To the Graduate Council :

I am submitting herewith a thesis written by R. P. Sundarraj entitled "Documentation of DECOMP : A Dantzig-Wolfe Decomposition Code for Linear Programming." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Management Science.

J. K. Ho, Major Professor

We have read this thesis
and recommend its acceptance :

Accepted for the council :

Vice Provost
and Dean of the Graduate School

DOCUMENTATION OF DECOMP : A DANTZIG-WOLFE DECOMPOSITION CODE FOR LINEAR PROGRAMMING

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

R. P. Sundarraj

June 1988

ACKNOWLEDGEMENTS

I would like to express my deep gratitude and appreciation to Dr. James K. Ho, my thesis director, for his expert help, encouragement and patience in the completion of this thesis. A special thanks to Dr. James K. Ho for helping me learn a lot about scientific writing. I would also like to thank each of the rest of my thesis committee : Dr. Kenneth C. Gilbert and Dr. Charles P. Pfleeger. I would especially like to thank them for giving me quick feedback on my thesis.

A special word of thanks to my friend and colleague Mr. S. Kingsley Gnanendran for the numerous invaluable discussions that we had during the initial stages of this thesis. Finally I would like to thank Mr. Ramakanth Ramapalli for helping me format the thesis.

ABSTRACT

This thesis is a documentation of DECOMP, an implementation of the Dantzig-Wolfe decomposition algorithm for solving block angular linear programming problems based on the scheme of Ho and Loute (1981).

We first present the specifications that must be met by any robust implementation of linear programming decomposition. A detailed explanation of the sub-routines of an IBM/370 version of DECOMP then follows with an objective to aid research on computational aspects of large scale linear programming.

TABLE OF CONTENTS

CHAPTER		PAGE
1.	INTRODUCTION	1
2.	SPECIFICATIONS FOR A ROBUST CODE	3
3.	PROGRAM SUBROUTINES	46
4.	PORTABILITY ISSUES	252
BIBLIOGRAPHY		257
APPENDICES		260
	APPENDIX A : USERS GUIDE	261
	APPENDIX B : VARIABLE DICTIONARY	271
VITA		283

LIST OF FIGURES

FIGURE		PAGE
2.1	Pictorial representation of a block angular linear program	32
2.2	Tabular representation of sub-problem data	38
2.3	Sub-problem basis	38
2.4	Sub-problem basis inverse	38
3.1	Organization of the direct access file	118
3.2	Structure of a general basis before inversion	148
3.3	Structure of the basis after step 1.1	148
3.4	An example basis	150
3.5	Structure of the example basis after step 1.1	150
3.6	Structure of the basis after step 1.2	150
3.7	Structure of the example basis after step 1.2	150
3.8	Structure of the basis after step 1.3	152
3.9	Structure of the example basis after step 1.3	152
3.10	Structure of the basis after step 1	154
3.11	Illustration of step 2	154
3.12	Structure of the basis after step 2	154
3.13	L-U factors of the basis	154
3.14	Illustration of merit count	156
3.15	A step of Gaussian elimination	156
3.16	Data array before and after purging	192
3.17	Master problem name's data type	239
4.1	Organization of the data array	255

CHAPTER 1

Introduction

DECOMP is a FORTRAN code of the Dantzig-Wolfe (D-W) decomposition algorithm for solving block-angular linear programs. Originally coded by C. Winkler at the Systems Optimization Laboratory (SOL) at Stanford University, DECOMP was built around LPM1 which is J. Tomlin's all-in-core implementation of the Revised Simplex Method. Since then Ho and his European collaborators, notably Etienne Loute at the Center for Operations Research and Econometrics in Belgium, had expanded and improved upon the code as well as adapting it to run on various machines, including CDC's Cyber series and DATA General's MV8000. It was the prototype for subsequent commercial-software -based implementations that provided state-of-the-art benchmark results in LP decomposition (Ho and Loute (1981)). DECOMP can solve problems with up to 4000 rows, 10,000 columns and 60,000 non-zero elements (Ho and Loute (1984)) and is very useful for research on computational aspects of large scale linear programming. Also, it has proven to be robust and relatively portable.

With the advent of parallel processors and supercomputers, the implementation of the D-W decomposition algorithm has to be re-examined to exploit new computing architecture for improved performance. For this purpose a robust serial implementation of the code is necessary due to two main reasons :

1. The serial code can be used to measure the speed-up achieved using parallel computation.

2. Most of the features of the serial code can be used in the parallel environment.

Since robust experimental codes for large-scale optimization are non-trivial to assemble, DECOMP is chosen as the benchmark serial program.

While this code went through many stages of development, there has never been a comprehensive documentation. This thesis presents a detailed documentation of DECOMP and serves as a programmer's guide to both the design and coding of DECOMP. Chapter 2 reviews the D-W decomposition principle as well as specifications for an efficient implementation. A pseudo-code for the algorithm is also presented. Chapter 3 describes details of the subroutines in DECOMP. Chapter 4 examines parts of the code that are machine dependent and gives examples of necessary changes for installation on different computers.

CHAPTER 2

Specifications For A Robust Code

2.1. Revised Simplex Method:

For the sake of convenience and completeness the revised simplex method (RSM) is stated briefly. Consider the linear program (LP)

$$\text{Min} \quad \mathbf{c'x'} \tag{2.1}$$

ST

(LP1)

$$\mathbf{A'x'} = \mathbf{b'} \tag{2.2}$$

$$\mathbf{x'} \geq \mathbf{0} \tag{2.3}$$

where $\mathbf{A'}$ is m by n, $\mathbf{c'}$ is 1 by n and $\mathbf{b'}$ is m by 1. Let

$\mathbf{c'_b}$ be the vector formed from the elements of $\mathbf{c'}$ corresponding to the basic variables,

$\mathbf{c1'_b}$ be the basic cost vector in phase 1,

c'_j be the j^{th} element of the vector $\mathbf{c'}$,

$c1'_j$ be the j^{th} element of the phase 1 objective vector,

$\mathbf{a'_j}$ be the j^{th} column of the matrix $\mathbf{A'}$,

$\mathbf{B'}$ be the basic matrix (or sometimes simply called basis),

$\mathbf{d'_j}$ be the vector obtained by updating $\mathbf{a'_j}$ with the basis $\mathbf{B'}$,

π' be the dual vector in phase 2,

$\mathbf{B}'$ be the basic matrix (or sometimes simply called basis),

$\mathbf{d}'_j$ be the vector obtained by updating $\mathbf{a}'_j$ with the basis $\mathbf{B}'$,

π' be the dual vector in phase 2,

$\pi\mathbf{1}'$ be the dual vector in phase 1 and

$\mathbf{x}'^*$ be the solution vector.

An LP in which the constraints (given by 2.2)) are equalities is said to be in standard form. Any LP can be brought to standard form by introducing a slack variables, surplus variables and artificial variables depending on the type of the constraint (less than or greater than or equality). These terms although very intuitive are cumbersome when it comes to explaning the implementation. Hence all variables that are introduced to bring the problem to standard form, will, in future be referred to **logical variables** or sometimes simply as **logicals**. If it is necessary to refer to the type of the logical, the type of the row is prefixed to the word "logical". For example, an artificial variable may be referred to as an equality logical and so on. The revised simplex algorithm can be stated in the following steps

1. Form the basic cost vector. In phase 2 it is $\mathbf{c}'_b$ and in phase 1 it is $\mathbf{c1}'_b$.
2. Compute the dual vector. Solve $\pi'\mathbf{B}' = \mathbf{c}'_b$ for π' in phase 2 and $\pi\mathbf{1}'\mathbf{B}' = \mathbf{c1}'_B$ for $\pi\mathbf{1}'$ in phase 1.
3. Compute the reduced cost for all columns j that are non-basic. In phase 1 the reduced cost, $c1'_j - \pi\mathbf{1}'\mathbf{a}'_j$ is $-\pi\mathbf{1}'\mathbf{a}'_j$ since $c1'_j$ is zero if the j^{th} column is non-basic. In phase 2 the reduced cost is $c'_j - \pi'\mathbf{a}'_j$.
4. Find the entering column based on the reduced cost. If there is no entering column then declare the problem optimal if in phase 2 or infeasible if in phase 1 and STOP.
5. Solve $\mathbf{B}'\mathbf{d}'_j = \mathbf{a}'_j$ to update the entering column $\mathbf{a}'_j$ to $\mathbf{d}'_j$.
6. Find the pivot row using ratio test. If none then declare the problem unbounded

and STOP.

7. Go to step 1.

2.2. Specifications for a Robust RSM Implementation :

The following techniques are used for an efficient implementation of RSM.

2.2.1. Column Wise Packing :

In order to save on memory space only non-zero elements of the data matrix are stored. The data structure consists of three arrays as described below.

(a) The first array A is stores the non-zero elements of the data matrix.

(b) The second array IA gives the row in the data matrix to which each element in A belongs.

(c) The third array LA gives the starting position in A, of the first non-zero element in each column of the data matrix.

Therefore, the number of non-zero elements in the i^{th} column is given by LA(i+1) - LA(i). Whenever a column of the data matrix is needed during the course of the simplex method, it has to be reconfigured using the information of the above arrays. This is frequently known as unpacking and is done as follows. Suppose the i^{th} column is to be unpacked into the array Y.

1. Find the number of non-zero elements in the i^{th} column.
2. Start with the element of A which is the first non-zero element of the i^{th} column of the data matrix. Use IA to assign the element to the appropriate component of the array Y and repeat this procedure for all non-zero elements of the i^{th} column.

For example, consider the data matrix

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 3 & 6 \\ 2 & 5 & 0 \end{bmatrix}$$

A = 1, 2, 3, 5, 6

IA = 1, 3, 2, 3, 2

LA = 1, 3, 5, 6

To unpack the 1st column consisting of 2 (= LA(2) - LA(1)) elements, Y is first initialized so that it becomes the zero vector. Then A(1) is assigned to Y(1) since IA(1) is 1 and A(2) is assigned to Y(3) since IA(2) is 3.

2.2.2 Factorization of Basis :

Consider an n by n basis matrix **B'**. Suppose $\mathbf{B'}_1$ is obtained from **B'** by replacing its r^{th} column with a column vector **s'**. Then

$$\mathbf{B'}_1 = \mathbf{B'} + (\mathbf{s'} - \mathbf{B'u_r})\mathbf{u_r}^t$$

where $\mathbf{u_r}$ is the r^{th} unit column vector. Therefore

$$B'_1 = B' [I + B'^{-1} (s' - B'u_r) u_r^t]$$

$$= B' [u_1,...,u_{r-1}, t', u_{r+1},...u_n]$$

$$= B' F'_1$$

where

$$t' = B'^{-1}s'. \tag{2.6}$$

If the operation was carried out k times then

$$B'_k = B'_o F'_1 ... F'_k \tag{2.7}$$

Due to factorization of the basis steps 2 and 5 of the RSM are executed as follows.

Step 2 : To solve for π' at the k^{th} iteration of RSM we have

$$\pi' B'_k = c'_b$$

which after using (2.7) becomes

$$\pi' = (((c'_b E'_k) E'_{k-1})...) E'_1 \tag{2.8}$$

where

$$E'_i = F'^{-1}_i \quad \text{for all } i = 1,...,k$$

Since F' is an elementary matrix, E' is also an elementary matrix. Therefore the solution for π' at the k^{th} step involves k inner products. Each of the elementary matrix

Ei' (i = 1 to k) is known as eta matrix. The non-trivial column of an eta matrix is called an eta vector and the set of all eta vectors generated is commonly called eta file.

Step 5 : To solve for $\mathbf{d'_j}$ at the k^{th} iteration of RSM we have

$$\mathbf{B'_k d'_j} = \mathbf{a'_j}$$

which after using (2.7) becomes

$$\mathbf{d'_j} = \mathbf{E'_1}\,(\ldots(\,\mathbf{E'_{k-1}}\,(\,\mathbf{E'_k a'_j}\,))) \qquad (2.9)$$

2.2.3. Augmented Basis:

The cost row is treated as a non-binding row in the original set of constraints, thereby changing the size of the basis to (m+1) by (m+1). Therefore the LP in (2.1)-(2.3) is written as shown below

$$\text{Min} \quad s_o \qquad (2.10)$$

ST

$$s_o + \mathbf{c'x'} = 0 \quad \text{(Non-binding row)} \qquad (2.11)$$

(LP2)

$$\mathbf{A'x'} = \mathbf{b'} \qquad (2.12)$$

$$\mathbf{x'} \geq \mathbf{0} \qquad (2.13)$$

$$s_o \geq 0 \qquad (2.14)$$

Note that, the j^{th} column of LP1 would correspond to the $(j+1)^{st}$ column of LP2. Since

(2.11) is declared as non-binding, s_o starts off as a basic variable and never leaves the basis. Let

c be the vector [1,**c**'],

x be the vector [s_o,**x**'],

A be the matrix $\begin{matrix} s_o & \mathbf{c}' \\ \mathbf{0} & \mathbf{A}' \end{matrix}$,

$\mathbf{c_b}$ be the basic cost vector of the augmented system,

c_j be the j^{th} element of **c**,

$\mathbf{a_j}$ be the j^{th} column of **A**,

B be the basic matrix of the augmented system,

$\mathbf{d_j}$ be the column obtained by updating $\mathbf{a_j}$ with the basis **B**,

π be the dual vector of the augmented system,

$\mathbf{x^*}$ be the solution vector of the augmented system.

The affected steps of RSM should now be stated in terms of LP2 as follows.

