
     1 IV, KK, J,KPRICE, KVEC
   32 FORMAT(1H ,A1)  ,6X,F16.3,4X,A1)  ,2X,2F16.3 ,5I6)
 1000 CONTINUE
 5000 CONTINUE
      CALL SECOND(KKITIME)
      TIMER = KKITIME - KITIME
      WRITE(6,9000)TIMER
 9000 FORMAT(//,*  TOTAL INPUT-OUTPUT TIME  = *,F8.2,*  SECONDS*)
      RETURN
      END
```

```
      SUBROUTINE PACK(KVEC,IIFROM)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      REAL KITIM,KJTIM,KITINV,KJTINV
C
      COMMON/ZTOL/ZTOLZE,ZTOLPV,ZTCOST,ZTOLRJ,KMULT,KPRICE,KWVEC,KLAST,
     1IBL,IA,IB,IC,IE,IF,IG,IH,II,IL,IM,IN,IO,IR,IU,NRMAX,NTMAX,NEMAX,NA
     2MAX,NLMAX,IRUNIT
C
      COMMON/MIP/IBSKA(4000),IA(4000),LE(602),LA(250),INBAS(250),
     1IBR(115),IBTYPE(115),NROW,NCOL,JRHS,NETA,NELEM,NPR1,NPR2,NPR3
C
      COMMON YA(115,22), X(1035), DCMIN(10), DSUM, DPROD, DY,
     1DE, DP, DELTA, ERMAX,
     1EEMAX,KMAX,COL1),SUMINF,SUM,TIMER,KOLA(4,15),KITIM,KJTIM,KITINV,
     2KJTINV,KNAME(1)),KTEMP(10),KD,KCASE,KINP,ITEMP(115),JBAS(5,80),
     3ICOLS(250),INCOL(250),LROW(1035),LNROW(1035),NTROW(16),MSTATU(15),
     4ITEMP(15),ISTAT,IOBJ,IROWP,IVIN,IVOUT,IFFEZ,NLELEM,NLETA,NGELEM,
     5NGETA,NJELEM,NJETA,JUSRHS,LSUB,LMAX,LMAXP1,NTOT,NOLO,NCOLO,NROWO,
     6LCORE, JCOLP(1)), JFROM(10), ICOLP, ITSINV, ITSINR,
     7ITOUT, INVFRQ), INVERW,
     7IRTEMP,INV,JTOP(14),LS(15),NTIME,MULT,NUETAS(15),ITSITP,ISTRAT
C
  100 IF(LA(ICOL)+1) .GT. (LE(1) - NRMAX))GO TO 1000
      NELA = LA(ICOL)+1) - 1
      KK = NTROW(IIFROM + 1) - NTROW(IIFROM) + NROWO
      DO 50) I=1,KK
      IF( ABS(YA(I,KVEC)) .LE. ZTOLZE)GO TO 500
      NELA = NELA + 1
      IA(NELA) = I
      A(NELA) = YA(I,KVEC)
  500 CONTINUE
      ICOLO = ICOL) + 1
      LA(NCOLO+1) = NELA + 1
      GO TO 50))
C
 1000 M = ICOL) + 1
      N = ICOL)
      ICOLO = ICOL)
      NELA = LA(ICOL) + 1) - 1
      DO 20)) I=M,N
      IF(ICOLS(I) .EQ. 0)GO TO 2000
      LL = LA(I)
      KK = LA(I + 1) - 1
      DO 120)) J=LL,KK
      NELA = NELA + 1
      IA(NELA) = IA())
      A(NELA) = A())
 1200 CONTINUE
      ICOLO = ICOL) + 1
      LA(NCOLO + 1) = NELA + 1
      ICOLS(ICOL)) = ICOLS(I)
      INCOL(ICOL)) = INCOL(I)
      INBAS(ICOL)) = INBAS(I)
      IM = INBAS(I)
      JJ(IM) = ICOL)
 2000 CONTINUE
```

```
      IF(IIFROM .NE. 1000 TO 100
 5000 RETURN
      END
```

```
      SUBROUTINE CHANGE(LSUP)
C
      IMPLICIT INTEGER (O)
      REAL KBW,KAR,KITIM,KJTIM,KITINV,KJTINV
C
      COMMON/BLOCK/ZTOLZE,ZTOLPV,ZTCOST,ZTOLRJ,KMULT,KPRICE,KWVEC,KLAST,
     1OBL,OA,OB,OC,OE,OF,OG,OH,OI,OL,OM,ON,OO,OR,OU,NRMAX,NTMAX,NEMAX,NA
     2MAX,NLAMAX,IROUT
C
      COMMON/BIGBCK/ A(4000),IA(4000),LE(602),LA(250),INBAS(250),
     1IH(115),ISTORE(115),NROW,NCOL,JRHS,NETA,NELEM,NPR1,NPR2,NPR3
C
      COMMON YA(115,22), X(1033), DCMIN(10), DSUM, DPROD, DY,
     1DE, OP, DELTA, ERMAX,
     1EEMAX,KIN(?,CODE),SUMINF,SUM,TIMER,KOLA(4,15),KITIM,KJTIM,KITINV,
     2KJTINV,KNAME(10),KTEMP(10),KO,KCASE,KINP,ITEMP(115),JBAS(5,80),
     3ICOLS(250),IHCOL(250),LROW(1035),LNROW(1035),NTROW(16),MSTATU(15),
     4JTEMP(15),ISTAT,IOBJ,IROWP,IVIN,IVOUT,IFFEZ,NLELEM,NLETA,NGELEM,
     5NGETA,NELLM,NUETA,JUSRHS,LSUB,LMAX,LMAXP1,NTOT,NOLO,NCOLO,NROWO,
     6LCORE, JCOLP(10), JFROM(10), ICOLP, ITSINV, ITSIWB,
     8ITCNT, INVFRQ, INVFRW,
     7IRTEMP,IDY,JTEMP(14),LS(15),NTIME,MULT,NUETAS(15),ITSITP,ISTRAT
C
      DIMENSION JB(1340)
      EQUIVALENCE (JB(1),LE(1))
C
      KL = LSUP
      IF(LSUP .EQ. 0)KL = LMAXP1
      IF(KL .EQ. LCORE)GO TO 5000
      CALL SECOND(KBW)
      IF(JTEMP(14) .EQ. 0)GO TO 3000
      KOLA(1,LCORE) = LE(NETA +1)
      KIS =1
 2050 KO = KOLA(3,LCORE)
      CALL VECTOR(KIS, A, IA, JB, KOLA(1,LCORE))
      IF(LCORE .EQ. KL)GO TO 4000
 3000 LCORE = KL
      KIS = 2
      GO TO 2050
 4000 CALL SECOND(KAR)
      NTIME = NTIME + KAR - KBW
      JTEMP(14) = 0
 5000 RETURN
      END
```

```
      SUBROUTINE INVRSE
C
      IMPLICIT INTEGER (Q)
      REAL KITIM,KJTIM,KITINV,KJTINV
C
      COMMON /BLOCK1/ZTOLZE,ZTOLPV,ZTCOST,ZTOLRJ,KMULT,KPRICE,KWVEC,KLAST,
     1QBL,QA,QB,QC,QE,QF,QG,QH,QI,QL,QM,QN,QO,QR,QU,JRMAX,NTMAX,NEMAX,NA
     2MAX,NLMAX,NROWT
C
      COMMON/BIGBK/ A(4000),IA(4000),LE(602),LA(250),INBAS(250),
     1 JH(115),ISTYPE(115),NROW,NCOL,JRHS,NETA,NELEM,NPR1,NPR2,NPR3
C
      COMMON YA(115,22), X(1035), DCMIN(10), DSUM, DPROD, DY,
     1DE, DP, DELTA, ERMAX,
     1EEMAX,KI(4,10)),SUMINF,SUM,TIMER,KOLA(4,15),KITIM,KJTIM,KITINV,
     2KJTINV,KLIM(10)),KTEMP(10),KD,KCASE,KINP,ITEMP(115),JBAS(5,80),
     3ICOLS(250),ICOL(250),LROW(1035),LNROW(1035),NTROW(16),MSTATU(15),
     4 ITEMP(15),ISTAT,IOBJ,IROWP,IVIN,IVOUT,IFFEZ,NLELEM,NLETA,NGELEM,
     5NGETA,NUELEM,NUETA,JUSRHS,LSUB,LMAX,LMAXP1,NTOT,NCLO,NCOLO,NROWO,
     5LCORE, JCOLP(10), JFROM(10), ICOLP, ITSINV, ITSIMP,
     6ITCNT, INVFRQ, INVFRW,
     7IRTEMP,INY,ITEMP(14),LS(15),NTIME,MULT,NUETAS(15),ITSITP,TSTRAT
C
      DIMENSION XY(115)
      EQUIVALENCE (YA(1),XY(1))
C
      WRITE(6,300)
  300 FORMAT(//,3X,*BEGIN COMPLETE INVERSION*,/)
      LSUB = 0
      CALL SAVEIS(LSUB)
      ITMA = 0
      DO 500 I=1,NROW
      YA(I,3) = 0.
      ITEMP(I) = JH(I)
  500 CONTINUE
      DO 1000 I=1,LMAX
      LSUB = I
      CALL CHANGE(LSUB)
      KJ = 1
      CALL XBASIS(KJ,KOUT)
      IF(KO .EQ. 1)GO TO 700
      CALL CHSOL(4)
      IF(ERMAX .LE. ZTOLRJ)GO TO 820
  700 CALL INVERT
      ITEMP(14) = 1
      NUETAS(LSUB) = 0
  750 DO 800 J=1,NROWO
      IF(ICOLS(ITEMP(J)) .NE. LSUB)GO TO 800
      ICOLS(ITEMP(J)) = 0
      IV = ICOL(ITEMP(J))
      CALL UNPACK(IV,1)
      CALL FTRAN(1,1)
      ITMA = ITMA + 1
      WRITE(3) J,LSUB,IV,NROW,XY
  800 CONTINUE
  820 LL = NTROW(I)
      DO 900 J=1,NROW
```