Step 1 : In phase 2 the basic cost vector

$$\mathbf{c_b} = (1, \mathbf{0}) \tag{2.15}$$

In phase 1 the basic cost vector

$$\mathbf{c_b} = \begin{cases} \{1 \text{ in the } i^{th} \text{ position if the } i^{th} \text{ equality logical is} < 0, \} \\ \{-1 \text{ in the } i^{th} \text{ position if the } i^{th} \text{ equality logical is} > 0, \} \\ \{1 \text{ in the } i^{th} \text{ position if the } i^{th} \text{ inequality logical is} < 0 \} \\ \{ 0\text{'s elsewhere.} \} \end{cases} \tag{2.16}$$

This scheme is motivated by the following advantages

(a) Changing of the objective row can be easily done by declaring the corresponding row as non-binding.

(b) Forming the cost vector in phase 2 is easily done by (2.13) and therefore transition from phase 1 to phase 2 is easy.

Step 2 : In the augmented system the new basis and its inverse are given by

$$\mathbf{B} = \begin{bmatrix} 1 & \mathbf{c}'_b \\ \mathbf{0} & \mathbf{B}' \end{bmatrix} \tag{2.17}$$

$$\mathbf{B}^{-1} = \begin{bmatrix} 1 & -\pi' \\ \mathbf{0} & \mathbf{B}'^{-1} \end{bmatrix} \tag{2.18}$$

In phase 1

$$\pi = [\,0\,, -\mathbf{c1}'_b\,]\,\mathbf{B}^{-1} = [\,0, -\pi\mathbf{1}'] \tag{2.19}$$

In phase 2

$$\pi = [\,1, \mathbf{0}\,]\,\mathbf{B}^{-1} = [\,1, -\pi'] \tag{2.20}$$

Hence, in both phase 1 and phase 2 the dual vector of the original problem (LP1) is, modulo a minus sign, the second component of the dual vector of the augmented scheme.

Step 3 : A column $\mathbf{a}_{j+1}$ of the augmented system is given by

$$\mathbf{a}_{j+1} = [\, c'_j, \mathbf{a}'_j]^t \tag{2.21}$$

The j^{th} column of LP1 corresponds to the $(j+1)^{st}$ column of LP2. Therefore, the reduced cost of the j^{th} column of LP1 is calculated by pre-multiplying the dual vector of the augmented system to $\mathbf{a}_{j+1}$. In phase 1

$$\pi \mathbf{a}_{j+1} = [\, 0, -\pi \mathbf{1}'] \; [\, c'_j, \mathbf{a}'_j \,]^t = -\pi \mathbf{1}' \mathbf{a}'_j \tag{2.22}$$

which is the reduced cost in phase 1. In phase 2

$$\pi \mathbf{a}_{j+1} = [\, 1, -\pi' \,] \; [\, c'_j, \mathbf{a}'_j \,]^t = c'_j - \pi' \mathbf{a}'_j \tag{2.23}$$

which is the reduced cost in phase 2.

Step 5 : The column update of the augmented system is

$$\mathbf{d}_{j+1} = \begin{bmatrix} 1 & -\pi' \\ 0 & \mathbf{B}'^{-1} \end{bmatrix} \begin{bmatrix} c'j \\ \mathbf{a}'_j \end{bmatrix} = \begin{bmatrix} c'_j - \pi' \mathbf{a}'_j \\ \mathbf{B}'^{-1} \mathbf{a}'_j \end{bmatrix} \tag{2.24}$$

The second term of $\mathbf{d}_{j+1}$ gives the required column update.

2.2.4. Multiple Pricing :

In the RSM, the reduced cost of all the non-basic columns is found and the one with the "best" reduced cost is normally choosen as the entering column. Conceivably the dual prices at a certain iteration would not greatly differ from those of the previous iteration, and therefore a number of columns which priced out "good" in the previous iteration would still be "good". In order to to extract every bit out of a complete pricing done in a particular iteration, multiple pricing is done and it consists of the following steps.

(a) Instead of considering only the column with the "best" reduced cost, a pool of candidate columns (sometimes called potential entering columns) with good reduced cost is selected.

(b) All these columns in the pool are updated for the current basis.

(c) Among those columns which can bring about a "good" change in the objective value, the column which provides the "best" change would enter the basis.

(d) The basis is updated, the etas are recorded and a signal is given to indicate that the column which just entered the basis is not an eligible entering column.

(e) To verify the possibility of having an entering column among the remaining set of candidate columns, the new reduced cost of a column is required and is found as follows. The latest update is obtained by pre-multiplying the last eta to the column updated before the last basis change. The reduced cost of the column is found by pre-multiplying the appropriate cost vector to the updated column.

(f) Steps c through f are repeated until all columns in the pool have either entered the basis or do not have a "good" reduced cost.

2.3. Pseudo-code for RSM Implementation :

The pseudo-code of the part of DECOMP that executes the simplex algorithm follows. For the purpose of easy identification later, procedures with the same names as in DECOMP are italicized.

Program SIMPLEX ;

Procedure *INVERT* ;

- Does the L-U factorization of the basis and also updates the solution.

Procedure *FORMC* ;

- Forms the basic cost vector using appropriate objectives for phase 1 and phase 2.

Procedure *BTRAN* ;

- Computes the dual price vector.

Procedure *PRICE* ;

- Computes the reduced costs of non-basic columns.
- Selects a pool of candidate columns with "good" reduced costs.

Procedure *FTRAN* ;

- Updates a column to the current basis.
- Also computes the reduced cost if called with a suitable flag during a multiple pricing pass.

Procedure *UNPACK* ;

- Unpacks a column stored under standard column-wise packing.

Procedure *CHUZR* ;

- If called with suitable flags
    - Finds for a given column, the maximum pivot among a set of pivots that provide the same best change in the objective value.

-Else

- Finds the pivot row for a given entering column. Also identifies if the given column is unbounded.

Procedure *UPBETA* ;

- Updates the solution after a basis change.

Procedure *WRETA* ;

- Writes to the eta file the eta just created due to the basis change.

Begin /* Driver Routine */

- Read the LP ;

-INVERT ;

-FORMC ;

-BTRAN ;

-PRICE ;

While there are entering columns

-*UNPACK* all candidate columns;

For each entering column

/* Refer to multiple pricing section */

For each entering column

If the column has not yet entered

- *FTRAN* ;
- If the reduced cost is "good" *CHUZR* ;
- Update the "best" change in the objective.

End If

End For

- If necessary call *CHUZR* with flag ;
- *UPBETA* ;
- Update the basis.
-*WRETA* ;

End For

- *INVERT* if the number of etas is large or if the inversion frequency has been reached.
- *FORMC* ;
- *BTRAN* ;

- *PRICE* ;

End While

End

Note that a few changes are necessary to implement the RSM in the Dantzig-Wolfe decomposition algorithm discussed in sections 2.5 through 2.7. For details of these changes refer to subroutine NORMAL in chapter 3.

2.4. Example of RSM Implementation :

Consider the LP

Min - 2x1 - 3x2 - 4x3

ST

x1 + x2 + x3	≤	50
2x1 + x2 + 3x3	≥	60
x1 - x2 + 2x3	=	20
x1, x2, x3	≥	0

During the input phase the above LP gets converted to

Min so

ST

s0				-2x1	-3x2	-4x3	=	0 (N)
	s1			+x1	+x2	+x3	=	50
		-s2		+2x1	+x2	+3x3	=	60
			s3	+x1	-x2	+2x3	=	20

Iteration 1 :

Basic = {so,s1,s2,s3} ; Non-basic = {x1,x2,x3}

Step 1 : Call FORMC to form the cost vector. The initial solution, the basic cost vector, the initial basis and its inverse are :

$$\mathbf{x^*} = (0,50,-60,20)^t \qquad (2.25)$$

$$\mathbf{c_b} = (0,0,1,-1)$$

$$\mathbf{B_o} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & -1 & \\ & & & 1 \end{bmatrix} = \mathbf{F_o} \qquad (2.26)$$

$$\mathbf{B_o^{-1}} = \mathbf{B_o} = \mathbf{E_o} \qquad (2.27)$$

It should be noted that the basis **inverse** is **never computed** and **only the basis factorization is done** as explained in subroutine INVERT in chapter 3 and in section 2.2.2. The **inverse used** in this example and in later sections is **only to make explanation easier.**

Step 2 : Call BTRAN to compute the dual prices. Using $\mathbf{c_b}$ and $\mathbf{B_o}$ found above we get

$$\pi \quad = \quad (0,0,-1,-1)$$

Step 3 : Call PRICE to compute the reduced cost of all non-basic columns and also to select a pool of candidate columns with "good" reduced costs. The reduced costs are

$$(0,0,-1,-1) \begin{bmatrix} -2 & -3 & -4 \\ 1 & 1 & 1 \\ 2 & 1 & 3 \\ 1 & -1 & 2 \end{bmatrix} = (-3,0,-5)$$

The pool of candidate columns corresponds to x1 and x3 and is

$$\begin{bmatrix} -2 & -4 \\ 1 & 1 \\ 2 & 3 \\ 1 & 2 \end{bmatrix}$$

Step 3a : Multiple Pricing

Pass 1

(a) Call FTRAN to update the first column in the pool (x1) using $\mathbf{d_j} = \mathbf{B_o}^{-1}\,\mathbf{a_j}$. This results in

$$\mathbf{d_j} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & -1 & \\ & & & 1 \end{bmatrix} \begin{bmatrix} -2 \\ 1 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} -2 \\ 1 \\ -2 \\ 1 \end{bmatrix}$$

(b) Call CHUZR to find the pivot row for this column. The ratios are :

{50/1 , 20/1 }

The minimum ratio is given by row 4 which becomes the pivot row. The change in objective value is (-3)*(20) = -60.

(c) The "best" change so far is -60 with a pivot in the column corresponding to x1 and row 4.

(d) Call FTRAN to update the next column in the pool (x3). The updated column is

$\mathbf{d_j} = (-4,1,-3,2)^t$

(e) Call CHUZR to find the pivot row for x3. The ratios are :

{50/1 , 20/2 }

The minimum ratio is given by row 4 which becomes the pivot row. The change in the objective value is (-5) (10) = -50.

(f) The "best" change is still given by step c. Therefore, the entering variable is x1 and the row corresponding to the leaving variable is row 4.

(g) Call UPBETA to update the solution. This results in

$$\mathbf{x}^* = (0+40,50-20,-60+40,20)^t = (40,30,-20,20)^t \qquad (2.28)$$

(h) We then have

Pivot row = 4 Pivot Column = x1 column

Basic = {s0,s1,s2,x1} Non-basic = {x2,x3,s3}

Also set flag so that on the next call FTRAN would also compute the reduced cost.

Pass 2 : The basis and its inverse at this point are

$$\mathbf{B}_1 = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & -1 & \\ & & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & -2 \\ & 1 & & 1 \\ & & 1 & -2 \\ & & & 1 \end{bmatrix} \qquad (2.29)$$

$$= \quad \mathbf{F_o} \qquad \mathbf{F_1} \tag{2.30}$$

$$\mathbf{B_1^{-1}} = \begin{bmatrix} 1 & & 2 \\ & 1 & -1 \\ & & 1 \end{bmatrix} \begin{bmatrix} 1 & & \\ & 1 & \\ & & -1 \end{bmatrix} \begin{bmatrix} & & \\ & & \\ & & 1 \end{bmatrix} \tag{2.31}$$

(a) Call FTRAN to update the column corresponding to x3 and also to compute the reduced cost. The new update

$$\mathbf{d''_j} = \mathbf{E_1 d_j} = \mathbf{E_1}\ (-4,1,-3,2)^t = (0,-1,1,2)^t$$

where $\mathbf{d_j}$ is the previous update. The reduced cost

$$\pi \mathbf{a_j} = \mathbf{c_b B_1^{-1} a_j} = (0,0,1,0)\ \ (0,-1,1,2)^t = 1$$

Since this is positive the column corresponding to x3 can no longer be a candidate to enter and other columns should be priced out using the latest dual vector.

Iteration 2 **:**

Basic = {s0,s1,s2,x1} Non-basic = {x2,x3,s3}

Step 1 : Call FORMC to form the basic cost vector. Reference to (2.28) gives

$$\mathbf{c_b} \quad = \quad (0,0,1,0)$$

Step 2 : Call BTRAN to compute π. Reference to (2.31) and (2.32) gives

$$\pi \quad = \quad (0,0,1,0)\ \mathbf{E_1}\ \mathbf{E_o} \quad = \quad (0,0,-1,2)$$

Step 3 : Call PRICE to compute the reduced cost of all non-basic columns and also to select a pool of candidate columns with "good" reduced costs. The reduced costs are

$$(0,0,-1,2) \begin{bmatrix} -3 & -4 \\ 1 & 1 \\ 1 & 3 \\ -1 & 2 \end{bmatrix} \quad = \quad (-3,1)$$

The pool consisting of one candidate columns corresponds to x2 and is $(-3,1,1,-1)^t$

Step 3a : Multiple Pricing. Since the pool has only one column there is only one multiple pricing pass.

Pass 1 :

(a) Call FTRAN to update the column corresponding to x2. The update is

$$\mathbf{d_j} \quad = \quad \mathbf{E_1}\ \mathbf{E_o}\ (-3,1,1,-1)^t \quad = \quad (-5,2,-3,-1)^t$$

(b) Call CHUZR to find the pivot row. The ratio is {30/2 } and is given by row 2 which becomes the pivot row.

(c) The change in the objective value is (-3) (15) = -45 with a pivot in the column corresponding to x2 and row 2. Since there is only one column in the pool, the entering variable is x2 and the leaving variable corresponds to row 2.

(d) Call UPBETA to update the solution. The updated solution is

$$\mathbf{x}^* = (40+5(15),15,-20+3(15),20+1(15))^t = (115,15,25,35)^t \quad (2.33)$$

(e) We then have

Pivot column	= x2	Pivot Row	= 2
Basic	= {s0,x2,s2,x1}	Non-basic	= {x3,s3,s1}

Iteration 3 :

Basics = {so,x2,s2,x1} Non-basics = {x3,s3,s1}

Step 1 : Call FORMC to form the basic cost vector. Since all the components of $\mathbf{x}^*$ in (2.33) are positive, phase 2 can be started and therefore

$$\mathbf{c_b} = (1,0,0,0)$$

The basis and its inverse are

$$\mathbf{B_2} = \mathbf{F_o}\ \mathbf{F_1} \begin{bmatrix} 1 & -5 & & \\ & 2 & & \\ & -3 & 1 & \\ & & -1 & 1 \end{bmatrix} \tag{2.34}$$

$$= \mathbf{F_o}\ \mathbf{F_1}\ \mathbf{F_2} \tag{2.35}$$

$$\mathbf{B_2^{-1}} = \begin{bmatrix} 1 & 2.5 & & \\ & 0.5 & & \\ & 1.5 & 1 & \\ & 0.5 & & 1 \end{bmatrix} \mathbf{E_1}\ \mathbf{E_o} \tag{2.36}$$

$$= \mathbf{E_2}\ \mathbf{E_1}\ \mathbf{E_o} \tag{2.37}$$

Step 2 : Call BTRAN to compute π. Reference to (2.36) and (2.37) gives

$$\pi = (1,0,0,0)\ \mathbf{E_2}\ \mathbf{E_1}\ \mathbf{E_o} = (1,2.5,0,-0.5)$$

Step 3 : Call PRICE to compute the reduced costs of all non-basic columns and also to select a pool of candidate columns with "good" reduced costs. The reduced costs are

$$(1,2.5,0,-0.5)\begin{bmatrix} 0 & -4 \\ 1 & 1 \\ 0 & 3 \\ 0 & 2 \end{bmatrix} = \quad (2.5,-2.5)$$

The pool consisting of one candidate column corresponds to x3 and is $(-4,1,3,2)^t$

Step 3a : Multiple Pricing. Since there is only candidate column in the pool, there would be only one multiple pricing pass.

Pass 1 :

(a) Call FTRAN to update the column corresponding to x3. The update is

$$\mathbf{d_j} = \mathbf{E_2}\ \mathbf{E_1}\ \mathbf{E_o}\ (-4,1,3,2)^t = (-2.5,-0.5,-0.5,1.5)^t$$

(b) Call CHUZR to find the pivot row. The ratio is {35/1.5} = 23.33 and is given by row 4 which becomes the pivot row.

(c) The change in the objective value is (-2.5)*(23.33) with a pivot in the column corresponding to x3 and row 4

(d) Call UPBETA to update the solution. The solution is

$$x^* = (115+2.5(23.33),\ 15+.5(23.33),\ 25+1.5(23.33),\ 23.33)^t$$
$$= (173,26.6,36.6,23.3)^t$$

(e) We then have

Pivot column	= x3	Pivot row	= 4
Basic	= {s0,x2,s2,x3}	Non-basic	= {x1,s3,s1}

Iteration 4 **:**

After calls to FORMC, BTRAN and PRICE it can be seen that there will be no entering column and this establishes optimality.

2.5. Dantzig-Wolfe Decomposition:

2.5.1. Theorem on Convex Combinations :

In order to develop the Dantzig-Wolfe (D-W) decomposition a theorem on convex combinations is stated below without proof.

Theorem **:** Let $X = \{\mathbf{x} \mid \mathbf{Ax} = \mathbf{b},\ \mathbf{x} \geq \mathbf{0}\}$ Then a point $\mathbf{x}$ is in X iff it can be written as a convex combination of its extreme points plus a linear combination of generators of extreme rays (homogeneous solutions) of X; (i.e)

$$\mathbf{x} = \sum_j (\beta_j \mathbf{x}^j) \qquad (2.38)$$

where

$$\sum_j (\beta_j \partial_j) = 1 \quad ; \quad \beta_j \geq 0 \text{ for all } j \qquad (2.39)$$

with ∂_j being zero if $\mathbf{x}^j$ is an extreme ray and one if $\mathbf{x}^j$ is an extreme point (2.40)

2.5.2 Development of the Decomposition Principle :

Consider the LP

$$\text{Min} \quad \mathbf{c}\,\mathbf{x} \tag{2.41}$$

ST

(LP3)

$$\mathbf{A_1}\,\mathbf{x} = \mathbf{b_1} \tag{2.42}$$

$$\mathbf{A_2}\,\mathbf{x} = \mathbf{b_2} \tag{2.43}$$

$$\mathbf{x} \geq \mathbf{0} \tag{2.44}$$

where $\mathbf{A_1}$ is m_o by n, $\mathbf{A_2}$ is m by n and all other vactors and matrices having suitable dimensions. Let

$$S = \{\mathbf{x} \mid \mathbf{A_2 x} = \mathbf{b_2},\ \mathbf{x} \geq \mathbf{0}\} \tag{2.45}$$

Then by the theorem on convex combinations any point in S is given by

$$\mathbf{x} = \sum(\beta_j \mathbf{x}^j) \tag{2.46}$$

where

$$\sum(\beta_j \partial_j) = 1 \quad ; \quad \beta_j \geq 0 \text{ for all } j \text{ and} \tag{2.47}$$

∂_j = zero if $\mathbf{x}^j$ is an extreme ray and one if $\mathbf{x}^j$ is an extreme point. (2.48)

Using (2.45) -(2.48) into LP3 gives

$$\text{Min} \quad \sum (cx^j)\beta_j \tag{2.49}$$

ST

(LP4)

$$\sum (A_1 x^j)\beta_j = b_1 \tag{2.50}$$

$$\sum (\beta_j \partial_j) = 1 \tag{2.51}$$

$$\beta \geq 0 \tag{2.52}$$

with ∂_j defined as in (2.48)

The two problems LP3 and LP4 are equivalent since, among all solutions of (2.43) only those that also satisfy (2.42) are chosen by LP4. This equivalent problem is called the master problem and eqn (2.51) of the master problem is known as the convexity constraint. The master problem has (m_o+1) rows compared to the (m_o+m) rows of the original problem LP3. On the other hand it has as many columns as there are extreme points and rays in S and this could be astronomical in many cases. Moreover, these columns are not known a-priori. An elegant scheme developed by Dantzig and Wolfe to find these columns is described below.

Column generation : To solve LP4 all the steps described in section 2.1, except the one of finding the reduced cost can be done routinely. To find the reduced cost

$$cx^j - \pi \begin{bmatrix} A_1 x^j \\ \partial_j \end{bmatrix} \tag{2.53}$$

the column $\begin{bmatrix} A_1 x^j \\ \partial_j \end{bmatrix}$ is required but is not readily available.

Partitioning

$$\pi = [\pi_o, \mu] \tag{2.54}$$

where π_o is the dual vector of the constraints in (2.50) and μ is the dual vector of the constraint (2.51) the reduced cost is

$$(\mathbf{c} - \pi_o \mathbf{A_1})\, \mathbf{x}^j - \mu\partial_j \tag{2.55}$$

To find the entering column by the usual entering column rule we need

$$\min \{(\mathbf{c} - \pi \mathbf{A_1})\mathbf{x}^j\} < \mu\partial_j \tag{2.56}$$

Finding the entering column is equivalent to solving the LP5 since

(a) $\mathbf{x}^j$ denotes the extreme point or extreme ray of S and

(b) the optimal solution of bounded LP is one its extreme points and the solution of an unbounded LP is one of its extreme rays

(LP5)

$$\begin{aligned} &\min && (\mathbf{c} - \pi_o \mathbf{A_1})\mathbf{x} \\ &\text{ST} \\ & && \mathbf{A_2 x} = \mathbf{b_2} \\ & && \mathbf{x} \geq \mathbf{0} \end{aligned}$$

LP5 is frequently called sub-problem. There are two cases at this juncture

(a) the sub-problem yields a solution $\mathbf{x}^*$ such that

$$(\mathbf{c} - \pi_o \mathbf{A}_1)\mathbf{x}^* > \mu\partial_j \tag{2.57}$$

in which case there are no entering columns to the master problem LP4, which inturn implies that LP4 is either optimal or infeasible.

(b) the sub-problem yields a solution $\mathbf{x}^*$ and it passes the **candidacy test** given by (2.58).

$$(\mathbf{c} - \pi_o \mathbf{A}_1)\mathbf{x}^* < \mu\partial_j \tag{2.58}$$

The required entering column is given by (2.59) if $\mathbf{x}^*$ is an extreme point and by (2.60) if $\mathbf{x}^*$ is an extreme ray.

$$\begin{bmatrix} \mathbf{cx}^* \\ \mathbf{A}_1\mathbf{x}^* \\ 1 \end{bmatrix} \tag{2.59}$$

$$\begin{bmatrix} \mathbf{cx}^* \\ \mathbf{A}_1\mathbf{x}^* \\ 0 \end{bmatrix} \tag{2.60}$$

These entering columns generated by solving the sub-problem are frequently called **proposals**.

2.5.3 D-W Agorithm for Block Angular Problems :

A block angular LP with R blocks has the form:

$$\text{Min} \quad z = \sum(\mathbf{c}_r\mathbf{x}_r) \tag{2.61}$$

ST

$$\sum(\mathbf{A1}_r\mathbf{x}_r) = \mathbf{b}_0 \tag{2.62}$$

(LP6)

$$\mathbf{A2}_r\mathbf{x}_r = \mathbf{b}_r \qquad r = 1, \ ,R \tag{2.63}$$

$$\mathbf{x}_r \geq \mathbf{0} \qquad r = 0, \ ,R \tag{2.64}$$

where $\mathbf{c}_r$ is 1 by n_r, $\mathbf{b}_r$ is m_r by 1 and all other vectors and matrices are of suitable dimensions. Pictorially a block angular problem is shown in figure (2.1).

Since each block is independent of the the other, LP6 is decomposed into R sub-problems. Assuming that each block of constraints (2.62) and (2.63) is feasible, the m_o constraints in (2.62), called the coupling constraints, will be used to define a master problem which, as seen later, differs from LP4 in two respects:

(a) It has R convexity rows, one for each block or sub-system.and

(b) Each block is represented by separate set of ß's.

At cycle k of the algorithm, the master problem determines prices π_o^k on the constraints (2.62) and (2.63). With these prices, a subproblem from block r at cycle k can be defined by

$$\text{min} \quad v_r^k = (\mathbf{c}_r - \pi_o^k\mathbf{A1}_r)\mathbf{x}_r \tag{2.65}$$

ST

$$\mathbf{A2}_r\mathbf{x}_r = \mathbf{b}_r \tag{2.66}$$

($LP7_r^k$)

$$\mathbf{x}_r \geq \mathbf{0} \tag{2.67}$$

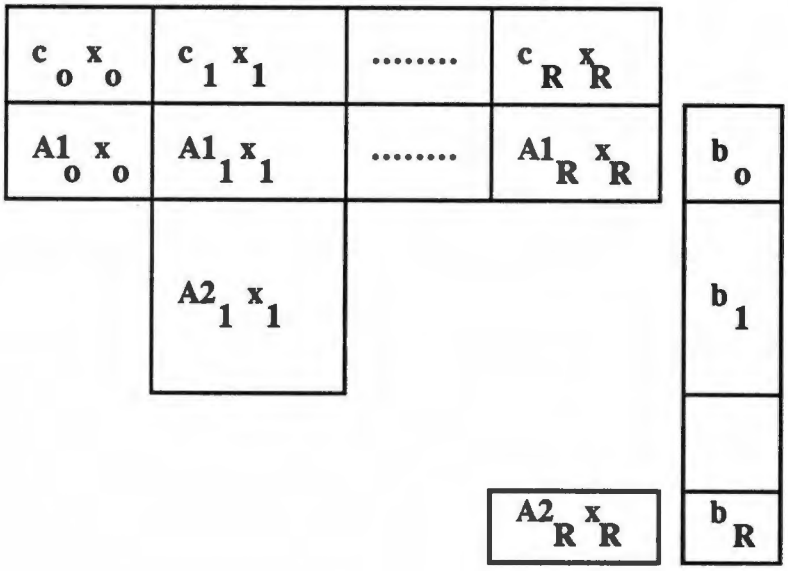


Figure 2.1 : Pictorial representation of a block angular linear program.

$LP7_r^k$ has either an extreme point or an extreme ray solution and in either case, if the solution x_r^{*k} passes the candidacy test, the k_r^{th} proposal generated by block r through cycle k for the master problem, can be defined as

$$\begin{bmatrix} p_{rkr} \\ q_r^{kr} \end{bmatrix} = \begin{bmatrix} c_r x_r^{*k} \\ A1_r x_r^{*k} \end{bmatrix} \tag{2.68}$$

Representing all the proposals from block r as a matrix, we have

$$\begin{bmatrix} p_r^{kr} \\ Q_r^{kr} \end{bmatrix} = \begin{bmatrix} p_{r1},\ldots,p_r^{kr} \\ q_{r1},\ldots,q_r^{kr} \end{bmatrix} \tag{2.69}$$

The master problem $LP8^k$ shown below is then obtained from (2.61) and (2.62) by substituting x_r, r = 1,...R with the convex combinations of the proposals.

$$\text{min} \quad z^k = c_o x_o + \sum(p_r^{kr} \beta_r^{kr}) \tag{2.70}$$

ST

$(LP8^k)$

$$A1_o x_o + \sum(Q_r^{kr}\beta_r^{kr}) = b_o \tag{2.71}$$

$$\partial_r^{kr}\beta_r^{kr} = 1 \tag{2.72}$$

$$x_o \geq 0, \quad \beta_r^{kr} \geq 0; \quad r = 1,\ldots R \tag{2.73}$$

where $\beta_r^{kr} = [\beta_{r1},\ldots,\beta_r^{kr}]^t$ with β_{rl} being the weight of the l^{th} proposal from block r; and $\partial_r^{kr} = [\partial_{r1},\ldots\partial_r^{kr}]$ with ∂_{rl} being one if the l^{th} proposal corresponds to an extreme

point and zero otherwise. Finally, denote the dual vector of LP8^k by $[\pi_o^k, \mu_1^k, ..., \mu_R^k]$ where π_o^k corresponds to the coupling constraints (2.71) and μ_r^k to the r^{th} convexity constraint in (2.72). The algorithm, consisting of three phases can now be described.