```
          IF(J .GT. IROW(I))GO TO 850
          YA(J,3) = YA(J,3) + YA(J,KPRICE)
          GO TO 900
      850 LL = LL + 1
          X(LL) = YA(J,KPRICE)
          LNROW(LL) = JH(J)
          LROW(LL) = 1
          IF(JH(J) .LE. IROWO)LROW(LL) = ISTYPE(JH(J))
      900 CONTINUE
     1000 CONTINUE
          END FILE 3
          REWIND 3
          LSUB = 0
          CALL CHANGE(LSUB)
          IF(ITMA .EQ. 0)GO TO 1500
          ILE = LE(1)
          LE(1) = LA(ICOLO+1)
          CALL PACK(1,LSUB)
          LE(1) = ILE
          DO 1200 I=1,ITMA
          READ(3)IROWP,INFROM,IVIN,ITYPE,XY
          CALL PACK(1,INFROM)
          JH(IROWP) = ICOLO
          INBAS(ICOLO) = IROWP
          ICOLS(ICOLO) = INFROM
          INCOL(ICOLO) = IVIN
     1200 CONTINUE
          REWIND 3
     1500 CALL INVERT
          ITEMP(14) = 1
          ITSITP = 0
          DO 1300 I=1,IROW
          IF(INBAS(ITEMP(I)) .NE. 0)GO TO 1300
          ITSITP = ITSITP + 1
          JBAS(3,ITSITP) = ICOLS(ITEMP(I))
          JBAS(4,ITSITP) = 0
          JBAS(5,ITSITP) = INCOL(ITEMP(I))
     1300 CONTINUE
          CALL UPDATE(JBAS,1)
          DO 1800 I=1,IROW
     1800 YA(I,1) = YA(I,1) + YA(I,3)
          CALL FTRAN(1,1)
          DO 2500 I=1,IROW
          X(I) = YA(I,1)
          LNROW(I) = JH(I)
          LROW(I) = 1
          IF(JH(I) .LE. IROWO)LROW(I) = ISTYPE(JH(I))
          JSUB = ICOLS(JH(I))
          IF(JSUB .EQ. 0)GO TO 2500
          LNROW(I) = INCOL(JH(I))
          LL = LA(JH(I))
          KK = LA(JH(I) + 1) - 1
          DO 2000 J=LL,KK
          IF(IA(J) .LE. IROWO)GO TO 2000
          IR = ITROW(JSUB) + IA(J) - NROJO
          X(IR) = X(IR) - A(J)*X(I)
     2000 CONTINUE
```

```
15          2500 SOLUTION
                 RETURN
                 END
```

```
      SUBROUTINE INDATA
C
      IMPLICIT INTEGER (O)
      REAL KITIM,KJTIM,KITINV,KJTINV,KITIMN,KJTIMN
C
      COMMON /XBLOCK/ZTOLZE,ZTOLPV,ZTCOST,ZTOLRJ,KMULT,KPRICE,KWVEC,KLAST,
     1OBL,OA,OB,OC,OI,OE,OG,OH,OI,OL,OM,ON,OO,OR,OU,NRMAX,NTMAX,NEMAX,NA
     2MAX,NLMAX,IROUT
C
      COMMON /BLOCK/ A(4000),IA(4000),LE(602),LA(250),INBAS(250),
     1 NB(115),ISTAT(115),IROW,ICOL,IRHS,NETA,NELEM,NPR1,NPR2,NPR3
C
      COMMON /ADDBLK/ IBB1(251),KXTE(251)
C
      COMMON Y(115,22), X(1035), FCMIN(10), OSUM, OPROD, DY,
     1DE, OP, DELTA, ERMAX,
     1CEMAX, JMAX, ... ,SUMDIF,SUM,TIMER,KOLA(4,15),KITIM,KJTIM,KITINV,
     2KJTINV,KNAME(10),KTEMP(10),...,KCASE,KINP,ITEMP(115),JBAS(5,80),
     3ICOLS(250),IJCOL(250),LROW(1035),LNROW(1035),NTROW(16),MSTATU(15),
     4ITEMP(15),ISTAT,IOUT,IROWP,IVIN,IVOUT,IFEEZ,NLELEM,NLETA,NGELEM,
     5NGETA,NCOLD,NUETA,JUSRHS,LSUB,LMAX,LMAXP1,NTOT,NOLO,NCOLO,NROWO,
     6LCORE, JCOLP(10), JEROW(10), ICOLP, ITSINV, ITSIWB,
     6ITCNT, INVER, INVERW,
     7IRTEMP,IDY,JTEMP(14),LS(15),ITIME,MULT,NUETAS(15),ITSITP,ISTRAT
C
      COMMON /ADLOCK/ KITIM
C
      DIMENSION NB(1040),KNAM(250)
      EQUIVALENCE (KNAM(1),Y(451)),(NB(1),LE(1))
C
      CALL MOVLEV(0,IBB1,201,0)
      CALL SECOND(KITIM)
      IROW = 0
      ITIME = 0
      DO 50 I=1,10
   50 ITEMP(I) = 0
      READ(5,70)LIN,IOUT,JUSRHS,KINP,INVERO,INVERW,LPNT,NSTR,MULT,NPP
   70 FORMAT(10I4)
      WRITE(6,79)
   79 FORMAT(3X,*      VERSION 0*,/,6X,*SINGLE PRECISION   *,//)
      WRITE(6,80)INVERO,INVERW
   80 FORMAT(3X,*INVERSION FREQUENCY : TOTAL*,I4,*  SUBPROBLEM*,I4,/)
      WRITE(6,81)NSTR,KMULT,MULT,JUSRHS,LPNT
   81 FORMAT(3X,*STRATEGY*,I4,*  KMULT*,I4,*  MULT*,I4,
     1 2X,*JUSRHS=*,I4,2X,*LPNT=*, I4,/)
      WRITE(6,82)ZTOLZE,ZTOLPV,ZTOLRJ,ZTCOST
   82 FORMAT(3X,4E10.2,//)
      ISTRAT = NSTR
      IF(NSTR .EQ. 0)ISTRAT = -2
      IF(KINP .EQ. 0)KINP = 5
      IF(IOUT .EQ. 0)IOUT = 1
      IF(JUSRHS .EQ. 0)JUSRHS = 1
      IF(MULT .EQ. 0) MULT = 1
      KCASE = 1
      IF(NSTR .EQ. 0)GO TO 1200
      LMAXP1 = LMAX + 1
      ITSIWB=(1)+NETA+2*(NTMAX+...(LMAX+NRMAX)+10)+7291)/7292
```

```
      NTRAC2 = (3*NLMAX + 7299)/7292
      KOLA(3,LMAXP1) = 2 + LMAXP1*NTRAC2
      KOLA(4,LMAXP1) = 2
      DO 100 I=1,LMAX
      KOLA(3,I) = KOLA(3,LMAXP1) + NTRAC1*I
      KOLA(4,I) = KOLA(4,LMAXP1) + NTRAC2*I
  100 CONTINUE
      ITEMP(13) = 0
C
      DO 1000 I=1,LMAXP1
      LSUB = I - 1
      LCORE = LSUB
      IF(LSUB .EQ. 0)LCORE = LMAXP1
      CALL INPUT1
      IPR1 = 1
      IPR2 = NCOL
      KD = KOLA(4,LCORE)
      KXTEM(1)=KTEMP(1)
      DO 105 I1=2,251
      KXTEM(I1) = KJAM(I1-1)
  105 CONTINUE
      CALL WRITMS(3,KXTEM, 251, KD,-1)
      IF(LSUB .EQ. 0)GO TO 200
C
      NTOT = NROW
      NTROW(1) = NTROW(LSUB) + NROW - NROWO
      IF(NSTAT .LE. 1)GO TO 110
      NETAS(LSUB) = INVERW
      GO TO 200
  110 CALL NORMAL
      IF(NSTAT .NE. 3)GO TO 150
      WRITE(6,120)LSUB,SUMINF
  120 FORMAT(//,*  SUBPROBLEM *,I4,*  HAS NO SOLUTION . SUM OF INFEASIBI
     1LITIES  = *,F15.8)
      STOP
  150 NSTATU(LSUB) = 0F
      NETAS(LSUB) = ITSINV
C
  200 KIKA = 1
      KOLA(1,LCORE) = LE(NETA+1)
      KD = KOLA(3,LCORE)
      CALL VECTOR(KIKA, A, IA, NB, KOLA(1,LCORE))
      IF(LSUB .EQ. 0)GO TO 800
      IF(LMAX .EQ. 0)GO TO 1000
      NTROW(1) = NROW
      DO 750 J=1,NROW
      ISTYPE(J) = 0
  750 X(J) = 1.
      GO TO 1000
  800 IF(LSUB .EQ. LMAX)GO TO 1000
  900 NROW = NROWO
      KD = KOLA(4,LCORE)
      CALL READMS(3,KXTEM, 251, KD)
      KTEMP(1)=KXTEM(1)
      DO 950 I1=2,251
      KJAM(I1-1)= KXTEM(I1)
  950 CONTINUE
```

```
       1100 CONTINUE
     C
            KD = KOLA(3,LMAXP1)
            LCORE = LMAXP1
            CALL VECTOR(2 , A, IA, NR, KOLA(1,LMAXP1))
            NTOT = NTROW(LMAXP1)
            NCASE = 1
            LI(1) = LPNT
            ITEMP(14) = 1
            ISTRAT = ISTR
            IF(ISTR .LT. -1 .OR. ISTR .EQ. 2)ISTRAT = 0
            GO TO 1800
       1200 LSUB = 1
            LMAXP1 = 1
            CALL INPUT1
            ITEMP(13) = NPR
            IF(ITEMP(13) .GE. 1)GO TO 1300
            NPR1 = 1
            NPR2 = ICOL
            GO TO 1500
       1300 NPR3 = (ICOL - NROW + NPR - 1)/NPR
            LS(1) = 1
            ITEMP(1) = NROW + NPR3
            DO 1400 J=2,NPR
            LS(J) = ITEMP(J-1) + 1
            ITEMP(J) = ITEMP(J-1) + NPR3
       1400 CONTINUE
            ITEMP(NPR) = ICOL
       1600 NROW = 1
            NTROW(1) = 1
            NTROW(2) = NROW
            NTOT = NROW
            ISTRAT = -1
            KD = 1
            KXTEM(1) = KTEMP(1)
            DO 1700 I=2,251
            KXTEM(I) = KTEMP(I-1)
       1700 CONTINUE
            CALL WRITMS(0,KXTEM, 251, KD,-1)
       1800 CALL SECOND(KOTIMN)
            TIMER = KOTIMN - KITIMN
            WRITE(6,2000)TIMER
       2000 FORMAT(3X,*TOTAL INPUT TIME = *,F8.2,*  SECONDS*,/,1H1)
            CPULT = IELT
            RETURN
            END
```