Phase 1 (For feasibility) :

Step 1 : (Feasibility of each block). Solve sub-problem LP7$_r^k$ for r = 1 to R. If one of the sub-problems is infeasible, then the whole problem is infeasible. Hence STOP. Otherwise, generate a proposal for the master problem LP8^1.

Step 2 : (Feasibility of coupling constraints). Use phase 2 to minimize the sum of infeasibilities in LP8^1. If the optimal value is positive then the problem is infeasible. hence STOP. Otherwise, restore the original objective function and proceed to phase 2.

Phase 2 (For optimality) : *Step 0* : Set k to 1 if the objective function is sum of infeasibilities. If the objective is the original one, then set z'^{k-1},the lower bound on the optimal value of the objective function.

Step 1 : Solve current master problem LP8^k. If it is unbounded, then the original problem is unbounded. Hence STOP. Otherwise, send the dual vector $[\pi_o^k, \mu_r^k]$ to sub-problem r for r = 1 to R.

Step 2 : Solve sub-problem LP7$_r^k$. If LP7$_r^k$ is unbounded or, if it is bounded with an optimal solution of x_r^{*k} and also passes the candidacy test given by (2.74), then generate a proposal according to (2.68)

$$v_r^{*k} = (c_r - \pi_o^k A 1_r) x_r^{*k} < \mu_r^k \qquad (2.74)$$

Step 3 : (Stopping criterion). If no proposal has been generated in step 2 of phase 2 then STOP.If the objective function is sum of infeasibilities, terminate step 2 of phase 1.

Otherwise proceed to phase 3. If all sub-problems have been optimized and the objective function is the original one, update lower bound on optimal value according to:

$$z'^k = \max(z'^{k-1}, z^k + \sum (v_r^{*k} - \mu_r^k)) \quad (2.75)$$

If $z^k - z'^k < \Delta$, where Δ is some small, positive, user supplied tolerance, then proceed to phase 3. Otherwise set k to k+1 and go to step 1 of phase 2.

Phase 3 (For reconstruction of the primal solution) :

Step 1 (Compute allocation). Let the *transpose* of the optimal solution of LP8k be $[z^{*k}; x_0^*, \beta_1^{*k}, \ldots, \beta_R^{*k}]$ Compute

$$y_r = Q_r^{kr} \beta_r^{*kr} \quad \text{, for } r = 1,\ldots,R \text{ and} \quad (2.76)$$

define for r = 1,...R:

$$\begin{aligned} &\min \quad c_r x_r \\ &\text{ST} \\ (LP9_r) \quad &A1_r x_r = y_r \\ &A2_r x_r = b_r \\ &x_r \geq 0 \end{aligned} \quad (2.77)$$

Step 2 (Reconstruction). Solve $LP9_r$ for x_r^*, r = 1,...,R. Then $[x_0^*, x_1^*, \ldots, x_R^*]$ constitutes a *transposed* solution to the original problem with objective value z^{*k}.

2.6. Specifications of a Robust Block D-W Implementation :

There are three major aspects in the efficient design of a decomposition code :

a. Solving the master and sub-problems by the RSM. This aspect has already been discussed in section 2.2.

b. Data handling to update and solve the master and sub-problems. This aspect is discussed in section 2.6.1.

c. Computational strategies discussed in section 2.6.2.

2.6.1. Augmenting Coupling Constraints to Sub-problem :

As seen in section 2.5.3 in order to be able to generate columns for the master problem, all subprobelms have to solved, which in turn requires the sub-problem objective given by (2.65).

But, it is not desirable to compute and store the objective function explicitly due to data management considerations :

(a) Since the non-zero structure of the master dual variable conceivably changes between cycles, forming the sub-problem objective explicitly, would involve inserting new coefficients which cannot be easily done if column-wise packing is used.

(b) One would have to make the distinction in data structure between **A1** and **A2**, and this would preclude the use of column-wise packing.

(c) A copy of **c** has to be kept to construct the cost vector.

Fortunately, there is no need to store the sub-problem objective explicitly. To see this recall that in the RSM the objective is required only to find the reduced cost

which can be done if sub-problem structure is as shown in figure (2.2). The basis $\mathbf{B_r}$, and the inverse of the basis $\mathbf{B_r^{-1}}$, corresponding to this structure are shown in figures (2.3) and (2.4) respectively. The steps of the RSM and D-W algorithms affected by this structure are now stated as follows :.

Step 1 of RSM : (Basic cost vector). In order for the remaining steps to work the basic cost vector is taken to be

$$[1,-\pi_o^k,0] \tag{2.78}$$

Step 2 of RSM : (Find dual prices) Pre-multiplying (2.78) to the inverse given in figure 4 yields :

$$[1,-\pi_o^k,-\pi_r^k] \tag{2.79}$$

so that the dual vector is given, modulo a minus sign, by the third term.

Step 3 of RSM : (Find reduced cost) Pre-multiplying (2.79) to the j^{th} non-basic column of the matrix given by figure (2.2) yields $c_j - \pi_o^k \mathbf{a1}_{rj} - \pi_r^k \mathbf{a2}_{rj}$. This by definition is the required reduced cost.

Extreme point proposal generation . Pre-multiplying the basis inverse by the right hand side vector $[0,0,\mathbf{b}^t_r]^t$, yields

$$\begin{bmatrix} -\mathbf{c}_{rb}\mathbf{B2}_r^{-1}\mathbf{b}_r \\ -\mathbf{B1}_r\mathbf{B2}_r^{-1}\mathbf{b}_r \\ \mathbf{B2}_r^{-1}\mathbf{b}_r \end{bmatrix} \tag{2.80}$$

which is the required extreme point proposal.

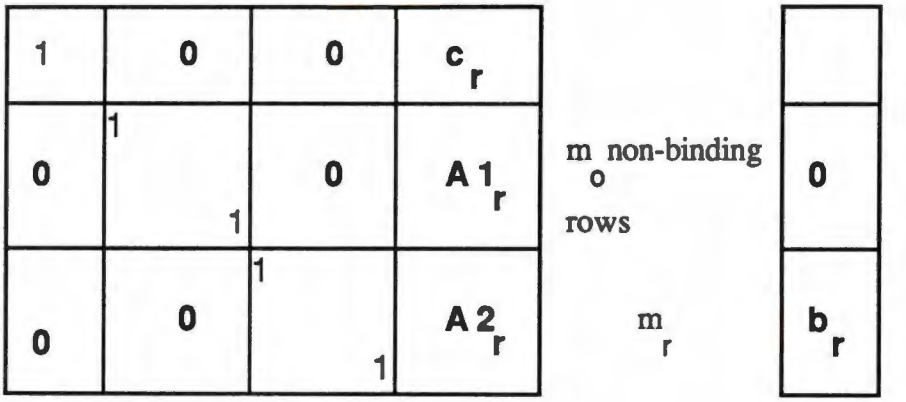


Figure 2.2 : Tabular representation of sub-problem data

1	0	$c_r b$
0	1 (1)	$B1_r$
0	0	$B2_r$

Figure 2.3 : Sub-problem basis

1	0	$c_r b \; B2_r^{-1}$
0	1 (1)	$-B1_r B2_r^{-1}$
0	0	$B2_r^{-1}$

Figure 2.4 : Sub-problem basis inverse

Extreme ray proposal generation. An extreme ray solution (same as unbounded solution) is obtained if a column prices out negatively. Suppose column j prices out negative; then pre-multiplying the basis inverse to the j^{th} column $[c_{rj}, \mathbf{a1}^t_{rj}, \mathbf{a2}^t_{rj}]^t$ yields:

$$\begin{bmatrix} c_{rj} - \mathbf{c}_{rb}\mathbf{B2}_r^{-1}\mathbf{a2}_{rj} \\ \mathbf{a1}_{rj} - \mathbf{B1}_r\mathbf{B2}_r^{-1}\mathbf{a2}_{rj} \\ \mathbf{B2}_r^{-1}\mathbf{a2}_{rj} \end{bmatrix} \qquad (2.81)$$

which is the required extreme ray proposal.

2.6.2 Computational Strategies :

The third aspect concerns refinements of the basic algorithm aimed at accelerating convergence. These refinements are heuristic in nature and their benefits cannot be guaranteed on theoretical grounds.

Intermediate stage proposal generation. Since any solution of the sub-problem satisfying the candidacy test given by (2.74) can lead to an improvement in the master problem, it can be used to generate a proposal. This allows the possibility of obtaining a multitude of proposals from a sub-problem during a single cycle. Empirical experience of Ho and Loute (1981) show that this strategy can accelerate convergence. An example of a mechanism to control proposal generation, used by Ho and Manne (1974) depends on three factors :

a. frequency control,

b. relative improvement of the reduced cost of the potential proposal and

c. total number of proposals generated.

The user specifies four parameters : q_{freq}, q_{perc}, q_{max} and q_{intr}. A proposal is generated as soon as the current solution passes the candidacy test. Then a proposal is generated every q_{freq} iterations or whenever the reduced cost of the potential proposal is bettered by q_{perc}. No more than q_{max} proposals are sent to the master. The current cycle is interrupted once a total of q_{intr} proposals have been generated by all sub-problems during the current cycle.

Interrupted cycles. The convergence of the decomposition algorithm is not lost if a cycle is interrupted. In fact we will see how a cycle interruption can be beneficial in some cases. A cycle can be interrupted for any of the following reasons :

1. As seen above, a cycle is interrupted if during the course of solving a sub-problem, a total of q_{intr} proposals gets generated in the current cycle. Subsequently, the next cycle is started with the next sub-problem as the first sub-problem. This is done to give all sub-problems an equal opportunity for generating proposals.
2. In contrast with the above reason, very often it may be useful to favor some sub-problems to generate more proposals than others. This is done if some knowledge of the nature of the problem indicates that some of the sub-problems are likely to be more active in proposal generation than others. In this case a cycle is interrupted if a sub-problem is not optimal and has generated q_{max} proposals. The next cycle is started with the same sub-problem as the first sub-problem.

Terminal stage proposal generation. A problem is in its terminal stage if it is optimal or if an unbounded column is discovered. If the problem is optimal one proposal is generated corresponding to the extreme point of the optimal solution. When it is unbounded two companion proposals are generated. The first proposal is from the generator of the extreme ray that caused the unboundedness and the second one is from the extreme point corresponding to the current basic feasible solution. In order to

discover other extreme rays, the column causing the undoundedness is fixed at zero and the optimization of the restricted sub-problem is pursued.

Proposal purging. As the size of the master problem grows with the addition of new proposals, the inactive columns of the master must be purged periodically to keep storage requirements within acceptable limits.

2.7. Pseudo-code for Block D-W Algorithm :

The pseudo-code of the program DECOMP, a FORTRAN version which implements the robust specifications of section 2.6 into the block D-W algorithm developed in section 2.5.3, is described below. As before those names used in this pseudo-code which are also used verbatim in DECOMP are italicized.

Program *DECOMP* ;

/*Only those procedures which are critical to understanding DECOMP are shown */

Procedure *INPUT* ;

- Read MPS data for the current problem

Procedure *CHECK* ;

- Check to see if a proposal can be written to the proposal buffer.

Procedure *PACK* ;

- Packs proposals from a proposal buffer into the data array.

Procedure SIMPLEX ;

/* Refer to section 2.3 for details */

- Solves the given LP. The changes whose necessity was stated at the end of section 2.3 are as follows. Call *CHECK* if an unbounded column is identified by*CHUZR*. Also call *CHECK* at the end of every basis update.

Procedure *INVERT* ;

- Does L-U factorization of the given basis and also updates the solution.

Procedure *NORMAL* ;

- If eta size gets large or if inversion frequency is attained *INVERT* ;
- If solving master problem and if there are candidates *PACK* ;
- SIMPLEX ;
- If SIMPLEX identifies an optimal solution then *CHECK* ;

Procedure *INDATA* ;

/* Does step 1 of D-W phase 1 */

For each problem

- *INPUT* ;

If the problem is not the master problem

- *NORMAL* ;
- If problem is infeasible then ;
- Declare the whole problem infeasible and STOP.

Else

- Write the proposal to its buffer.

End If

End If

End For

- Read the master problem into core ;

-*PACK* ;

Procedure *POLICY(KK)* ;

- Based on the strategy, set KK to the number of problems that have to be solved in the current cycle. Also, set the order in which the problems are to be solved and the proposal policy.

Procedure SOLVMSTR(KCYC,RELDIFF) ;

/* Called to optimize master problem */

- Increment KCYC to indicate the beginning of a new cycle ;
- *NORMAL* ;
- If the master problem (coupling constraints) is infeasible then STOP;
- Compute the relative difference in the objective value (RELDIFF).
- If the RELDIFF > tolerance for the stopping criterion (RSTOP) then
    - Save the dual vector of the master problem in a buffer.