```
      SUBROUTINE VECTOR(KS, X, IAX, NBX, K1)
C
      IMPLICIT INTEGER (O)
      REAL KITIM,KJTIM,KITINV,KJTINV
C
      COMMON YA(115,32), X(1035), DC1IT(10), DSUM, DPROD, EY,
     1DE, DP, DELTA, SUMAX,
     2CEMAX,KAUX(2011),SUMINF,SUMTIMER,KOLA(4,16),KITIM,KJTIM,KITINV,
     3KJTINV,KMEMC(1)),KTEMP(10),KD,KCASE,KINP,ITEMP(115),JBAS(5,80),
     3ICOLS(25)),ICOL(250),LROW(1035),NTROW(16),MSTATU(15),
     4ITEMP(15),ISTAT,IORD,IDUMP,IVIEW,IVOUT,IFREZ,NLELEM,NLETA,NGELEM,
     5JGETA,JDELTA,JDETA,JUSRHS,LSO),LMAX,LMAXP1,NTOT,NOLO,NCOLO,NROWO,
     6LCORE, JCOL(1)), JFROM(13), ICOLP, ITSINV, ITSINB,
     6ITOUT, INVFRQ), INVFRM,
     7IRTEMP,IJK,ITEMP(14),LS(15),MTIME,MULT,NUETAS(15),ITSITP,ISTRAT
C
      DIMENSION AX((1), IAX(K1),NBX(1340)
C
      GO TO (100,200),KS
  100 CALL RDIT1(), AX, K1, KD,-1)
      KD=KD+1
      CALL RDIT1(), IAX, K1, KD, -1)
      KD=KD+1
      CALL RDIT1(), NBX, 1340, KD, -1)
      GO TO 1000
  200 CALL RDV1(), AX, K1, KD)
      KD=KD+1
      CALL RDV1(), IAX,K1, KD)
      KD=KD+1
      CALL RDV1(), NBX,1340, KD)
 1000 RETURN
      END
```

```
      SUBROUTINE FORMC
C
      IMPLICIT INTEGER (O)
      REAL KITIM,KJTIM,KITINV,KJTINV
C
      COMMON /BLOCK/ZTOLZE,ZTOLPV,ZTOLST,ZTOLRJ,KMULT,KPRICE,KWVEC,KLAST,
     1OBL,OA,OB,OC,OE,OF,OG,OH,OI,OL,OM,ON,OO,OR,OU,NRMAX,NTMAX,NEMAX,NA
     2MAX,NLMAX,IB)IT
C
      COMMON /INDEX/ A(4000),IA(4000),LE(602),LA(250),INBAS(250),
     1IH(115),ISTYPE(115),NROW,NCOL,JRHS,NETA,NELEM,NPR1,NPR2,NPR3
C
      COMMON YA(115,22), X(1135), DCMIN(10), DSUM, DPROD, DY,
     1DE, DP, DELTA, ERMAX,
     1EEMAX,GMAX,OXIO,SUMINF,SUM,TIMES,KOLA(4,15),KITIM,KJTIM,KITINV,
     2KJTINV,KOLME(1)),KTEMP(1)),KD,KCASE,KINP,ITEMP(115),JBAS(5,80),
     3ICOLS(250),ICOL(250),LROW(1035),LROW(1035),NTROW(16),MSTATU(15),
     4ITEMP(15),ISTAT,IOBJ,IPOINP,IVI,IPOST,IFEFZ,NLELEM,NLETA,NBELEM,
     5JBETA,IBEL,IJ,IBETA,JJSAVS,LSU),LMAXP1,NTOT,NOLO,NCOLO,NROWO,
     6LCOPE, JCOLP(10), JERO((10), ICOLO, ITSINV, ITSINB,
     7ITCNT, INFO(1), INVERS,
     7IRTEMP,IHY,ITEMP(16),LS(15),NTINF,NOLT,NUETAS(15),ITSITP,ISTRAT
C
      IF(KCASE .EQ. 2 .AND. LSU4 .GE. 1000 TO 50
      ISTAT = 0F
      KK = 0
      LL = 1
      M = 1
      NN = 1000
      KFORMC = -1
      IF(NCYCLE .EQ. 0)KFORMC = 0
      GO TO 90
   50 M = JBAS(5,1)
      KK = ITRO((LSU3)
      NN = ITRO((LSU3+1) - KK + NROWO
      IF(ITEMP(LSU3) .EQ. OF .OR. MSTAT .EQ. OF)GO TO 2500
      LL = IROW(KK) + 1
      KFORMC = 1
   90 IFEFZ = 1
      DO 10 J = LL,NN
      YA(J,KPRICE) = 0.
  100 CONTINUE
      SUM = 0.
C
      DO 400 J = M,NN
      KK = KK + 1
      IF (LROW(KK)) 200,400,300
C
  200 IF ( ABS(X(KK)) .LE. ZTOLRJ) GO TO 400
      IF(X(KK) .LT. 0.) YA(J,KPRICE) = -1.
      IF(X(KK) .GT. 0.) YA(J,KPRICE) = +1.
      SUM = SUM - ABS(X(KK))
      GO TO 400
C
  300 IF(X(KK) .GT. -ZTOLRJ) GO TO 400
      YA(J,KPRICE) = +1.
      SUM = SUM - X(KK)
```

```
      350 IFFLG = 0
      400 CONTINUE
C
          IF(KP)410,420,450,450,5000
      420 IF(IFFLG .EQ. 0)ISTAT=0I
          IF(IFFLG .EQ. 1)X(IX3J,KPRICE) = 1.
          SUMINF = 3.11
          GO TO 500)
C
      450 IF(IFFLG .EQ. 0)ISTAT=0I
          SUMINF = 3.11
          II = (IX3J) + 1
          DO 10)) I=1,LLMAX
          ITEMP(I) = IF
          KK = (IX3J)(I)
          IN = (IX3J)(I+1) - KK + JROWO
C
          IFFLG = 1
          DO 50) J = 1,II
          YA(J,KWVEC) = 0.
      500 CONTINUE
          SUM = 0.
C
          DO 65) J = KK, IN
          KK = KK + 1
          IF (LX3J)(JX3J) 65),650,600
C
      550 IF ( ABS(X(KK)) .LE. ZTOLRJ) GO TO 650
          IF(X(KK) .LT. 0.) YA(J,KWVEC) = +1.
          IF(X(KK) .GT. 0.) YA(J,KWVEC) = -1.
          SUM = SUM +  ABS(X(KK))
          GO TO 625
C
      600 IF(X(KK) .GT. -ZTOLRJ) GO TO 650
          YA(J,KWVEC) = +1.
          SUM = SUM - X(KK)
      625 IFFLG = 0
      650 CONTINUE
C
          IF(IFFLG .EQ. 1)GO TO 1000
          ISTAT = 0I
          ITEMP(I) = 0I
          SUMINF = SUMINF + SUM
          DO 80) J=1,IIJJ
          IF(IC)LS(JJ(J)) .NE. I)GO TO 800
          LL = LA(JJ(J))
          KK = LA(JJ(J)+1) - 1
          DO 70) K=LL,KK
          IF(IX(K) .LE. IROW0)GO TO 700
          YA(J,IX(K)) = YA(J,KPRICE) - A(K)*YA(IA(K),KWVEC)
      700 CONTINUE
      800 CONTINUE
     1000 CONTINUE
          IF(ISTAT .EQ. 0I) GO TO 5000
          X(IX3J,KPRICE) = 1.
          GO TO 5000
C
```

```
15          2500 DO 3000 I=M1,M1
            3000 A(I,KPRICE) = 0.
            5000 RETURN
                 END
```

```
      SUBROUTINE XBASIS(KRU,KOUT)
C
      IMPLICIT INTEGER (O)
      REAL KITIT1,KJTIM,KITINV,KJTINV
C
      COMMON/BLOCK/ZTOLZE,ZTOLPV,ZTOST,ZTOLRJ,KMULT,KPRICE,KAVEC,KLAST,
     1OBL,OA,OB,OC,OD,OF,OG,OH,OI,OL,OM,ON,OO,OR,OU,JRMAX,NTMAX,NEMAX,NA
     2MAX,NLMAX,IROUT
C
      COMMON/ILIB/ISK(A(4)00),IA(4000),LE(602),LA(250),INBAS(250),
     1IN(115),ISTYPE(115),NROW,NCOL,JRHS,NETA,NFLEM,NPR1,NPR2,NPR3
C
      COMMON YA(115,22), X(1035), DCHIN(10), DSUM, DPROD, DY,
     1DE, DP, DELTA, ERMAX,
     1EMAX,XMAX,ZERO,SUMINF,SUM,TIMER,KOLA(4,15),KITIM,KJTIM,KITINV,
     2KJTINV,KOLE(10),KTEMP(10),KO,KCASE,KINP,ITEMP(115),JBAS(5,80),
     3ICOLS(250),IROWS(250),LROW(1035),LNROW(1035),NTROW(16),MSTATU(15),
     4ITEMP(15),IST,T,IOBJ,IROWP,IVIN,IVOUT,IFREZ,NLELEM,NLETA,NGELEM,
     5JBETA,NELEM,IBETA,JUSRAS,LSUB,LMAX,LMAXP1,NTOT,NOLO,NCOLO,NPOWO,
     6LCORE, JCOLP(1), JFROM(10), ICOLP, ITSINV, ITSIWR,
     6ITOUT, INVFR), INVERW,
     7IRTCMP,INV, JTEMP(14),LS(15),NTIME,MULT,NUETAS(15),ITSITP,ISTRAT
C
      KOUT = 0
      IF(ITSITP .EQ. 0 .OR. LS(3 .EQ. 0)GO TO 9000
C
      LRU = 0
      IF(KRU .EQ. 1 .AND. NUETAS(LSUB) .LT. (2*INVERW)/3)LRU=2
      DO 50)) I=1,ITSINV
      IF(JBAS(1,I) .NE. LSUB)GO TO 1000
      IF(JTEMP(14) .EQ. 0)JTEMP(14) = 1
      INBAS(JBAS(2,I)) = NROW + 1
      JBAS(1,I) = 0
      JBAS(2,I) = 0
C
 100  IF(JBAS(3,I) .NE. LSUB)GO TO 5000
      IF(JTEMP(14) .EQ. 0)JTEMP(14) = 1
      IF(JBAS(4,I) .EQ. 0)GO TO 300)
      IROWP = JBAS(4,I)
      INBAS(IBAS(3,IROWP)) = 0
      INBAS(JBAS(2,I)) = IROWP
      BI(IROWP) = JBAS(5,I)
      IF(LRU .EQ. 1)GO TO 400)
C
      IF(NELEM+NCOLO+(NEMAX-NROWMAX)*AND*NUETAS(LSUB).LE.INVERW)GO TO 1500
      LRU = 1
      GO TO 400)
C
 1500 CALL FTRAN(KOUT,BAS(5,I),KAVEC)
      CALL PTRAN(I,KAVEC)
      CALL UPETA(KAVEC)
      JUETAS(LSUB) = JUETAS(LSUB) + 1
      GO TO 500)
C
 3000 INBAS(IBAS(3,I)) = 0
 4000 JBAS(3,I) = 0
      JBAS(4,I) = 0
```

```
 5000 CONTINUE
C
60          IF(LRU .EQ. KRU)GO TO 9000
            IF(KRU .EQ. 2)GO TO 7500
            IF(NORETAS(LSTD) .GE. (2*INVERN)/3)LRU=1
            KRU = LRU
            GO TO 5000
65          7500 KOUT = -1
C
            9000 RETURN
                 END
```

```
      SUBROUTINE UPVEC(KVEC,KINST)
C
      IMPLICIT INTEGER (Q)
      REAL KITIM,KJTIM,KITINV,KJTINV
C
      COMMON/BLOCK/ZTOLZE,ZTOLPV,ZTOST,ZTOLRJ,KMULT,KPRICE,KWVEC,KLAST,
     1QBL,QA,QB,QC,QE,QF,QG,QH,QI,QL,QM,QN,QO,QR,QU,NRMAX,NTMAX,NEMAX,NA
     2MAX,NLMAX,IRTYT
C
      COMMON/IND/SK(4000),IA(4000),LE(602),LA(250),INBAS(250),
     1JH(115),ISTYPE(115),NROW,NCOL,JRHS,NETA,NELEM,NPR1,NPR2,NPR3
C
      COMMON YA(115,22), X(1035), DCMIN(10), DSUM, DPROD, DY,
     1DE, DP, DELTA, ERMAX,
     1EEMAX,CMAX,COLD,SUMINF,SUM,TIMER,KOLA(4,15),KITIM,KJTIM,KITINV,
     2KJTINV,KJAMIC(10),KTEMP(10),KD,KCASE,KINP,ITEMP(115),JBAS(280),
     3ICOLS(250),ICOL(250),LROW(1035),LNROW(1035),NTROW(16),MSTATU(15),
     4ITEMP(15),ISTAT,IOBJ,IROWP,IVIN,IVOUT,IFREZ,NLELEM,NLETA,NGELEM,
     5IGETA,IMCLCH,IJETA,JUSRHS,LSUB,LMAX,LMAXP1,NTOT,NOLD,NCOLD,NROWD,
     6LCORE, JCOLP(10), JFROW(10), ICOLP, ITSINV, ITSIWB,
     6ITCNT, IJFR2, IJVERM,
     7IRTEMP,IBY,ITEMP(14),LS(15),NTIME,MULT,NUETAS(15),ITSITP,ISTRAT
C
      DIMENSION Y(1035), YTEMP(115)
      EQUIVALENCE (Y(1),YA(1266)),(YTEMP(1),YA(2416))
C
      GO TO (100,200),KINST
C
  100 DO 120 K=1,KMULT
      IF(JFROW(K) .NE. LSUB)GO TO 120
      CALL UNPACK(JCOLP(K),K)
      IF(LS(1) .EQ. 1)GO TO 120
      CALL FTRAN(1,K)
  120 CONTINUE
      GO TO 10000
C
  200 IF(KCASE .EQ. 1)GO TO 10000
      INV = KVEC
      DO 3200 I=1,NTOT
 3200 Y(I) = 0.
      DO 3300 I=1,NROWO
 3300 YA(I,KWVEC) = YA(I,KVEC)
      CALL FTRAN(1,KWVEC)
      DO 4000 I=1,NROWO
      IV = IH(I)
      IF(ICOLS(IV) .EQ. 0 .OR. ABS(YA(I,KWVEC)) .LE. ZTOLZE)GO TO 4000
      LL = LA(IV)
      KK = LA(IV+1) - 1
      DO 3500 J=LL,KK
      IF(IA(J) .GE. NROWO)GO TO 3500
      IR = II + IA(J)
      Y(IR) = Y(IR) - A(J)*YA(I,KWVEC)
 3500 CONTINUE
 4000 CONTINUE
      IF(JFROW(KVEC) .EQ. 0)GO TO 10000
      II = IROW(JFROW(KVEC))
```

```
      KK = ITRO(JF(J)+KVEC) + 1) - N
      DO 42)) I=1,KK
60  4200 Y(N+I) = Y(N+I) + YA((NRO+O+I)*KVEC)
10000 RETURN
      END
```

```
      SUBROUTINE PRICE
C
      IMPLICIT INTEGER (O)
      REAL KITIM,KJTIM,KITINV,KJTINV
C
      COMMON /BLOCK/ZTOLZE,ZTOLPV,ZTCOST,ZTOLRJ,KMULT,KPRICE,KWVEC,KLAST,
     1OBL,OA,OB,OC,OE,OF,OG,OH,OI,OL,OM,ON,OO,OR,OU,NRMAX,NTMAX,NEMAX,NA
     2MAX,NLMAX,NRJIT
C
      COMMON/MDIBK/ A(4000),IA(4000),LE(602),LA(250),INBAS(250),
     1JA(115),ISTYPE(115),NROW,NCOL,JRHS,NETA,NELEM,NPR1,NPR2,NPR3
C
      COMMON YA(115,32), X(1035), DCMIN(10), DSUM, DPROD, DY,
     1DE, DP, DELTA, ERMAX,
     1EEMAX,XMAX,COND,SUMINF,SUM,TIMER,KOLA(4,15),KITIM,KJTIM,KITINV,
     2KJTIN/,KNAME(12),KTEMP(10),KD,KCASE,KINP,ITEMP(115),JBAS(5,80),
     3ICOLS(250),IRCOL(250),LROW(1035),LNROW(1035),NTROW(16),MSTATU(15),
     4ITEMP(15),ISTAT,IOBJ,IROWP,IVIN,IVOUT,IFEEZ,NLELEM,NLETA,NGELEM,
     5JGETA,NELEM,NJETA,JUSRHS,LSUB,LMAX,LMAXP1,NTOT,NOLO,NCOLO,NROWO,
     6LCORE, JCOLP(10), JFROM(10), ICOLP, ITSINV, ITSIW8,
     7ITCNT, INVFRQ, INVFRW,
     7IRTEMP,INV,JTEMP(14),LS(15),NTIME,MULT,NUETAS(15),ITSITP,ISTRAT
C
      DO 1000 J=NPR1,NPR2
      IF(IN)AS(J) .NE. 0)GO TO 1000
      IF(J .LE. IROW .AND. ISTYPE(J) .NE. 1)GO TO 1000
      DSUM = 0.
      LL = LA(J)
      KK = LA(J + 1) - 1
      DO 500 I=LL,KK
      DSUM = DSUM + YA(IA(I),KPRICE)*A(I)
  500 CONTINUE
      IF(DSUM .GE. DELTA)GO TO 1000
C
      IF(MULT .LT. KMULT)GO TO 700
      DCMIN(JMIN) = DSUM
      JCOLP(JMIN) = J
      JFROM(JMIN) = LSUB
  550 DELTA = DSUM
      DO 600 I = 1,KMULT
      IF(DELTA .GE. DCMIN(I))GO TO 600
      DELTA = DCMIN(I)
      JMIN = I
  600 CONTINUE
      GO TO 1000
C
  700 MULT = MULT + 1
      DCMIN(MULT) = DSUM
      JCOLP(MULT) = J
      JFROM(MULT) = LSUB
      IF(MULT .LT. KMULT)GO TO 1000
      JMIN = MULT
      GO TO 550
 1000 CONTINUE
      RETURN
      END
```

```
1                SUBROUTINE CHIZR(KVEC,NTYPE,KENTRY)
           C
                 IMPLICIT INTEGER (Q)
                 REAL KITIM,KJTIM,KITINV,KJTINV
5          C
                 COMMON/BLOCK/ZTOLZE,ZTOLPV,ZTCOST,ZTOLRJ,KMULT,KPRICE,KWVEC,KLAST,
                1QBL,QA,QQ,QC,QE,QF,QG,QH,QI,QL,QM,QN,QO,QR,QU,NRMAX,NTMAX,NEMAX,NA
                2MAX,NLMAX,QP(17)
           C
10               COMMON/IBIG/ A(4000),IA(4000),LE(602),LA(250),INBAS(250),
                1II(115),ISTYPE(115),NROW,NCOL,JRHS,NETA,NELEM,NPR1,NPR2,NPR3
           C
                 COMMON YA(115,22), X(1035), DCMIN(10), DSUM, DPROD, DY,
                1DE, DP, DELTA, ERMAX,
15               1EEMAX,XMAX,CII),SUMINF,SUM,TIMER,KOLA(4,15),KITIM,KJTIM,KITINV,
                2KJTINV,KJVEC(1),KTEMP(10),KD,KCASE,KINP,ITEMP(115),JRAS(5,80),
                3ICOLS(25)),ICOL(250),LROW(1035),LNROW(1035),NTROW(16),MSTATU(15),
                4ITEMP(15),ISTAT,IOBJ,IROWP,IVIN,IVOUT,IFFEZ,NLELEM,NLETA,NGELEM,
                5JGETA,JDELTA,JETA,JUSRHS,LSUB,LMAX,LMAXP1,NTOT,NOLO,NCOLO,NROWO,
20               6LCORE, JCOLP(1)), JFROM(10), ICOLP, ITSINV, ITSIWB,
                7ITCNT, ITJFR), INVERW,
                7IRTEMP,LIY,JTEMP(14),LS(15),NTIME,MULT,NUETAS(15),ITSITP,ISTRAT
           C
                 INDEX = 0
25               LVEC = KVEC
                 IF(KCASE .EQ. 1)LVEC = KVEC
                 IF(KENTRY .EQ. 2)GO TO 1400
                 DRMIN = 1.E25
                 IROWP = 0
30               DRMAX =-1.E25
                 DRART = 1.E25
           C
                 DO 100 I = 1,ITOT
                 INDEX = INDEX + 1
35               IF(INDEX .LE. JRMAX)GO TO 30
                 INDEX = 1
                 LVEC = LVEC + 1
              30 IF( ABS(YA(INDEX,LVEC)) .GT. ZTOLPV)GO TO 100
                 IF( ABS(YA(INDEX,LVEC)) .GT. ZTOLZE)GO TO 1000
40               YA(INDEX,LVEC) = 0.0
                 GO TO 1000
             100 IF(LR)(I) .EQ. 0)GO TO 1000
                 IF(YA(INDEX,LVEC) .LT.-ZTOLPV)GO TO 500
             150 IF(X(I) .LT. -ZTOLRJ)GO TO 1000
45               DTHETA = (X(I) + ZTOLRJ)/YA(INDEX,LVEC)
                 IF(DTHETA .GE. DRMIN)GO TO 1000
                 DRMIN = DTHETA
                 GO TO 900
             500 IF(LR)(I) .EQ. -1)GO TO 800
50               IF(X(I) .GE. 0)GO TO 1000
                 DTHETA = X(I) / YA(INDEX,LVEC)
                 IF (DTHETA .LE. DRMAX) GO TO 1000
                 DRMAX = DTHETA
                 IROWP = I
55               GO TO 1000
             800 IF(X(I) .GT. ZTOLRJ)GO TO 1000
                 DTHETA = (X(I) - ZTOLRJ)/YA(INDEX,LVEC)
```