End If

Procedure SOLVSUB(KKK,KCYC) ;

/* Called to optimize sub-problem */

- *NORMAL* ;

- Decide if the current cycle is to be interrupted. (cf. Section 2.6.2)
- If cycle is to be interrupted then
    - SOLVMSTR(KCYC,RELDIFF) ;
    - If RELDIFF < RSTOP then the first sub-problem to be solved in the new cycle depends on the user's strategy and is determined as follows.
        Case 1 : the KKKth sub-problem is the first sub-problem of the new cycle. In this case
        SOLVSUB(KKK,KCYC) ;
        Case 2 : the (KKK+1)st sub-problem is the first sub-problem of the new cycle. In this case just proceed.
    End If

Procedure *MASTER* ;

/* Does step 2 of D-W phase 1 and D-W phase 2 */

- Set the cycle number KCYC to 0 ;
- SOLVMSTR(KCYC,RELDIFF) ;
- While the RELDIFF is less than tolerance of the stopping criterion
    - *POLICY(KK)* ;
    - For each problem KKK = 1 to KK-1
        - SOLVSUB(KKK,KCYC) ;
      End For
    - SOLVMSTR(KCYC,RELDIFF) ;
  End While

Procedure *RESULT* ;

/* Does D-W phase 3 */

- Compute the allocation to each sub-problem
- Solve sub-problem $LP9_r$ (r = 1 to R) to reconstruct the solution x_r^*

Begin /* Main Routine */

- *INDATA* ;
- *MASTER* ;
- *RESULT* ;

End

CHAPTER 3

Program Subroutines

In this chapter the functions of all sub-routines in the program are explained in detail in alphabetical order. The explanation is organized under a combination of the following headings : "Major Task," "Inputs," "Outputs," "Called By," "Calling," "Description," "Steps" and "Detailed Comments" (sometimes simply "Comments"). The "description" section contains a narrative of the functions of the subroutine and a summary of this is given in "steps." The actual code is related to these functions in "comments." The nature of the contents under the other headings is self explanatory.

3.1. Subroutine BTRAN :

3.1.1. Major Tasks :

Compute the simplex multipliers π by solving the system $\pi B = c_b$.

3.1.2. Inputs :

1. The buffer YA(*,*) with $\mathbf{c_b}$ stored in one of its columns.

2. KVEC indicating the column of YA(*,*) containing $\mathbf{c_b}$.

3. Eta file in the array A(*).

3.1.3. Outputs :

The dual vector π in the buffer YA(*,KVEC).

3.1.4. Called By :

NORMAL after the cost vector has been formed by a call to FORMC.

3.1.5. Description :

Solving for π from the system $\pi\mathbf{B} = \mathbf{c_b}$ is done through transformation of $\mathbf{c_b}$ by a set of elementary eta matrices as indicated in (3.1)

$$\pi = \mathbf{c_b E_k ... E_1} \tag{3.1}$$

where

$$\mathbf{B} = \mathbf{F_1 ... F_k} \tag{3.2}$$

and

$$\mathbf{F_i^{-1}} = \mathbf{E_i} \tag{3.3}$$

Let $\mathbf{E_j}$ be the j^{th} eta matrix and let the p^{th} column of $\mathbf{E_j}$ be the non-trivial column as shown below.

$$\mathbf{E_j} = [\mathbf{u_1},...,\mathbf{u_{p-1}},\mathbf{n_p},\mathbf{u_{p+1}},...,\mathbf{u_n}] \qquad j = 1,...,k \qquad (3.4)$$

where

$$\mathbf{n_p} = \begin{cases} -e_{ip}/e_{pp} & i \neq p \\ 1/e_{pp} & i = p \end{cases} \qquad (3.5)$$

and $\mathbf{u_i}$ is the i^{th} unit vector.

Without loss of generality consider the transformation by the eta matrix $\mathbf{E_j}$. Note that at this point $\mathbf{c_b}$ has already been transformed to $\mathbf{c''_b}$ (say) by eta matrices $\mathbf{E_k},...,\mathbf{E_{j+1}}$. Therefore, the j^{th} stage of the transformation is

$$\mathbf{c''_b E_j} = \begin{cases} (\mathbf{c''_b})_i & i \neq p \\ \{ (\mathbf{c''_b})_p - \sum_{j \neq p} (\mathbf{c''_b})_j e_{jp} \} / e_{pp} & i = p \end{cases} \qquad (3.6)$$

$$= \begin{cases} (\mathbf{c''_b})_i & i \neq p \\ (R-S) / e_{pp} & i = p \end{cases} \qquad (3.7)$$

where

$$R = (c''_B)_p \text{ and } S = \sum_{j \neq p} (c''_B)_j e_{jp} \qquad (3.8)$$

since only the p^{th} column of E_j is non-trivial. From (3.6) it is clear that at the j^{th} stage, (j = 1,...k) the only element that changes in c''_B is the one corresponding to the pivot position of E_j and this element is computed by an inner product. Since the pivot element of each eta is stored first, the pivot position of the eta is readily inferred. Therefore, modification of the pivot element of c''_B can be done without requiring explicit storage of the pivot element of the eta vector.

3.1.6. Detailed Comments :

```
C*****************************************************************
C
C     NETA        : Number of etas generated.
C     YA(*,*)     : Buffer one of whose columns contains the cost vector.
C     KVEC        : Indicates the column of YA(*,*) containing the cost
C                   vector.
C     KFASE       : Normally set to 2 in which case the cost vector is set
C                   up in YA(*,6). When a sub-problem's solution is
C                   infeasible in D-W phase 2 then KFASE is set to 1 and
C                   in this case the cost vector is formed in YA(*,5).
C     LL          : First non-zero element of the non-trivial column of the
```

C eta E_j being used in the transformation.

C KK : Last non-zero element of the non-trivial column of the

C eta E_j.

C IPIV : Pivot position of the non-trivial column of E_j

C A(*) : Array containing the eta file.

C

C 1. If there are no etas then RETURN. Else proceed. [line 1]

C 2. Find the column in YA(*,*) where the cost vector is stored. This is

C decided by KFASE. [lines 3-4]

C 3. For each eta vector j do :

C

C - Set IK to the eta index starting from the last since by (3.1) the

C transformation starts with the last eta and ends with the first eta.

C [line 6]

C - Set LL and KK to the first and the last non-zero element of the

C non-trivial column of E_j. [lines 7-8].

C - Since the pivot element of the eta vector is stored first, the row

C position of the first element of the eta vector should be the pivot

C position. Set DP to pivot element. Hence, DP now contains

C $1/e_{pp}$ of (3.6). [lines 9-10]

C - Set DY to R of the eqn (3.8). [line 11]

C - If pivot element is the only element of the eta vector then S of

C (3.8) is 0. Hence just find DY*DP = R/epp and assign it to

C YA(IPIV,KVEC) since only the pivot position alters.

```
C                [lines 13,19]
C                Else proceed.
C              - Find DSUM = S of (3.8).  [lines 15-18]
C                Also find (R - S)/e_pp and assign it to the YA(IPIV,KVEC)
C                since only the pivot position alters.  [line 19]
C
C        End For
C
C*********************************************************************
      IF (NETA .LE. 0) GOTO 9000                                    1
      CALL TICTAC(JTIM)                                             2
      KVEC = 6                                                      3
      IF (KFASE .EQ. 1) KFASE = 5                                   4
      DO 1000 I = 1,NETA                                            5
      IK = NETA - I + 1                                             6
      LL = LE(IK)                                                   7
      KK = LE(IK+1) - 1                                             8
      IPIV = IA(LL)                                                 9
      DP = A(LL)                                                    10
      DY = YA(IPIV,KVEC)                                            11
      DSUM = 0.0                                                    12
      IF (LL .LE. KK) GOTO 600                                      13
      LL = LL + 1                                                   14
      DO 500 J = LL,KK                                              15
      IR = IA(J)                                                    16
```

```
      DSUM = DSUM + A(J)*YA(IPIV,KVEC)                    17
500   CONTINUE                                            18
600   YA(IPIV,KVEC) = (DY - DSUM)*DP                      19
1000  CONTINUE                                            20
      CALL TICTAC(ITINV)                                  21
      JTINV = JTINV + ITINV - JTIM                        22
      RETURN                                              23
      END                                                 24
```

3.2. Subroutine CHANGE :

3.2.1. Major Task :

Exchange problems between the direct access device (DAD) and core.

3.2.2. Input :

1. LSUP indicating the problem that is to be read into core.
2. NOLA(*,*) indicating where in the direct access device each problem is stored.

3.2.3. Called By :

INDATA,NORMAL,RESULT

3.2.4. Calling :

VECTOR to actually effect the exchange of problems.

3.2.5. Description :

CHANGE first checks to see if the problem to be read is already in core. If the problem is not in core then problems have to be exchanged between the DAD and core as follows. First, the position in DAD where the problem in core is to be written is obtained. The subroutine VECTOR is called to write the problem in core

to DAD. Next, the position where the problem to be read is residing in DAD is obtained and again VECTOR is called to read the problem into core.

CHANGE, therefore consists of the following steps.

3.2.6. Steps :

1. Write the problem in core to DAD.
2. Read the problem required in core from DAD.

3.2.7. Detailed Comments :

```
C*************************************************************************
C
C       Step 1 :
C
C       LSUP          : Index of the problem to be read.
C       LCORE         : Indicates if the problem to be read is in core.
C       A(*)          : Data array.
C       NOLA(2,*)     : Indicates in the data array A(*) the end of the eta file of
C                       the *th problem.
C       NOLA(3,*)     : Indicates the position in DAD where all the data (with eta)
C                       of the *th problem is residing.
C       KIKA          : Flag to VECTOR. When one it indicates that VECTOR
C                       should write the problem currently in core. When two it
C                       indicates that the LSUPth problem is to be read from
```

```
C                    DAD by VECTOR.
C     LMAX           : Number of sub-problems.
C     LMAXP1         : LMAX plus one.
C     KK             : Index of the problem that is to be read. KK = LMAXP1
C                      indicates the master problem.
C
C     1. If the problem is already in core then there is nothing to be done. Hence
C        RETURN. [line 3].
C        Otherwise proceed.
C     2. Set NETA to zero if the number of iterations since basis inversion has
C        exceeded the inversion frequency or if there is not enough space in the
C        data array A(*). [lines 6-10].
C        Recall, NETA = 0 triggers a re-inversion.
C     3. Set IAUX to NOLA(2,KK). Since the eta vectors are written after the
C        data this would indicate the number of elements of the array A(*) that is
C        to be read. [lines 11,14].
C     4. Set KD to point to the record in DAD to which the problem in core is to
C        be written. [line 13].
C     5. Set KIKA to 1 (indicating write) and call VECTOR to write the problem
C        in core to the KD^th record of DAD.
C
C*************************************************************************
      DIMENSION NB(3612)                                              1
      EQUIVALENCE (JH(1),NB(1))                                       2
      IF(LSUP .EQ. LCORE)GO TO 5000                                   3
```

```
      CALL TICTAC(LI)                                                  4
      IF (LCORE .EQ. -1) GO TO 3000                                    5
      INVF = INVFRQ                                                    6
      IF (LCORE .EQ. 0) INVF = INVFMA                                  7
      KK = LCORE                                                       8
      IF(KK .EQ. 0)KK = LMAXP1                                         9
      IF(NELEM.GT.(NEMAX-NRMAX).OR.ITSINV.GT.                          10
     1(4*INVF)/5)NETA = 0
      NOLA(2,KK) = LE(NETA+1) - 1                                      11
      KIKA = 1                                                         12
2050  KD = NOLA(3,KK)                                                  13
      IAUX = NOLA(2,KK)                                                14
      CALL VECTOR(KIKA,A,IA,NB,IAUX)                                   15


C*********************************************************************
C
C     Step 2 :
C
C     Refer to step 1 for variable definitions.
C
C     1. Set KIKA to two to indicate read.  [line 20].
C     2. Set KD to the record in DAD from where the LSUP^th problem is to
C        be read.  [lines 13].
C     3. Set IAUX to indicate the numberof elements of the data array A(*) that is
```

```
C           to be read.  [line 14].
C       4. Call VECTOR to read the problem into core.  [line 15].
C
C*********************************************************************

 2500   IF(LCORE .EQ. LSUP)GO TO 4000                          16
 3000   LCORE = LSUP                                           17
        KK = LSUP                                              18
        IF(LSUP .EQ. 0)KK = LMAXP1                             19
        KIKA = 2                                               20
        GO TO 2050                                             21
 4000   CALL TICTAC(LF)                                        22
        NX=NX+LF-LI                                            23
 5000   RETURN                                                 24
        END                                                    25
```

3.3. Subroutine CHECK :

3.3.1. Major Tasks :

Checks if a proposal can be made.

3.3.2. Input :

YA(*,KVEC) contains the unbounded column if any.

3.3.3. Outputs :

1. When appropriate, proposals are made in D-W phase 2. The proposals, if any, will be in Z(*,*).
2. Sets flag (IRTN) to indicate if an INVERT was done.
3. Sets flag KOUT to indicate that multiple pricing has to be interrupted.

3.3.4. Called By :

NORMAL.

3.3.5. Calling :

1. INVERT to make a more accurate proposal.
2. FORMC subsequent to the call to INVERT.

3. If an unbounded proposal is to be made then, UNPACK is called to unpack the column causing the unboundedness.
4. If an unbounded proposal is to be made then, FTRAN is called to update the column causing unboundedness.

3.3.6. Description :

The object of CHECK is to see if a proposal can be generated from the sub-problem being solved. Reference to section 2.5.3 shows that a proposal can be generated as soon as the current solution passes the candidacy test. All subsequent solutions until optimality can be used to generate proposals if the user chooses the appropriate strategy. Strategies can be classified into two major types. The first type permits intermediate proposal generation by allowing a proposal to be generated during intermediate stages of solving a sub-problem. In the second type proposals can be generated only at optimality or unboundedness. For either type a proposal can be generated only if further criteria in addition to the candidacy test are met.

Case 1 : (No intermediate proposals) The test in this case would simply be to check for optimality or unboundedness. If the sub-problem is optimal then the proposal is from the extreme point corresponding to the optimal solution. If unbounded then two companinon proposals are generated. One is from the columnm that caused the unboundedness and the other is from the extreme point corresponding to the current basic feasible solution.

Case 2 : (Intermediate proposals) There are several tests as described

below.

Test A : If the reduced cost of the sub-problem's entering variable is "better" than some user specified value RCTEST or if the allowable increase in the potential entering column is less than 10 then Test A is passed.

Test B : If the allowable increase in the sub-problem's potential entering column is less than 100 then Test B is passed.