```
                IF(DTHETA .GE. DRART)GO TO 1000
                DRART = DTHETA
60            900 DY = UC(IN)(KVEC)*DTHETA
                IF(DY .GE. DELTA)GO TO 10000
             1000 CONTINUE
C
                IF(DRMIN .GT. 1.E24.AND. DRART .GT. 1.E24)GO TO 1300
65              IF(DRART .LT. DRMIN)GO TO 1200
                NTYPE = 1
                DE = DRMIN
                GO TO 1400
             1200 DE = DRART
70              NTYPE = 3
                GO TO 1400
             1300 IF(DRMAX .LE. -1.E24)GO TO 9000
                DE = DRMAX
                NTYPE = 2
75              IROWP = IROWP2
                GO TO 1400
C
C       FIND BEST ELIGIBLE PIVOT
C
80           1400 MAXPIV = 0.0
                DRO = DE
                DO 2000 I=1,NT
                INDEX = INDEX + 1
                IF(INDEX .LE. NRMAX)GO TO 1450
85              INDEX = 1
                LVEC = LVEC + 1
             1450 IF(LROW(I) .EQ. 0)GO TO 2000
                IF(YA(INDEX,LVEC) .LE. ZTOLPV)GO TO 1500
                IF(NTYPE .NE. 1)GO TO 2000
90              IF(X(I) .LT. -ZTOLRJ)GO TO 2000
                DTHETA = X(I)/YA(INDEX,LVEC)
                IF(DTHETA .GT. DRO)GO TO 2000
                IF(YA(INDEX,LVEC) .LT. MAXPIV)GO TO 2000
                MAXPIV = YA(INDEX,LVEC)
95              GO TO 1600
             1500 IF(YA(INDEX,LVEC) .GE. -ZTOLPV)GO TO 2000
                IF(NTYPE .NE. 3)GO TO 2000
                IF(LROW(I) .NE. -1)GO TO 2000
                IF(X(I) .GT. ZTOLRJ)GO TO 2000
00              DTHETA = X(I)/YA(INDEX,LVEC)
                IF(DTHETA .GT. DRO)GO TO 2000
                IF(-YA(INDEX,LVEC) .LT. MAXPIV)GO TO 2000
                MAXPIV = - YA(INDEX,LVEC)
             1600 DE = DTHETA
05              IROWP = I
             2000 CONTINUE
C
C       CHECK FOR NEGATIVE RHS ELEMENT
C
10              IF(NTYPE .EQ. 3)GO TO 3000
                IF(X(IROWP) .GE. 0)GO TO 10000
                DE = 0
                X(IROWP) = 0
                GO TO 10000
```

```
 SUBROUTINE CHUZR       73/173  OPT=1                          FTN 4.6+460        12
                                                                                160
15          3000 IF(X(IP3(P) .LT. 0)GO TO 10000
                 DE = 0
                 X(IP3(P) = 0
                 GO TO 10000
            9000 ISTAT = 0
20               WRITE(6,9500)IP3(P),DP(X,DPART,DOMIN(KVEC),KVEC,MULT,KCASE
            9500 FORMAT(4E13.3,3I4)
                C
                 DE = 1.E25
            9000 DY = DSIN(RAND(0)*DE
15          10000 RETURN
                 END
```

```
      SUBROUTINE CHSOL(KINST)
C
      IMPLICIT INTEGER (O)
      REAL KITIM,KJTIM,KITINV,KJTINV
C
      COMMON/BLOCK/ZTOLZE,ZTOLPV,ZTCOST,ZTOLRJ,KMULT,KPRICE,KWVEC,KLAST,
     1OBL,OA,OB,OC,OE,OF,OG,OH,OI,OL,OM,ON,OO,OR,OU,NRMAX,NTMAX,NEMAX,NA
     2MAX,NLVMAX,NROWT
C
      COMMON/INDISK/ A(4000),IA(4000),LE(602),LA(250),INBAS(250),
     1JH(115),ISTYPE(115),NROW,NCOL,JRHS,NETA,NELEM,NPR1,NPR2,NPR3
C
      COMMON YA(115,22), X(1035), DCMIN(10), DSUM, DPROD, DY,
     1DE, DP, DELTA, ERMAX,
     1EEMAX,KMAX,COND,SUMINF,SUM,TIMER,KOLA(4,15),KITIM,KJTIM,KITINV,
     2KJTINV,KINVE(10),KTEMP(10),KD,KCASE,KINP,ITEMP(115),JBAS(5,80),
     3ICOLS(250),ICOL(250),LROW(1035),LNROW(1035),NTROW(16),MSTATU(15),
     4ITEMP(15), ISTAT,IOBJ,IROWP,IVIN,IVOUT,IFFEZ,NLELEM,NLETA,NGELEM,
     5NGETA,NELEM1,NETA,JUSRHS,LSUB,LMAX,LMAXP1,NTOT,NOLO,NCOLO,NROWO,
     6LCORE, JCOLP(10), JFROM(10), ICOLP, ITSINV, ITSIWR,
     6ITCNT, INVFRQ, INVFRW,
     7IRTEMP,INV,JTEMP(14),LS(15),NTIME,MULT,NUFTAS(15),ITSITP,ISTRAT
C
      DIMENSION Y(1035), YTEMP(115)
      EQUIVALENCE (Y(1),YA(1266)),(YTEMP(1),YA(2416))
C
      GO TO (2000,3000,1000,1000),KINST
C
 1000 ERMAX = 0
      IF(LSUB .NE. 0)GO TO 1200
      DO 1100 I=1,NROW
 1100 YA(I,1) = 10.
      GO TO 1250
 1200 CALL UNPACK(JRHS,1)
 1250 DO 1300 I=1,NROW
 1300 YA(I,KPRICE) = YA(I,1)
      CALL FTRAN (1,KPRICE)
      DO 1800 I=1,NROW
      LL = LA(JH(I))
      KK = LA(JH(I)+1) -1
      DO 1500 J=LL,KK
 1500 YA(IA(J),1) = YA(IA(J),1) - A(J)*YA(I,KPRICE)
 1800 CONTINUE
      DO 1900 I=1,NROW
      IF(LR)(I) .EQ. 0)GO TO 1900
      IF( ABS(YA(I,1)) .GT. ERMAX)ERMAX= ABS(YA(I,1))
 1900 CONTINUE
      GO TO 10000
C
 2000 CALL UNPACK(JRHS,KLAST)
      IF(LSUB .GE. 1)GO TO 2800
      DO 2700 J=1,NROW
 2700 Y(J) = YA(J,KLAST)
      GO TO 3000
 2800 II = NROW + 1
      LL = ITR)(LSUB)
      DO 2900 J=II,NROW
```

```
      LL = LL + 1
 2900 Y(LL) = YA(J,KLAST)
 3000 DO 4000 J=1,NROW
      IF(J .GT. NROW0)GO TO 3500
      IF(ICOLS(ITEMP(J)) .NE. LSUB)GO TO 4000
      IF(LSUB .EQ. 1)GO TO 3200
      LL = LA(ITEMP(J))
      KK = LA(ITEMP(J)+1) - 1
      DO 3100 K=LL,KK
 3100 Y(IA(K)) = Y(IA(K)) - A(K)*X(J)
      GO TO 4000
 3200 NN = J
      IVEC = INCOL(ITEMP(J))
      GO TO 3600
 3500 IVEC = JJ(J)
      NN = NTROW(LSUB) + J - NROW0
 3600 LL = LA(IVEC)
      KK = LA(IVEC+1) - 1
      DO 3900 K=LL,KK
      IR = IA(K)
      IF(IR .GT. NROW0) IR = NTROW(LSUB) + IA(K) - NROW0
 3900 Y(IR) = Y(IR) - A(K)*X(NN)
 4000 CONTINUE
      IF(LSUB .EQ. 0)GO TO 10000
      DO 4500 J=1,NROW
      YA(J,KLAST) = 0.
      IF(J .LE. NROW0)GO TO 4500
      IR = NTROW(LSUB) + J - NROW0
      YA(J,KLAST) = Y(IR)
      IF( ABS(Y(IR)) .GT. ERMAX)ERMAX =  ABS(Y(IR))
 4500 CONTINUE
      CALL FTRAN(1,KLAST)
      DO 4800 J=1,NROW
      IF(J .GT. NROW0)GO TO 4600
      YTEMP(J) = YTEMP(J) + YA(J,KLAST)
      GO TO 4800
 4600 IR = NTROW(LSUB) + J - NROW0
      Y(IR) = YA(J,KLAST)
 4800 CONTINUE
      GO TO 10000
C
 5000 DO 5500 J=1,NROW0
      IF( ABS(Y(J)) .GT. ERMAX)ERMAX =  ABS(Y(J))
      YA(J,KLAST) = Y(J) + YTEMP(J)
 5500 CONTINUE
      CALL FTRAN(1,KLAST)
      DO 6500 J=1,NROW0
      Y(J) = YA(J,KLAST)
      IVEC = ICOLS(JJ(J))
      IF(IVEC .EQ. 0)GO TO 6500
      LL = LA(JJ(J))
      KK = LA(JJ(J)+1) - 1
      DO 6000 K=LL,KK
      IF(IA(K) .LE. NROW0)GO TO 6000
      IR = NTROW(IVEC) + IA(K) - NROW0
      Y(IR) = Y(IR) - A(K)*Y(J)
 6000 CONTINUE
```

```
      6500 CONTINUE
      6800 DO 7000 J=1,NTOT
           IF(LPIV(J) .EQ. 0)GO TO 7000
           IF( ABS(Y(J)) .GT. ERMAX)ERMAX =  ABS(Y(J))
           IF( ABS(X(J)) .GT. XMAX)XMAX =  ABS(X(J))
      7000 CONTINUE
           COND = (ERMAX*2**24)/XMAX
           WRITE(6,8000)ERMAX,COND
      8000 FORMAT(20H0MAXIMUM ERROR = ,E14.5/33H0APPROX CONDITION NUMBER
         1OF MATRIX = ,E14.5)
     10000 RETURN
           END
```