Percentage test : If the relative improvement in the reduced cost of the proposal column is greater than some user specified value RPERC then the Percentage test is passed.

Frequency test : If the frequency of proposal generation is better than some user specified value NCANFR then the frequency test is passed.

Proposals generated in this case are extreme point proposals corresponding to the current basic feasible solution.

Actual generation of proposals depends on whether an extreme point or an extreme ray is selected for generation and involves writing an extreme point solution or an extreme ray column to a column of the proposal buffer Z(*,*).

Case 1 : (Extreme point proposal) Reference to subroutine NORMAL shows that it is possible for two consecutive calls to CHECK to correspond to the same extreme point. In this case a proposal need not be written to Z(*,*). Otherwise, the next step is to decide where in Z(*,*) to write the proposal, and this depends on the strategy chosen

Case 1.1 : (Intermediate proposals not permitted) If there is space in Z(*,*) then the proposal is accomodated in the first

available free column. On the other hand if the number of proposals generated exceeds the number of columns NPROS in Z(*,*), then the current contents of Z(*,*) are written to the end of the unit 8 work file and subsequent proposals are written beginning with the first column of Z(*,*).

Case 1.2 : (Intermediate proposals permitted) Let MAXCA be the number of proposals sent to the master problem. This case is further broken up into two sub-cases. In the first sub-case more than MAXCA proposals can be generated but only last MAXCA of them are sent to the master problem. For example, assume that MAXCA proposals have been generated. Then the $(MAXCA+1)^{st}$ proposal overwrites the proposal in the 1st column of Z(*,*). This way, no more than MAXCA proposals are kept. Hence, the count on the number of proposals kept, is incremented only if the number generated is less than MAXCA. In the second sub-case the solution of the problem is terminated for the current cycle once MAXCA proposals have been generated and the flag KOUT is set to 1. The decision about where in Z(*,*) to write the proposal is the same as case 1.1.

The last step would be to actually write the proposal to Z(*,*). This is done by copying the first NROWO (number of coupling rows) elements of the solution vector X(*) into the appropriate column of Z(*,*). Depending on the number of iterations since the last INVERT, it may be necessary to perform an INVERT before copying it to Z(*,*) to ensure sending a more accurate proposal.

Case 2 : (Extreme ray proposal) This is very similar to case 1 but for the following differences.

1. In this case the proposal is the column that caused the unboundedness and this would have to be updated before copying it to Z(*,*).
2. After writing an extreme ray column, case 1 is called to see if a companion extreme point proposal can be generated.

CHECK therefore comprises the following steps

3.3.7. Steps :

1. Determine which proposal tests, if any, have to be done.
2. Perform
   (a) test a, test b, percentage test and frequency test **or**
   (b) unbounded and optimality test as is appropriate.
3. Incorporate
   (a) an extreme point proposal **or**
   (b) an extreme ray proposal and thereafter, possibly an extreme point proposal.

3.3.8. Detailed Comments :

```
C***********************************************************************
C
C     Step 1 :
```

```
C
C     MAST        : Indicates stage of the problem. If 0 then executing
C                   step 1 of D-W phase 1. If 1 then executing step 2 of
C                   D-W phase 1 or D-W phase 2. If 2 then executing
C                   D-W phase 3.
C     MSTAT       : Indicates status of the problem.
C     DX          : Sub-problem objective value minus the dual variable of
C                   the corresponding convexity constraint. DX is
C                   therefore the reduced cost of the proposal column.
C     KRIT(*)     : Indicates the type of the strategy chosen by the user. If
C                   intermediate proposals can be generated then its value
C                   is 2. Otherwise its value is 1.
C     LSUB        : Indicates problem index. Zero for master problem.
C
C     Case 1 : (Proposal not written due the problem's status)
C     - While executing step 1 of D-W phase 1 (MAST = 0) the proposal will
C       be copied in the subroutine INDATA. [line 4]
C     - If the problem's current solution is infeasible or if the master problem is
C       being solved then a proposal need not be generated. [line 3]
C
C     Case 2 : (Proposal not generated due to the strategy chosen by user)
C     - If MSTAT = QF and the user has chosen the strategy of not generating
C     intermediate proposals then RETURN. [lines 6-7]
C
C     Case 3 : (Intermediate proposals can be generated)
```

```
C       - Go to step 2a for further tests.
C
C       Case 4 : (Optimal and unbounded proposal)
C       - If MSTAT ≠ QF then it should be optimal or unbounded since if it were
C         QI then we would have gone to case 1. Go to step 2b
C
C*****************************************************************
        KOUT = 0                                                    1
        IRTN = 0                                                    2
         IF(KFASE .EQ. 1 .OR. LSUB .EQ. 0)GO TO 1000                3
        IF(MAST .NE. 1)GO TO 1000                                   4
        DX = DOB + YTEMP(LSUB)                                      5
        IF(MSTAT .NE. QF)GO TO 2100                                 6
        MM = KRIT(KKK)                                              7
        GO TO (5000,2000), MM                                       8
1000    IF(MSTAT.NE. QF .AND. MSTAT .NE. QI)KOUT = 1                9
        IF (LSUB .NE. 0) GO TO 5000                                 10
        IF (INCO .EQ. -1) KOUT = 1                                  11
        GO TO 5000                                                  12
C*****************************************************************
C
C       Step 2a :
C
C       DX          : Reduced cost of the potential proposal. [line 5]
C       DCMIN       : Reduced cost of the sub-problem's candidate
```

```
C                    column.
C     RCTEST         : User supplied parameter for test a.
C     DP             : Allowable increase in the sub-problem's candidate
C                      column.
C     IPROP(*)       : Number of proposals generated under the (*) th
C                      criterion.
C     QPROP          : Criterion of the most recent proposal generated.
C
C     - If the candidacy test is not met then RETURN.  Else proceed.  [line 13]
C     - Test A.  If Test A is not met then proceed.  [lines 14-17]
C     - Test B.  If Test B is not met then proceed.  [lines 18-21]
C     - Percentage test.  If percentage test is not met then proceed. [lines 22-26]
C     - Frequency test.  If this test is not met then RETURN.  [lines 27-29]
C
C*****************************************************************
2000  IF(DX .GE. -ZTCAND)GO TO 5000                                  13
      IF(DCMIN(IYIV) .GT. RCTEST .OR. DP .LT. 10.)GO TO 2010         14
      IPROP(3) = IPROP(3) + 1                                        15
      QPROP = QA                                                     16
      GO TO 2450                                                     17
2010  IF(DP .LT. 100.)GO TO 2020                                     18
      IPROP(4) = IPROP(4) + 1                                        19
      QPROP = QB                                                     20
      GO TO 2450                                                     21
2020  DP = RPERC*DZ                                                  22
```

```
      IF (DX .GE. DP) GO TO 2030                                    23
      IPROP(1) = IPROP(1) + 1                                       24
      QPROP = QR                                                    25
      GO TO 2400                                                    26
 2030 IF(ITCNT - ITSLCA .LT. NCANFR)GO TO 5000                      27
      IPROP(2) = IPROP(2) + 1                                       28
      QPROP = QF                                                    29

C***********************************************************************
C
C     Step 2b :
C
C     - Unbounded test.  If this test is not met then proceed.  [lines 30-33]
C     - Optiamlity test.  [lines 34-39]
C
C     - If optimality test is met then KOUT should be set to 1 to indicate that no
C       more simplex iterations need to be done to solve this problem.  [line 37]
C     - If the candidacy test is not met then RETURN.  Else go to step 4 to
C       write the proposal to Z.  [line 39]
C
C***********************************************************************
 2100 IF(MSTAT .NE. QU)GO TO 2110                                   30
      IPROP(6) = IPROP(6) + 1                                       31
      QPROP = QU                                                    32
      GO TO 3000                                                    33
```

```
2110  IF(MSTAT .NE. QBL)GO TO 2400                                    34
      QPROP = QO                                                      35
      IPROP(5) = IPROP(5) + 1                                         36
      KOUT = 1                                                        37
2200  IF(ITCNT .EQ. ITMA)GO TO 5000                                   38
      IF(DX .GE. -ZTCOST)GO TO 5000                                   39
2400  IF(DABS(DX - DZ) .LE. ZTCOST)GO TO 5000                         40

C*********************************************************************
C
C     Step 3a :
C
C     MAXCA     : Number of proposals that can be sent to the master in
C                 one cycle.
C     NCASUB    : Number of proposals generated by the sub-problem
C                 currently solved.
C     NCAND     : Number of proposals kept in this cycle.
C     IPROS     : Pointer to the column in Z(*,*) where the proposal can
C                 be written.
C     NPROS     : Column dimension of Z(*,*).
C     KSTR      : User's strategy. If 5 then only the last MAXCA
C                 proposals are kept. If 1 then no intermediate stage
C                 proposal can be generated. Otherwise problem gets
C                 interrupted once MAXCA proposals are generated.
C     NROWO     : Number of coupling rows.
```

```
C     X(*)            : Solution vector. First NROWO components of X(*)
C                       contains the proposal.
C
C     - Check if the current call to CHECK corresponds to the same extreme
C       point as a previous call. [lines 43-44]
C     - If KSTR is 5 and MAXCA proposals have been generated then the
C       number of proposals NCASUB generated by the current sub-problem
C       and the number of proposals NACND generated in the current cycle
C       would not be incremented. This is because the proposal being generated
C       would overwrite a proposal already existing in Z(*,*). [lines 45,49-52]
C       If KSTR is 5 and MAXCA proposals have not been generated then
C       increment NCAND and NCASUB. [45-47]
C       If KSTR is not 5 and more proposals are generated than can be
C       accomodated by Z(*,*), then write the current contents of Z(*,*) to the
C       unit 8 work file and write the proposal being generated to the first
C       column of Z(*,*). [lines 53-55]
C       If KSTR is not 5 and there is space in Z(*,*) then write the current
C       proposal to the first available column of Z(*,*). [line 49]
C     - Check if an inversion is required and if so call INVERT and set the flag
C       IRTN to 1. [lines 56-59]
C       Set KFASE to 1 and call FORMC. If the solution is infeasible then
C       since KFASE is 1, FORMC would already have set up the cost vector in
C       the buffer YA(*,5). Otherwise set KFASE to 2 and call FORMC again
C       to set up the cost vector in YA(*,6). [lines 60-69]
C       Write the proposal into Z(*,*) and the reduced cost of the proposal
```

```
C       column in (NROWO+1)st row of Z(*,*). [lines 70-72]
C       Note the iteration count for proposal generation. This is to enable the
C       first check of this step. [line 73]
C     - If KSTR ≠ 5 then mark the origin of the NCANDth proposal. [line 75]
C       Also check if the solution of problem is to be terminated or if the
C       multiple pricing cycle is to be interrupted. Set KOUT to 1 in both cases.
C       [lines 75,120]
C       Otherwise do the following. ICAND is pointer to the last proposal of
C       the previous sub-problem. Since IPROS gives the proposal position
C       beyond ICAND, mark the origin of the (ICAND+IPROS)th proposal.
C       [lines 77-78]
C
C*************************************************************************
 2450 DZ = DX                                                        41
      IF (KSTR .EQ. 5 .AND. NCUNB .EQ. MAXCA) GO TO 5000             42
      IF (ITSLCA .EQ. ITCNT) GO TO 5000                              43
      IF(MSTAT .EQ. QU .AND. ITSLCA .EQ. ITCNT-1)GOTO 5000           44
      IF (KSTR .EQ. 5 .AND. NCASUB .GE. MAXCA) GO TO 2452            45
      NCAND = NCAND + 1                                              46
      NCASUB = NCASUB + 1                                            47
      IF (ISUNB .GT. 0) NCUNB = NCUNB + 1                            48
 2452 IPROS = IPROS + 1                                              49
      IF (KSTR .NE. 5) GO TO 2453                                    50
      IF (IPROS .GT.MAXCA) IPROS = 1                                 51
      GO TO 2455                                                     52
```

```
2453  IF (IPROS .LE. NPROS) GO TO 2455                          53
      WRITE (8) Z                                               54
      IPROS = 1                                                 55
2455  IF (ITSINV .LE. INVFRQ) GO TO 2475                        56
      CALL INVERT                                               57
      ITSINV = 0                                                58
      IRTN = 1                                                  59
      KFASE = 1                                                 60
      JSTAT = MSTAT                                             61
      CALL FORMC                                                62
      IF (MSTAT .EQ. QI) GO TO 2475                             63
2460  KFASE = 2                                                 64
      IF (JSTAT .EQ. QBL .OR. JSTAT .EQ. QU) MSTAT = JSTAT      65
      DOB = 0                                                   66
      DO 2470 I = 1,NROWO                                       67
2470   DOB = DOB -X(I)*YA(I,6)                                  68
      CALL FORMC                                                69
2475  DO 2500 I=1,NROWO                                         70
2500  Z(I,IPROS) = -X(I)                                        71
      Z(NROWO+1,IPROS) = DOB + YTEMP(LSUB)                      72
      ITSLCA = ITCNT                                            73
      IF (KSTR .EQ. 5) GO TO 2550                               74
      NAME(NCAND) = LSUB                                        75
      GO TO 4000                                                76
2550  NAME(ICAND+IPROS) = LSUB                                  77
```

```
      GO TO 5000                                                        78

C*********************************************************************
C
C     Step 3b :
C
C     Same as step 4 but for two differences :
C
C     1. Call FTRAN to update the extreme column for the current basis.
C        [line 107]
C        Write the extreme ray updated in the buffer YA(*,*) to Z(*,*).
C        [lines 110-111]
C     2. If maximum limit on proposal generation has not been exceeded then
C        go to step 3a to generate an extreme point proposal. [lines 115-119]
C
C*********************************************************************
 3000 KOUT = 1                                                          79
      IF (ISUNB .GT. 0) NCUNB = NCUNB + 1                               80
      IF (KSTR .EQ. 5 .AND. NCASUB .GE.MAXCA) GO TO 3010                81
      NCAND = NCAND + 1                                                 82
      NCASUB = NCASUB + 1                                               83
 3010 IPROS = IPROS + 1                                                 84
      IF (KSTR .NE. 5) GO TO 3015                                       85
      IF (IPROS .GT. MAXCA) IPROS = 1                                   86
      GO TO 3050                                                        87
```

```
3015  IF (IPROS .LE. NPROS) GO TO 3050                          88
      WRITE (8) Z                                               89
      IPROS = 1                                                 90
3050  IF(ITSINV .LE. INVFRQ) GO TO 3100                         91
      CALL INVERT                                               92
      ITSINV = 0                                                93
      IRTN = 1                                                  94
      KFASE = 1                                                 95
      CALL FORMC                                                96
      IF (MSTAT .NE. QI) GO TO 3060                             97
      WRITE (6,6666) LSUB                                       98
6666  FORMAT('0LOSS OF FEASIBILITY IN SUBPROBLEM ',I4)          99
      STOP                                                      100
3060  KFASE = 2                                                 101
      DOB = 0                                                   102
      DO 3080 I = 1,NROWO                                       103
3080   DOB = DOB - X(I)*YA(I,6)                                 104
      CALL FORMC                                                105
      MSTAT = QU                                                106
      J = JCOLP(KVEC)                                           107
      CALL UNPACK(J,KVEC)                                       108
      CALL FTRAN(1,KVEC)                                        109
3100  DO 3200 I = 1,NROWO                                       110
3200   Z(I,IPROS) = YA(I,KVEC)                                  111
      Z(NROWO+1,IPROS) = DCMIN(KVEC)                            112
```

```
      IF (KSTR .EQ. 5) GO TO 3250                              113
      NAME(NCAND) = -LSUB                                      114
      IF(NCAND .LT. MAXNCA)GO TO 2200                          115
      GO TO 5000                                               116
3250  NAME(ICAND+IPROS) = -LSUB                                117
      IF (NCASUB .LE. NC(KKK)) GO TO 2200                      118
      GO TO 5000                                               119
4000  IF(NCAND .GE. MAXNCA .OR. NCASUB .GE. NC(KKK))           120
      1KOUT = 1
5000  RETURN                                                   121
      END                                                      122
```

3.4. Subroutine CHSOL :

3.4.1. Major Task :

Helps decide if a re-inversion is necessary.

3.4.2. Input :

1. The rhs **b** of the LP in the buffer YA(*,3).
2. The array X(*) containing $\mathbf{x}+\partial\mathbf{x}$ of (3.10).

3.4.3. Output :

1. The condition number of the basis **B** if executing step 1 of D-W phase 1.
2. The infinity norm of $\partial\mathbf{b}$ or $\partial\mathbf{x}$ (cf. 3.11 and 3.12).

3.4.4. Called By :

NORMAL.

3.4.5. Calling :

FTRAN.

3.4.6. Description :

Let **x** and **x+∂x** be the true and the computed solutions of the system

$$\mathbf{Bx} = \mathbf{b} \tag{3.9}$$

where **B** and **b** are a basis and the rhs vector of the given LP respectively. We then have

$$\mathbf{B}(\mathbf{x}+\partial\mathbf{x}) = \mathbf{b}+\partial\mathbf{b} \tag{3.10}$$

$$\partial\mathbf{x} = \mathbf{B}^{-1}\partial\mathbf{b} \tag{3.11}$$

where **∂b** is known as the residual rhs and is given by

$$\partial\mathbf{b} = \mathbf{B}(\mathbf{x}+\partial\mathbf{x}) - \mathbf{b} \tag{3.12}$$

The condition number K of the basis **B** is given by

$$K = K1/(MN) \tag{3.13}$$

where

$$M = (1 - \| \partial\mathbf{b}\mathbf{B}^{-1} \|)^{-1} \tag{3.14}$$

$$N = \| \partial\mathbf{b} \| / \| \mathbf{b} \| + \| \partial\mathbf{B} \| / \| \mathbf{B} \| \tag{3.15}$$

$$K1 = \| \partial\mathbf{x} \| / \| \mathbf{x} \| \tag{3.16}$$

Since **∂b** and **∂B** are small M is 1 and N is small. Therefore, the approximate condition number is determined by first computing K1 and then

multiplying it by a large number.

CHSOL therefore, has the following steps.

3.4.7. Steps :

1. Compute $\| \mathbf{x} \|$. Use (3.12) to compute $\partial \mathbf{b}$.

2. Compute $\| \partial \mathbf{b} \|$.

3. Compute $\| \partial \mathbf{x} \|$ and the approximate condition number if necessary.

Note : By default the norm that is used is the infinity norm.

3.4.8. Detailed Comments :

```
C***********************************************************************
C
C     Step 1 :
C
C     X(*)        : Contains the solution vector x.
C     XMAX        : At the end of step 1 it gives the ||x||.
C     NROW        : Number of rows in the problem under consideration.
C     JH(*)       : Indicates the column that is basic in the *th row.
C     IV          : Indicates the basic column of the row being considered.
C     YA(*,3)     : Before step 1 contains the rhs and at the end of step 1
C                   it has the residual rhs.
C     A(*)        : Array containing the data of the problem.
C     IA(*)       : Array indicating the row index in the matrix of each
```

```
C                    element in the array A(*).
C
C     For each row i do :
C
C          1. Obtain the basic column of the basic column of the i^th row
C             in IV.
C          2. If the i^th row is non-binding then set XMAX <-- |X(i)| if
C             XMAX  is less than the absolute value of X(i).  [lines 3-6].
C          3. Subtract from the rhs (YA(*,3)) the contribution due to the
C             IV^th column of the matrix A.  This is given by the formula :
C             ∂b <--- ∂b - a_IV x_IV.  [lines 8-12].
C
C     End For
C
C*********************************************************************
      XMAX = 0.                                                    1
      DO 1000 J = 1,NROW                                           2
      IV = JH(J)                                                   3
      IF (IV .GT. NROW) GO TO 90                                   4
      IF (ISTYPE(IV) .EQ. 0) GO TO 100                             5
   90 IF (DABS(X(J)) .GT. XMAX) XMAX = DABS(X(J))                  6
  100 IF (IV .EQ. 0) GO TO 1000                                    7
      LL = LA(IV)                                                  8
      KK = LA(IV+1) - 1                                            9
      DO 700 I = LL,KK                                             10
```

```
      IR = IA(I)                                                        11
      YA(IR,3) = YA(IR,3) - A(I)*X(J)                                   12
  700 CONTINUE                                                          13
 1000 CONTINUE                                                          14

C***********************************************************************
C
C     Step 2 :
C
C     YA(*,3)        : Contains the residual rhs ∂b. If the program is
C                      executing INDATA (step 1 of D-W phase 1) then it
C                      would contain ∂x as given by (3.11).
C     ERMAX          : At the end of step 2 it contains ||∂b||.
C
C     1. For each row i set ERMAX to max{ERMAX,|YA(i,3)|}. [lines 15-18]
C     2. Call FTRAN to compute ∂x = B^-1 ∂b. [lines 19-20].
C
C***********************************************************************
      ERMAX = 0.                                                        15
      DO 2000 I = 1,NROW                                                16
      IF (DABS(YA(I,3)) .GT. ERMAX) ERMAX = DABS(YA(I,3))               17
 2000 CONTINUE                                                          18
      IF(MAST .GE. 1)GO TO 10000                                        19
      CALL FTRAN(1,3)                                                   20
```

```
C*****************************************************************
C
C     Step 3 :
C
C     NROW          : Number of rows in the problem.
C     EEMAX         : At the end of step 3 it contains the maximum row error
C     XMAX          : Conains ||x||.
C     YA(*,3)       : At the end of step 2 it contains ∂x.
C     COND          : At the end of step 3 it contains the approximate
C                     condition number.
C     JH(*)         : Indicates the basic column of the *th row.
C     IV            : Indicates the basic column of the row being considered.
C     ISTYPE(*)     : Indicates the type (binding etc..) of the IVth row.
C
C     For each row i do :
C
C           1. Obtain the basic variable of the ith row. [line 23]
C           2. If it is binding then set EEMAX <-- max{EEMAX,YA(i,3)}.
C              [lines 24-26].
C
C     End For.
C     Compute the approximate condition number. [line 28]
C
C*****************************************************************
      EEMAX = 0.                                                    21
```

```
      DO 3000 I = 1,NROW                                              22
      IV = JH(I)                                                      23
      IF (IV .GT. NROW) GO TO 2500                                    24
      IF (ISTYPE(IV) .EQ. 0) GO TO 3000                               25
 2500 IF (DABS(YA(I,3)) .GT. EEMAX) EEMAX = DABS(YA(I,3))             26
 3000 CONTINUE                                                        27
      COND = (EEMAX*2**24)/XMAX                                       28
      IF(ERMAX .LT. ZTOLZE)GO TO 10000                                29
      WRITE(6,8000)ERMAX,COND                                         30
 8000 FORMAT(21H0MAXIMUM ROW ERROR = ,D14.5/36H
     1APPROX CONDITION NUMBER OF BASIS = ,D14.5     )                 31
10000 RETURN                                                          32
      END                                                             33
```

3.5. Subroutine CHUZR :

3.5.1. Major Task :

To find the pivot row's type or to find the maximum (max) pivot row from among those which gives the same best change in the objective value.

3.5.2. Inputs :

1. The buffer YA(*,KVEC) which contains the candidate column stored in one of its columns **or**
2. The flag (JENTRY) set to 2 to find the maximum pivot row for the pivot column. In this case the following inputs are given.
    a. The pivot column in the buffer YA(*,KVEC).
    b. The variable DE giving the allowable increase in the entering variable corresponding to the pivot column.
    c. The variable NTYPE giving the ratio type (cf. § 3.5.5.) for the pivot column.

3.5.3. Outputs :

In not flagged (JENTRY $\neq$ 2) then

- NTYPE gives the ratio type
- DE gives the allowable increase in the entering variable if the variable corresponding to the candidate column enters.

- DP gives the change in the objective value if the varaible corresponding to the candidate column enters.

Else

- IROWP gives the maximum pivot row.

End if

3.5.4. Called By :

NORMAL.

Description :

If CHUZR is not called with a flag then the leaving variable is decided based on two broad objectives :

1. To drive a basic equality logical out of the basis at the first opportunity without creating any new infeasibilities.
2. In the absence of basic equality logicals, to remove as many infeasibilities as possible if there are any infeasibilities.

Since the rhs is not necessarily positive, some changes as illustrated in the example below will have to be made to the normal RSM ratio test.

x2					+4x1	=	+36	(3.17)
	x3				+1x1	=	+24	(3.18)
		x4			-2x1	=	-36	(3.19)
			x5		-3x1	=	-24	(3.20)
				a1	-3x1	=	-36	(3.21)
					a2 -9x1	=	-36	(3.22)

Here a1and a2 are equality logicals, x2 to x5 are variables that are not equality logicals and x1 is the entering variable. The following observations can be made about this example.

1. Unlike the normal RSM ratio test (3.19)-(3.22) cannot be ignored. In fact in keeping with the first objective of driving the artificial out of the basis, the leaving variable is a2. Note however, that a1 could not be driven out of the basis since this would make x2 infeasible. Also note that the leaving variable a2 corresponds to the min ratio of (3.21) and (3.22).
2. If the coefficient of x1 in (3.22) is -1 instead of -9 then no equality logical can be driven out of the basis without creating new infeasibilities and in this case the leaving variable is x2. Note that the leaving variable corresponds to the minimum (sometimes called simply min) ratio of (3.17) and (3.18).
3. If eqns (3.17)-(3.18) and (3.21)-(3.22) were not present then, the leaving variable would be x4 in keeping with the second objetive of removing as many infeasibilities. Note that the leaving variable corresponds to the maximum (sometimes also called simply max) of (3.19)-(3.20).

The above example suggests the need for classifying rows/ratios into three types:

Type 1 : If both pivot and rhs of a row are both greater than 0 then we call such a row, type 1. Its ratio is called type 1 ratio. The type 1 ratio should be a min ratio just like the normal RSM ratio test.

Type 2 : If for a row the pivot and rhs are both lesser than 0 and the basic variable for the row is not an equality logical then we call such a row, type 2. Its ratio is called type 2 ratio. The ratio for a type 2 row should be a max ratio due to the second objective of removing as many infeasibilities.

Type 3 : If both the pivot and rhs of arow are lesser than 0 and the basic variable for the row is an equality logical then we call such a row, type 3. The ratio for such a row is called type 3 ratio. The ratio for a type 3 row should be min ratio due to the first objective.

Type ignored : If the pivot and rhs of a row are of opposite sign then the basic variable for such a row would not be considered to leave the basis since no bound is provided by such a row.

These are summarized below.

RHS	Pivot	Variable Type	Action	Type	Ratio Name
+	+	Any type	Choose Min	1	DRMIN
+	-	Any type	Ignore	--	--
-	+	Any type	Ignore	--	--
-	-	Except equality logical	Choose Max	2	DRMAX
-	-	Equality logical	Choose Min	3	DRART

Thus, at the end of one pass through all the rows there would be in general

be three ratios DRMIN, DRMAX and DRART correponding to the three types. The type of the pivot row is then decided by :

1. If there are no rows under type 1 or type 3 then the pivot row is of type 2 and is decided by DRMAX due to the second objective.
2. If there are rows under type or type 3 then the pivot row is of type 1 or 3 and is decided by min{DRMIN,DRART}.
3. If there are no rows under types 1, 2 or 3 then all rows have rhs and pivots with opposite signs and the problem is therefore unbounded.