```
      SUBROUTINE UPBETA(KVEC)
C
      IMPLICIT INTEGER (Q)
      REAL KITIM,KJTIM,KITINV,KJTINV
C
      COMMON/OLDBK/ZTOLZE,ZTOLPV,ZTCOST,ZTOLRJ,KMULT,KPRICE,KWVEC,KLAST,
     1QBL,QA,QB,QC,QD,QF,QG,QH,QI,QL,QM,QN,QO,QR,QU,NRMAX,NTMAX,NEMAX,NA
     2MAX,NLMAX,IROUT
C
      COMMON/INDBK/ A(4000),IA(4000),LE(602),LA(250),INBAS(250),
     1JH(115),IGTYPE(115),NROW,NCOL,JRHS,NETA,NELEM,NPR1,NPR2,NPR3
C
      COMMON YA(115,22), X(1035), DCMIN(10), DSUM, DPROD, DY,
     1DE, DP, DELTA, ERMAX,
     1EEMAX,GMAX,COND,SUMINF,SUM,TIMER,KOLA(4,15),KITIM,KJTIM,KITINV,
     2KJTINV,KHIDE(10),KTEMP(10),KD,KCASE,KINP,ITEMP(115),JBAS(5,80),
     3ICOLS(250),INCOL(250),LROW(1035),LNROW(1035),NTROW(16),MSTATU(15),
     4ITEMP(15),ISTAT,IOBJ,IROWP,IVIN,IVOUT,IFFEZ,NLELEM,NLETA,NGELEM,
     5NGETA,NUCLMI,NUETA,JUSRHS,LSUB,LMAX,LMAXP1,NTOT,NOLO,NCOLO,NROWO,
     6LCORE, JCOLP(10), JFROM(10), ICOLP, ITSINV, ITSIWR,
     6ITCNT, INVERD, INVERW,
     7IRTEMP,INY,JTEMP(14),LS(15),NTIME,MULT,NUETAS(15),ITSITP,ISTRAT
C
      DIMENSION Y(1035), YTEMP(115)
      EQUIVALENCE (Y(1),YA(1266)),(YTEMP(1),YA(2416))
C
      IF(KCASE .EQ. 1)GO TO 5000
C
      DE = X(IROWP)/Y(IROWP)
      X(IROWP) = DE
      DO 1000 I=1,NTOT
      IF(I .EQ. IROWP)GO TO 1000
      X(I) = X(I) - Y(I)*DE
 1000 CONTINUE
C
      DO 2000 I=1,LMAXP1
      IF(IROWP .GT. NTROW(I))GO TO 2000
      JTEMP(1) = JCOLP(KVEC)
      JTEMP(2) = JFROM(KVEC)
      JTEMP(3) = IROWP
      JTEMP(4) = I - 1
      JTEMP(12) = LNROW(IROWP)
      IF(I .LE. 1)GO TO 1500
      JTEMP(5) = IROWP - NTROW(I-1) + NROWO
      GO TO 9000
 1500 JTEMP(5) = IROWP
      IV = JH(IROWP)
      IF(ICOLS(IV) .EQ. 0)GO TO 9000
      JTEMP(6) = ICOLS(IV)
      JTEMP(7) = INCOL(IV)
      GO TO 9000
 2000 CONTINUE
C
C
 5000 DE = X(IROWP)/YA(IROWP,KVEC)
      X(IROWP) = DE
      DO 6000 I=1,NTOT
```

```
      IF(I .EQ. 2)IP1GO TO 600)
      X(I) = X(I) - YA(I,KVEC)*DE
60    6000 CONTINUE
C
 9000 RETURN
      END
```

```
      SUBROUTINE UPDVRS(KVEC,KOUT)
C
      IMPLICIT INTEGER (I)
      REAL KITIM,KJTIM,KITINV,KJTINV
C
      COMMON /BLOCK/ZTOLEC,ZTOLPV,ZTOLZT,ZTOLRJ,KMULT,KPRICE,KWVEC,KLAST,
     1IBL,OA,IA,IC,IE,IF,IG,IH,IL,IM,IN,IO,IR,IU,NRMAX,NTMAX,NEMAX,NA
     2MAX,NLAMAX,NRHIT
C
      COMMON /1/IDSKA(4000),IA(4000),LE(602),LA(250),INBAS(250),
     1JR(115),IJTYPE(115),NROW,NCOL,JRHS,NETA,NELEM,NPR1,NPR2,NPP3
C
      COMMON YA(115,22), X(1035), DCMIN(10), DSUM, DPROD, DY,
     1DE, DP, DELTA, ERMAX,
     1EERMAX,KIA(5,C,D),SUMINF,SUM,TINER,KOLA(4,15),KITIM,KJTIM,KITINV,
     2KJTINV,KNAME(1),KTEMP(10),KO,KCASE,KINP,ITEMP(115),JBAS(5,80),
     3ICOLS(250),IJCOL(250),LROW(1035),LNROW(1035),NTROW(16),MSTATU(15),
     4ITEMP(15),ISTAT,IOBJ,IROWP,IVIN,IVOUT,IFFEZ,NLELEM,NLETA,NGELEM,
     5JGETA,NELEM1,NJETA,JUSRHS,LSUB,LMAX,LMAXP1,NTOT,NOLO,NCOLO,NROWO,
     6LCORE, JCOLP(1)), JFROM(10), ICOLP, ITSINV, ITSIWB,
     6ITCNT, IINVRW,
     7IRTEMP,IDY,ITEMP(14),L3(15),NTINE,MULT,NUETAS(15),ITSITP,ISTRAT
C
      DIMENSION Y(1035), YTEMP(115)
      EQUIVALENCE (Y(1),YA(1266)),(YTEMP(1),YA(2416))
C
      KOUT = 0
      IF(KCASE .EQ. 1)GO TO 3000
      IRTEMP = 0
      IF(JTEMP(4) .NE. 0)GO TO 1000
C
C     OUTGOING VECTOR IN WORKING BASIS
C
      IV = II(IROWP)
      JTEMP(11) = IV
      IF(IV .GT. NOL)JTEMP(11) = INCOL(IV)
  500 INBAS(IV) = 0
      ICOLS(IV) = 0
      IVIN = JTEMP(1)
      LROW(IROWP) = 1
      LNROW(IROWP) = JCOLP(KVEC)
      IF(JFROM(KVEC) .EQ. 0)GO TO 800
      CALL PACK(KVEC,JFROM(KVEC))
      INBAS(NCOLO) = IROWP
      ICOLS(NCOLO) = JFROM(KVEC)
      INCOL(NCOLO) = IVIN
      II(IROWP) = NCOLO
      ITSINV = ITSINV + 1
  700 IF(ITSINV .GE. INVERW)GO TO 750
      IF(NELEM .GT. (NEMAX-NROW))GO TO 750
      CALL INVERT(KVEC)
      GO TO 3000
  750 DO 755 I=1,NOL
      ITEMP(I) = II(I)
  755 YTEMP(I) = X(I)
      CALL INVERT
      ITSINV = 0
```

```
                DO 780 I=1,IR)/
                IVS = INBAS(ITEMP(I))
                IF(IVS .GT. 0)GO TO 790
                X(IVS) = YTEMP(I)
                LNROW(IVS) = ICOL(JH(IVS))
                LROW(IVS) = 1
                IF(JH(IVS) .LE. NROWO)LROW(IVS)=ISTYPE(JH(IVS))
  780 CONTINUE
                KOUT = 1
                IRETA = -1
                GO TO 5000
  790 KOUT = -1
                DO 795 I=1,IR)/
                IF(INBAS(ITEMP(I)) .NE. 0)GO TO 795
                JBAS(3,ITSIW) = ICOLS(ITEMP(I))
                JBAS(5,ITSIW) = 0
                JBAS(6,ITSIW) = INCOL(ITEMP(I))
                ITSIW = ITSIW + 1
  795 CONTINUE
                GO TO 5000
  800 INBAS(IVIN) = IROWP
                JH(IROWP) = IVIN
                GO TO 700
C
C       OUTGOING (ICT) IN A SUBPROBLEM
C
 1000 IF(NCOLO .GT. NOLO)GO TO 2000
                DO 1050 I=1,NROW
 1050 YA(I,KLAST) = 0.
                DE = 0
                INDEX = 0
                DO 110) I=1,NROW
                IF(ICOLS(JH(I)) .NE. JTEMP(4))GO TO 1100
                LL = LA(JH(I))
                KK = LA(JH(I)+1) - 1
                DO 103) J=LL,KK
                IF(IA(J) .NE. JTEMP(5))GO TO 1080
                YA(I,KLAST) = -A(J)
                IF( ABS(YA(I,KLAST)) .LE. DE)GO TO 1100
                INDEX = INDEX + 1
                DE = YA(I,KLAST)
                JTEMP(3) = I
                GO TO 110)
 1080 CONTINUE
 1100 CONTINUE
                IF(INDEX .GT. 0)GO TO 2000
C
C    EXCHANGE OUTGOING VECTOR WITH VECTOR IN WORKING BASIS
C                  AND WRITE ROW ETA
C
                IROWP = JTEMP(3)
 1150 ITSIW = ITSIW + 1
                IF(NELEM .GT. (MEMAX-NROW))GO TO 1155
                CALL WRETA(KLAST)
 1155 JTEMP(3) = 0
C
C     CHANGE VALUES OF X AND Y DUES TO ROW PERMUTATION
```

```
      C
            IV = JTEMP(3)
            LNROW(IV) = LNROW(IROWP)
            LROW(IV) = 1
            DY = X(IV)
            X(IV) = X(JCOL)
            X(JCOL) = DY
            DY = Y(IV)
            Y(IV) = Y(IROWP)
            Y(IROWP) = DY
            IF(JTEMP(4) .NE. JTEMP(2))GO TO 1200
            IF(IRTEMP .EQ. 0)GO TO 1170
            IF( ABS(YA(JTEMP(5),KVEC)).LT.ZTOLPV)WRITE(6,116)YA(JTEMP(5),KVEC)
        116 FORMAT(* PIVOT ELEMENT  = *,E16.8)
            DY = 1./YA(JTEMP(5),KVEC)
            IV = NEWA
            LNROW(IROWP) = INDEX
            IROWP = 0
            GO TO 1250
      C
      1170 IF( ABS(YA(IROWP,KLAST)) .GE. ZTOLPV)GO TO 1250
            IF( ABS(YA(JTEMP(5),KVEC)) .LE.  ABS(YA(IROWP,KLAST)))GO TO 1200
            IRTEMP = 1
            YA(IROWP,KLAST) = - YA(JTEMP(5),KVEC)
            DY = 1./DE
            DO 1130 I=1,NROW
      1130 YA(I,KLAST) = YA(I,KLAST)*DY
            CALL PACK(KVEC,JFROM(KVEC))
            ITSAV = ITSAV + 1
            CALL PRET(KVEC)
            NEWA = JCOL
            INDEX = LNROW(IROWP)
            LNROW(IROWP) = JTEMP(1)
            JTEMP(3) = IROWP
            GO TO 1130
      C
      C     UPDATE SOLUTION IN WORKING BASIS
      C
      1200 IF( ABS(YA(IROWP,KLAST)) .GT. ZTOLPV)GO TO 1250
            WRITE(6,1245)YA(IROWP,KLAST)
      1245 FORMAT(* LARGEST ELEMENT IN V AND PIVOT ELEMENT  =  *,E16.8)
      1250 DE =-YA(IROWP,KLAST)
            DY = 1./DE
            IV = NEW(IROWP)
            NEWA = IV
      1260 JTEMP(3) = IV
            INBAS(NEWA) = -NEWA
            ICOLS(NEWA) = -NEWA
            JTEMP(10) = ICOL(IV)
            LL = L(IV)
            KK = L(IV+1) - 1
            NROW = JTEMP(4)+1) - JTEMP(JTEMP(4)) + NROW0
            DO 1300 I=1,NROW0
            IF(I .EQ. IROWP)GO TO 1300
            IF(ICOL(NEW(I)) .EQ. JTEMP(4))GO TO 1300
            IF( ABS(YA(I,KLAST)) .LE. ZTOLPV)GO TO 1300
            CALL UPDATE(NEW(I),-DE*I)
```

```
      DP = YA(JTEMP(3),KPRICE)*DY
      YA(JTEMP(3),KPRICE) = DP
      DO 1280 J=LL,KK
      IF(IA(J) .EQ. JTEMP(3))GO TO 1280
      YA(IA(J),KPRICE) = YA(IA(J),KPRICE) - A(J)*DP
 1280 CONTINUE
      JTEMP2 = INCOL(JH(I))
      INBAS(JH(I)) = 0
      ICOLS(JH(I)) = 0
      IANA = ICOLO
      CALL PACK(KPRICE,JTEMP(4))
      INBAS(ICOLO) = I
      ICOLS(ICOLO) = JTEMP(3)
      INCOL(ICOLO) = JTEMP2
      JH(I) = ICOLO
      IF(NCOLO .GT. IANA)GO TO 1300
      DO 1290 M=1,NCOLO
      IF(ICOLS(M) .NE. -IINA)GO TO 1290
      IINA = M
      ICOLS(M) = -1
      INBAS(M) = -1
      LL = LA(M)
      KK = LA(M+1) - 1
      GO TO 1300
 1290 CONTINUE
 1300 CONTINUE
C
C     UPDATE [illegible] VECTORS
C
      DO 1390 I=1,NULT
      IF(JCOLP(I) .EQ. 0)GO TO 1380
      IF(JFROM(I) .NE.JTEMP(4))GO TO 1380
      DP = YA(JTEMP(3),I)*DY
      YA(JTEMP(3),I) = DP
      DO 1350 J=LL,KK
      IF(IA(J) .EQ. JTEMP(3))GO TO 1350
      YA(IA(J),I) = YA(IA(J),I) - A(J)*DP
 1350 CONTINUE
 1380 CONTINUE
C
      IROW = [illegible]
      IF(IRTEMP .EQ. 0)GO TO 1500
      ICOLS(IINA) = 0
      INBAS(IINA) = 0
      IF(ITEMP .NE. INVFR.OR.NELEM .GE.(NEMAX-NRMAX))GO TO 750
      GO TO 5000
C
C       UPDATE [illegible] VECTOR
C
 1500 IV = IINA
      LROW(JTEMP(3)) = 1
      IRTEMP = 1
      GO TO 5000
C
 2000 IRTEMP = 0
      LROW(IROW) = JCOLP(KVEC)
      LROW(JTEMP(3)) = 1
```

```
            IF(JTEMP(4) .EQ. JTEMP(2)) GO TO 3000
30          WRITE(6,20000) JTEMP(4)
        20000 FORMAT(' ***ERROR*** VECTOR FROM ',I4,'  AND NO VECTOR TO REPLACE IT')
            STOP
         3000 DO 3500 I=1,NLT
            IF(JCOLP(I) .EQ. 0)GO TO 3500
35          IF(JFROM(I) .EQ. JTEMP(4))GO TO 3500
            IF(I .EQ. KVEC)GO TO 3500
            OP =YA(JTEMP(5),I)/YA(JTEMP(5),KVEC)
            YA(JTEMP(5),I) = OP
            II = IROW(I) + JTROW(JTEMP(4)+1) - ITROW(JTEMP(4))
40          DO 3200 J=1,II
            IF(J .EQ. JTEMP(5))GO TO 3200
            YA(J,I) = YA(J,I) - YA(J,KVEC)*OP
         3200 CONTINUE
         3500 CONTINUE
45          GO TO 3500
        C
        C   STORE INFORMATION TO UPDATE SUBPROBLEM BASIS
        C
         6000 DO 6200 I=1,LMAX
50       6200 ISTAT(I) = 0
            ITEMP(14) = 1
            IF(IRTEMP .EQ. 2)GO TO 6500
            IBAS(1,ITSIDV) = JTEMP(2)
            IBAS(2,ITSIDV) = JTEMP(1)
55          IF(IRTEMP .EQ. 0)GO TO 6800
         6500 IBAS(3,ITSIDV) = JTEMP(4)
            IBAS(4,ITSIDV) = JTEMP(5)
            IBAS(5,ITSIDV) = JTEMP(1)
            IF(IRTEMP .EQ. 1)IBAS(5,ITSIDV) = JTEMP(10)
60          GO TO 7000
         6800 IBAS(3,ITSIDV) = JTEMP(6)
            IBAS(4,ITSIDV) = 0
            IBAS(5,ITSIDV) = JTEMP(7)
        C
65      C   CHECK IF THERE IS ANOTHER VECTOR TO INTRODUCE INTO BASIS
        C
         7000 JCOLP(KVEC) = 0
            IF(KOUT .NE. 0)GO TO 10000
            DO 7200 I=1,NLT
70          IF(JCOLP(I) .EQ. 0)GO TO 7200
            GO TO 7300
         7200 CONTINUE
            IBETA = -1
            IF(IRTEMP .EQ. 3) IBETA = 0
75       7250 KOUT = 1
            IF(ITSIDV .NE. INVERS)KOUT = -1
            GO TO 10000
        C
         7300 IBETA = 0
80          IF(IRTEMP .EQ. 3)GO TO 7310
            CALL PRICE
            CALL ITRAN
         7310 IV = -1
            DO 7320 I=1,NLT
85          IF(JCOLP(I) .EQ. 0) GO TO 7320
```

```
      IF(JFR)1(I) .EQ. 0)GO TO 7320
      IF(IV .EQ. JFR)W(I))GO TO 7500
      IF(NTROW(I) .NE. 0)GO TO 7350
 7320 IV = IROW(I)
      GO TO 7500
 7350 LSUB = JFR)1(I)
      CALL FORIC
      LSUB = 0
      IV = JFR)1(I)
      M = NROW(IV) + NTROW(IV+1) - NTROW(IV)
 7500 DCMIN(I) = 0.
      DO 7600 I=1,M
 7600 DCMIN(I) = DCMIN(I) + YA(J,I)*YA(J,KPRICE)
      IF(DCMIN(I) .GE. -1.) JCOLP(I) = 0
 7800 CONTINUE
C
      DO 7900 I=1,NVLT
      IF(JCOLP(I) .GE. 1)GO TO 10000
 7900 CONTINUE
      GO TO 7350
C
C   CASE  1  :   UPDATE BASIS AND CHECK IF THERE IS ANOTHER
C                VECTOR TO IMPROVE PRESENT BASIS
C
 8000 JTEMP(2) = JA(IROWP)
      JTEMP(1) = JCOLP(KVEC)
      LNROW(IROWP) = JCOLP(KVEC)
      LROW(IROWP) = 1
      INBAS(JCOLP(KVEC)) = IROWP
      INBAS(JA(IROWP)) = 0
      JA(IROWP) = JCOLP(KVEC)
      IF(NELEM .GT. (NEMAX-NROW))GO TO 8500
      CALL MRETA(KVEC)
      JCOLP(KVEC) = 0
      ICOLP = 0
      IF(NSTAT .EQ. 0)GO TO 8100
      CALL FORIC
 8100 DO 8400 K=1,NVLT
      IF(JCOLP(K) .EQ. 0)GO TO 8400
      CALL FTRAN(3,K)
      IF(NSTAT .EQ. 0)GO TO 8300
      DCMIN(K) = 0.
      DO 8200 J=1,NROW
 8200 DCMIN(K) = DCMIN(K) + YA(J,K)*YA(J,KPRICE)
 8300 IF(NSTAT .EQ. 0)DCMIN(K) = YA(IORJ,K)
      IF(DCMIN(K) .GE. -1.) JCOLP(K) = 0
      IF(JCOLP(K) .EQ. 0)GO TO 8400
      ICOLP = ICOLP + 1
 8400 CONTINUE
C
      IF(ICOLP .GE. 1)GO TO 10000
 8500 JOUT = 1
10000 RETURN
      END
```

```
      SUBROUTINE [illegible]
C
      IMPLICIT INTEGER (I)
      REAL KITIM,KUTIM,KITINV,KUTINV
C
      COMMON/BLOCK/ZTOLZE,ZTOLPV,ZTCOST,ZTOLRJ,KMULT,KPRICE,KWVEC,KLAST,
     1NRL,NR,IR,JC,DI,DF,DG,DH,DI,DL,DM,DN,DO,DP,DU,DPMAX,NTMAX,NEMAX,NA
     2MAX,NL,NN,NPNT
C
      COMMON/INDIGR/ A(4000),IA(4000),LE(602),LA(250),INBAS(250),
     1IH(115),ISTYPE(115),NROW,NCOL,JRHS,NETA,NELEM,NPR1,NPR2,NPR3
C
      COMMON YA(115,22), X(1035), DCMIN(10), DSUM, DPROD, DY,
     1DI, DP, DELTA, ERMAX,
     1EDMAX, LMAX, COLD, SUMINF, SUM, TIMER, KOLA(4,15), KITIM, KUTIM, KITINV,
     2KUTINV, KNAME(10), KTEMP(10), KO, KCASE, KINP, ITEMP(115), JBAS(5,80),
     3ICOLS(250), ICOL(250), LROW(1035), LNROW(1035), NTROW(16), ISTATU(15),
     4ITEMP(15), ISTAT, IOBJ, IROWP, IVIN, IVOUT, IFEEZ, NLELEM, NLETA, NGELEM,
     5NGETA, NDELEM, NDETA, JJSRHS, LSIR, LMAX, LMAXP1, NTOT, NOLO, NCOLO, NROWO,
     5LCOLS, JCOLF(10), JFROM(10), ICOLP, ITSINV, ITSINB,
     6ITCNT, ITMAXR, ITMFRU,
     7IRTEMP, ITM, JTEMP(14), LS(15), NTIME, MULT, NUETAS(15), ITSITP, ISTRAT
C
      DIMENSION Y(115), YTEMP(115)
      EQUIVALENCE (Y(1),YA(1266)),(YTEMP(1),YA(2416))
C
      CALL SECOND(KITIM)
      ITCNT = 0
      ITMA = -1
      IRTEMP = 10
      KVECS = 1
      LSTAT = 0.0L
      LAST = 0
      IF(KCASE .NE. 1)GO TO 450
  300 CALL INVERT
      ITSINV = 0
      CALL ITEROP(0,KVEC,LSTAT)
  400 CALL FORMC
      CALL BTRAN
      MULT = 0
      DELTA = - ZTCOST
      IF(JTEMP(13) .EQ. 0)GO TO 430
      LAST = LAST + 1
      ILAST = LAST
  420 IF(LAST .GT. LMAX)LAST = 1
      NPR1 = LS(LAST)
      NPR2 = JTEMP(LAST)
      CALL PRICE
      IF(MULT .GE. 1)GO TO 440
      LAST = LAST + 1
      IF(LAST .EQ. ILAST)GO TO 1650
      GO TO 420
  430 CALL PRICE
      IF(MULT .EQ. 0)GO TO 1650
  440 CALL INVEC(KVEC+1)
      GO TO 300
  450 DO 500 I=1,37
```

```
          DO 500 I=1,?
          MAT(J,I) = 0
    500   CONTINUE
          ITSIM = 0
C
   1000 CALL INPUT
          ITSIM = 0
          XMAX = 0.0
          ZEMAX = 0.0
          ZRMAX = 0.0
          ITSIM = 0
          DO 1200 I=1,MROWS
          ITEMP(I) = JH(I)
   1200 XTEMP(I) = 0.0
   1220 DO 1240 I=1,LMAXP1
   1240 LS(I) = I - 1
          MULT = 0
          DELTA = -ITCOST
          NN = LMAXP1
          GO TO 1390
C
C     BEGINNING OF MAJOR CYCLE . SELECTION OF SUBPROBLEM .
C
   1250 MULT = 0
          ITMA = ITCNT
          IF(MULTA .EQ. -1)ITMA = ITCNT - 1
          DELTA = -ITCOST
   1260 DO 1280 I=1,LMAX
          IF(ISTATU(I) .EQ. DBL)GO TO 1280
          GO TO 1300
   1280 CONTINUE
          IF(IRTCHP .GE. 5)GO TO 1600
          IRTCHP = 10
          LAST = 0
          GO TO 1350
   1300 LAST = LAST + 1
          IF(LAST .LE. LMAX)GO TO 1320
          LAST = 1
   1320 IF(ISTATU(LAST) .EQ. DBL)GO TO 1390
          IF(ITMA .EQ. ITCNT)GO TO 1350
          NN = 2
          LS(1) = 0
          LS(2) = LAST
          ITMA = ITCNT
          GO TO 1390
   1350 NN = 1
          LS(1) = LAST
C
C     BEGINNING A SIMPLEX MAJOR ITERATION
C
   1390 DO 1500 I=1,NN
          LSUB = LS(I)
          CALL CHNGSC(LSUB)
          CALL CHNSCS(?,NOPT)
          IF(NOPT .EQ. -1)GO TO 1800
          CALL ?RPIC
          CALL ?R?
```

```
      CALL PRIOR
      IF(MULT .GE. 1)GO TO 1420
      IF(LSUB .GE. 1)MSTATU(LSUB) = DBL
      IF(ITSINV .EQ. 0)GO TO 1450
      IF(I .NE. II)GO TO 1500
      IF(ISTRAT)3500,1260,1500
 1420 CALL UPVEC(KVEC,1)
      IF(ITSINV .NE. 0)GO TO 1500
 1450 CALL DISOL(1)
 1500 CONTINUE
C
      LSUB = 0
      CALL CHANGE(LSUB)
      IF(ITSINV .NE. 0)GO TO 1600
      CALL DISOL(2)
      CALL ITERO(0,KVEC,LSTAT)
      IF(ITSITP .EQ. 0)GO TO 1600
      DO 1550 I=1,ITSITP
      IV = IBAS(3,I)
 1550 MSTATU(IV) = DF
      ITSINV = ITSITP
      IF(MULT .EQ. 0)GO TO 1260
C
 1600 IF(MULT .GE. 1)GO TO 3000
 1650 IF(MSTAT .EQ. DI)GO TO 2000
      MSTAT = DBL
      GO TO 5000
 2000 MSTAT = DI
      GO TO 5000
C
 3000 LSTAT = DI
      DO 5000 I=1,MULT
      DELTA = 1.0
C
      DO 4500 K=1,MULT
      IF(JCOL2(K) .EQ. 0)GO TO 4500
      CALL UPVEC(K,2)
      CALL OTIZR(K,ITYPE,1)
      IF(MSTAT .EQ. DU)GO TO 6000
      IF(DELTA .LE. DY)GO TO 4500
      DELTA = DY
      DTEMP = DZ
      DTYPE = ITYPE
      DROUP = IGROUP
      KVPOS = K
 4500 CONTINUE
C
      IF(DELTA .GE. 1.0)GO TO 5200
      IF(KVPOS .EQ. INY .OR. KCASE .EQ. 1)GO TO 4550
      CALL UPVEC(KVPOS,2)
 4550 IF(DTYPE .NE. 2)GO TO 4600
      IGROUP = DROUP
      GO TO 4650
 4600 DZ = DTEMP
      CALL OTIZR(KVPOS,DTYPE,2)
 4650 CALL UPBET(KVPOS)
      ITCNT = ITCNT + 1
```

```
      ITSINV = ITSINV + 1
      CALL [illegible](IVECS,KOUT)
      CALL [illegible](1,IVECS,LSTAT)
      IF(KOUT)[illegible],[illegible],[illegible]
 4800 LSTAT = [illegible]
 5000 CONTINUE
C
 [illegible] IF(IST[illegible])[illegible],[illegible],[illegible]
 [illegible] IF([illegible] .[illegible]. ([illegible] -[illegible]) .[illegible]. ITSINV .GE. INVFRW)GO TO 3[illegible]
      IF(IST[illegible] .[illegible]. -2 .OR. [illegible]STAT .NE. OF)GO TO 400
C
 6000 CALL [illegible](1,IVECS,LSTAT)
      RETURN
      END
```

```
      VERSION 2
   STABLE PRECISION
```

```
 INVERSION FREQUENCY = TOTAL  20  SUBPROBLEM  10
 STRATEGY   0  KMULT  10  NULT   3  ITSRHS=   1  LPNT= 800
    .10E-07   .10E-05   .10E-04   .10E-04

 NAME          TESIS
 ROWS
 COLUMNS
 RHS
 ENDATA


PROBLEM STATISTICS

   3 ROWS

   2 STRUCTURAL COLUMNS

   5 NON-ZERO ELEMENTS

DENSITY =  .83333

 INPUT TIME SUBPROBLEM TESIS      =      .05 SECONDS

 NAME          SUB1
 ROWS
 COLUMNS
 RHS
 ENDATA


PROBLEM STATISTICS

   4 ROWS

   3 STRUCTURAL COLUMNS

  10 NON-ZERO ELEMENTS

DENSITY =  .83333

 INPUT TIME SUBPROBLEM SUB1       =      .06 SECONDS


 INVERSION SUBPROBLEM    1          OLD :    0 ETAS :    0 ELEMENTS
 MAXIMUM ROW ERROR =    0.          NEW :    0 ETAS :    0 ELEMENTS      0 STRUCTURAL COLUMNS      4 NON-ZEROES


ITCOUNT  STATUS  OBJ.VALUE      VECIN FROM  VECOUT FROM  IN      DJ          NETA  NELEM  TIME  DLA  DIA  CASE

   1     F        -3.000          5           4          1     -1.000          3     .02    0    0    0
```

2	F	-24.000	7	1	5	1	1	-2.000	2	7	.03	0	0	0
2		-24.000	9	1	0	1	1	-2.000	2	7	.03	0	0	0

177

NAME SUB2
ROWS
COLUMNS
RHS
ENDATA

PROBLEM STATISTICS

5 ROWS

4 STRUCTURAL COLUMNS

15 NON-ZERO ELEMENTS

DENSITY = .75000

INPUT TIME SUBPROBLEM 3132 = .07 SECONDS

INVERSION SUBPROBLEM 2 OLD : 0 ETAS ; 0 ELEMENTS
MAXIMUM ROW ERROR = 0. NEW : 0 ETAS ; 0 ELEMENTS 0 STRUCTURAL COLUMNS 5 NON-ZEROES

ITCOUNT	STATUS	OBJ.VALUE	VECTI FROM	VECOUT FROM	TN	DJ	NETA	NELEM	TIME	DLA	DTA	CASE
1 (1)	I	0.000	7	2	5	2	2	-2.000	1	5	.02	0	0	0
2 (1)	F	-3.000	9	2	4	2	2	-1.000	2	9	.03	0	0	0
3 (1)	F	-24.000	1	2	7	2	2	-4.000	3	14	.04	0	0	0
3 (1)		-24.000	9	2	0	2	2	-4.000	3	14	.05	0	0	0

TOTAL INPUT TIME = .36 SECONDS

WITH COMPLETE INVERSION

INVERSION SUBPROBLEM 1 OLD : 2 ETAS ; 7 ELEMENTS
MAXIMUM ROW ERROR = 0. NEW : 1 ETAS ; 4 ELEMENTS 1 STRUCTURAL COLUMNS 7 NON-ZEROES

INVERSION SUBPROBLEM 2 OLD : 3 ETAS ; 14 ELEMENTS
MAXIMUM ROW ERROR = 0. NEW : 2 ETAS ; 7 ELEMENTS 2 STRUCTURAL COLUMNS 10 NON-ZEROES

INVERSION SUBPROBLEM 0 OLD : 0 ETAS ; 0 ELEMENTS
MAXIMUM ROW ERROR = 0. NEW : 0 ETAS ; 0 ELEMENTS 0 STRUCTURAL COLUMNS 3 NON-ZEROES

MAXIMUM ROW ERROR = 0.

APPROX CONDITION NUMBER OF BASIS = 0.

ITCOUNT	STATUS	OBJ.VALUE	VCOL1 FROM	VCOL2 FROM	IN	DJ	NETA	NELEM	TIME	OLA	DIA	CASE
1	N	55.000	0	1	7	1	1	-5.000	0	0	.15	0	0	4
2		3.000	3	2	2	0	0	-6.000	1	2	.16	1	3	1
3		3.000	2	1	5	0	0	-2.000	2	4	.22	2	6	1
4	N	-37.000	2	4	6	2	2	-3.000	4	8	.28	2	6	2
4		-37.000	0	0	0	0	0	-3.000	4	8	.36	2	6	11

0

179 PROBLEM TESIS

OBJ(I)	VALUE	ROW NAMES	PI(I)	RHS					
OBJ	37.000	R1	1.000	0.000	1	1	1	11	1
X2	1.667	R2	.500	22.000	5	2	2	11	1
X3	1.333	R3	.500	10.000	5	3	3	11	1
X4	5.667	R4	.500	8.000	6	4	4	11	2
X8	9.000	R5	1.500	6.000	8	5	4	11	2
X9	4.000	R6	2.000	4.000	9	6	5	11	2

TOTAL INPUT-OUTPUT TIME = .71 SECONDS