As mentioned earlier, CHUZR is called to find the max pivot row (when JENTRY = 2) from among those pivots which provide the same best change in the objective value. To find the max pivot row given the inputs mentioned above a scan is done through all the rows of the pivot column having the same type as indicated by NTYPE. The value of the max pivot (MAXPIV) is initialized to zero and then a typical scan would be as explained below. Consider the i^{th} having the same row type as NTYPE. If the ratio of the row does not exceed DE, the max pivot row is set to i provided the value of its pivot element exceeds MAXPIV. MAXPIV is then set to the value of this pivot. Otherwise, the next row is taken up. For example, assume that in (3.21) the coefficient of x1 is -6 and the rhs is -24. If the pivot column is the column corresponding to x1 then the ratio is of type 3. This would have been decided during the first call to CHUZR from NORMAL. When CHUZR is called again with JENTRY set to 2, the pivot row is given by (3.21), since the coefficient of x1 in (3.21) is higher than the coefficient of x1 in (3.22).

CHUZR therefore has the following steps

3.5.6. Steps :

If not called with a flag (JENTRY ≠ 2) then

1. Compute the type of the pivot row for the given candidate column.

4. Compute the change in the objective value.

Else

2. Find the best eligible pivot

3. Check for negative right hand side (rhs) element and make appropriate changes.

End if

3.5.7. Detailed Comments :

```
C*****************************************************************
C
C     Step 1 :
C
C     YA(*,*)      : Buffer containing a pool of candidate columns.
C     KVEC         : Column of YA that has a candidate column currently
C                    being considered.
C     JENTRY       : Flag.  If 2 indicates that the maximum pivot is to be
C                    found.  If not 2 indicates that the type of the pivot row
C                    is to be found.
C     X(*)         : Array containing the solution vector.
C     NROW         : Number of rows.
```

C

C 1. If JENTRY is not 2 then for each row i, do:

C - If there is a zero in the pivot position of the i^{th} row then go to

C the next row as such a row does not provide a bound.

C [line 7-9].

C Else do

C - Set IVEC to the basic column for the i^{th} row. [line 10]

C - If row i is non-binding row go to the next row since the basic

C variable of a non-binding row is never a candidate to leave the

C basis. [line 11-12]

C If binding then

C

C *Case 1* : (Pivot and rhs are > 0)

C YA(I,KVEC) > 0 implies pivot > 0. [line 13]

C X(I) > 0 implies rhs is > 0. [line16]

C Since this is the case of a type 1 ratio, compute the ratio

C DTHETA and update the type 1 min ratio DRMIN.

C [lines 17-20]

C

C *Case 2* : (Pivot and rhs < 0 and inequality logical basic)

C YA(I,KVEC) < 0 implies pivot is < 0 [line 14]

C IVEC < NROW implies equality logical and

C ISTYPE(IVEC) ≠ -1 implies inequality logical.

C [line 21-22].

C X(I) < 0 implies rhs is < 0. [line 23]

```
C                    Compute the ratio DTHETA and since this is the case of
C                    a type 2 ratio update the type 2 max ratio DRMAX.
C                    [line 24-28].
C
C                    Case 3 : (Pivot and rhs < 0 and equality logical is basic)
C                    ISTYPE(IVEC) = -1 implies equality logical. [line 22].
C                    Compute the ratio and since this is the case of a type 3
C                    ratio and update the type 3 min ratio DRART.[line30-33]
C               End If
C          End For
C          2. The type of the pivot row is decided as follows.
C
C               Case 1 : (No type 1 or type 2 or type 3 rows)
C               DRMIN, DRMAX, DRART being at their initialized value
C               implies this. Note that even though these values were initialized
C                to 1.0D30 and -1.0D30 they can only be tested for 1.0D29 and
C               -1.0D29due to the precision limit on the computer. [lines 34,42]
C               Therefore, all rows have pivots and rhs with opposite signs. Go
C               to step 2 to set the status as unbounded. [line 42]
C
C               Case 2 : (No type 1 or type 3 rows)
C               Line 34 tests for this case. The ratio type is 2 and DRMAX gives
C               the row corresponding to this type. [lines 43-46]
C
C               Case 3 : (Type 1 or type 3 rows exist)
```

```
C           Line 34 tests for this case. Pivot row type is decided according
C           to the min{DRMIN,DRART}. [lines 35-40]
C           Go to step 2 to find the change in the objective value. [line 41]
C
C*****************************************************************
```

```
      IF(JENTRY .EQ. 2)GO TO 1400                                    1
      DRMIN = 1.D30                                                  2
      IROWP = 0                                                      3
      DRMAX = -1.D30                                                 4
      DRART = 1.D30                                                  5
      DO 1000 I = 1,NROW                                             6
      IF(DABS(YA(I,KVEC)) .GT. ZTOLZE)GO TO 50                       7
      YA(I,KVEC) = 0.0                                               8
      GO TO 1000                                                     9
   50 IVEC = JH(I)                                                   10
      IF ( IVEC .GT. NROW) GO TO 100                                 11
      IF(ISTYPE(IVEC) .EQ. 0)GO TO 1000                              12
  100 IF(YA(I,KVEC) .GT. ZTOLPV)GO TO 150                            13
      IF(YA(I,KVEC) .LT.-ZTOLPV)GO TO 500                            14
      GO TO 1000                                                     15
  150 IF(X(I) .LT. -ZTOLRJ)GO TO 1000                                16
      DTHETA = (X(I) + ZTOLRJ)/YA(I,KVEC)                            17
      IF(DTHETA .GE. DRMIN)GO TO 1000                                18
      DRMIN = DTHETA                                                 19
```

```
      GO TO 1000                                                    20
  500 IF(IVEC .GT. NROW)GO TO 710                                   21
      IF(ISTYPE(IVEC) .EQ. -1)GO TO 800                             22
  710 IF(X(I) .GE. 0)GO TO 1000                                     23
      DTHETA = X(I) / YA(I,KVEC)                                    24
      IF (DTHETA .LE. DRMAX) GO TO 1000                             25
      DRMAX = DTHETA                                                26
      IROWP2 = I                                                    27
      GO TO 1000                                                    28
  800 IF(X(I) .GT. ZTOLRJ)GO TO 1000                                29
      DTHETA = (X(I) - ZTOLRJ)/YA(I,KVEC)                           30
      IF(DTHETA .GE. DRART)GO TO 1000                               31
      DRART = DTHETA                                                32
 1000 CONTINUE                                                      33
C
      IF(DRMIN .GT. 1.D29 .AND. DRART .GT. 1.D29)GO TO 1300         34
      IF(DRART .LT. DRMIN)GO TO 1200                                35
      NTYPE = 1                                                     36
      DE = DRMIN                                                    37
      GO TO 9900                                                    38
 1200 DE = DRART                                                    39
      NTYPE = 3                                                     40
      GO TO 9900                                                    41
 1300 IF(DRMAX .LE. -1.D29)GO TO 9000                               42
      DE = DRMAX                                                    43
```

```
      NTYPE = 2                                                    44
      IROWP = IROWP2                                               45
      GO TO 9900                                                   46


C*********************************************************************
C
C     Step 2 :
C
C     YA(*,KVEC)    : Contains the pivot column.
C     X(*)          : Array containing the solution vector.
C     DE            : Starts with the allowable increase in the entering
C                     variable taking all rows into consideration.  As each
C                     row is considered during the second call to CHUZR
C                     with JENTRY flagged to 2, DE gets updated to indicate
C                     the allowable increase in the entering variable at that
C                     point.
C     DRO           : Initialized to allowable increase in the entering variable.
C     DTHETA        : Ratio for the row being considered.
C     NTYPE         : Pivot row's type set during the previous call with
C                     JENTRY ≠ 2
C
C     For each row i row do :
C
C           Case 1  : (Pivot < 0)
```

```
C              - YA(I,KVEC) < 0 implies this case. [line 53]
C              - If NTYPE ≠ 3 then NTYPE should be 1 and this implies that
C                pivot of the min ratio's row was > 0. Therefore, the i^th row
C                could not be the one that set the pivot row's type. Go to the
C                next row. Else if NTYPE is 3 then proceed. [line 62]
C              - If the basic variable is not an equality logical or if the rhs is > 0
C                then the i^th row cannot be of type 3. Therefore go to the next
C                row. Else proceed. [lines 63-65]
C              - Find the ratio DTHETA for the i^th row. If DTHETA > DRO
C                then the i^th row did not give the min ratio. So go to the next
C                row. Else update the value of the maximum pivot (MAXPIV)
C                and IROWP to i to indicate that the so far the i^th row has yielded
C                the maximum pivot. [lines 66-71].
C
C              Case 2 : (Pivot > 0)
C              This is the complement of case 1. [lines 54-60,70-71]
C
C       End For
C
C*****************************************************************

 1400  MAXPIV = 0.0                                          47
       DRO = DE                                             48
       DO 2000 I=1,NROW                                     49
       IVEC = JH(I)                                         50
```

```
      IF(IVEC .GT. NROW)GO TO 1450                          51
      IF(ISTYPE(IVEC) .EQ. 0)GO TO 2000                     52
1450  IF(YA(I,KVEC) .LE. ZTOLPV)GO TO 1500                  53
      IF(NTYPE .NE. 1)GO TO 2000                            54
      IF(X(I) .LT. -ZTOLRJ)GO TO 2000                       55
      DTHETA = X(I)/YA(I,KVEC)                              56
      IF(DTHETA .GT. DRO)GO TO 2000                         57
      IF(YA(I,KVEC) .LT. MAXPIV)GO TO 2000                  58
      MAXPIV = YA(I,KVEC)                                   59
      GO TO 1600                                            60
1500  IF(YA(I,KVEC) .GE. -ZTOLPV)GO TO 2000                 61
      IF(NTYPE .NE. 3)GO TO 2000                            62
      IF(IVEC .GT. NROW)GO TO 2000                          63
      IF(ISTYPE(IVEC) .NE. -1)GO TO 2000                    64
      IF(X(I) .GT. ZTOLRJ)GO TO 2000                        65
      DTHETA = X(I)/YA(I,KVEC)                              66
      IF(DTHETA .GT. DRO)GO TO 2000                         67
      IF(-YA(I,KVEC) .LT. MAXPIV)GO TO 2000                 68
      MAXPIV = - YA(I,KVEC)                                 69
1600  DE = DTHETA                                           70
      IROWP = I                                             71
2000  CONTINUE                                              72
```

```
C*****************************************************************
C
C     Step 3 :
C
C     If pivot row is of type 3 then the rhs of the pivot row must be ≤ 0 failing
C     which it must be set to 0.  Likewise if the pivot row is of type 1 then the
C     rhs of the pivot row must be ≥ 0 failing which it must be set to 0.
C     [lines 73-81]
C
C     Step 4 :
C
C     Set MSTAT = 'QU' if the problem is unbounded.  [line 82]
C     Compute the change DY in the objective value by multiplying the reduced
C     cost DCMIN(KVEC) of the entering column with the allowable increase
C     DE in the entering variable.  [line 84]
C
*****************************************************************
      IF(NTYPE .EQ. 3)GO TO 3000                                    73
      IF(X(IROWP) .GE. 0)GO TO 10000                                74
      DE = 0                                                        75
      X(IROWP) = 0                                                  76
      GO TO 10000                                                   77
3000  IF(X(IROWP) .LT. 0)GO TO 10000                                78
      DE = 0                                                        79
```

```
      X(IROWP) = 0                                                    80
      GO TO 10000                                                     81
 9000 MSTAT = QU                                                      83
      DE = 1.E10                                                      84
 9900 DY = DCMIN(KVEC)*DE                                             85
10000 RETURN                                                          86
      END                                                             87
```

3.6. Subroutine FORMC

3.6.1. Major Task :

Form the basic cost vector for each simplex iteration.

3.6.2. Inputs :

The solution vector in the array X(*).

3.6.3. Outputs :

1. Forms the basic cost vector into the appropriate column of the buffer named YA(*,*).
2. Also sets variables to denote the feasibility of the current solution.

3.6.4. Called by :

FTRAN, NORMAL.

3.6.5. Description :

FORMC sets up the cost vector in a buffer named YA(*,*). To set up the cost vector two things have to be decided. The first is the column of YA(*,*) in which to form the cost vector and the second is the contents of the cost vector.

The column of the buffer YA in which to store the cost vector depends on a variable called KFASE. KFASE is normally set to 2 and in this case the cost vector is set up in YA(*,6). When the solution of a sub-problem becomes infeasible, KFASE is set to 1 and in this case the cost vector is set up in YA(*,5).

The contents of the buffer depends on the status of the problem and is given as follows.

Case 1 : (Feasible solution and executing step 1 of D-W phase 1) Reference to section 2.6.1 shows that the cost vector for feasible solution should be $(1,-\pi_0^k,0)$. Since π_0^k is a **0** vector to start with, (k = 0) the cost vector is (1,**0, 0**).

Case 2 : (Feasible solution and executing D-W phase 3) In this case the objective function is fixed (cf. section 2.5.3). Therefore, the cost vector for this case is (1,**0, 0**) which is the cost vector for an LP in RSM phase 2.

Case 3 : (Feasible and executing either step 2 of D-W phase 1 or D-W phase 2)

Case 3.1 : (Solving Master problem) In this case the objective function is fixed. Therefore, the cost vector for this case is (1,**0**,**0**)

Case 3.2 : (Solving sub-problem) From section 2.6.1 it can be seen that the cost vector is $(1,-\pi_0^k,0)$. Note that the 1 and $-\pi_0^k$ are stored in the sixth column of YA(*,*) by the subroutine MASTER before FORMC is called.

Case 4 : (Infeasible solution and any phase) Cost vector corresponds to infeasibilities as discussed in section 2.2.3.

FORMC therefore comprises the following steps:

3.6.6. Steps :

1. Set the appropriate buffer vector and determine if step 2 should be performed. If not go to 3.
2. Checks the solution vector for infeasibilities and produces the cost vector accordingly.
3. Find the cost vector for feasible case.

3.6.7. Detailed Comments :

```
C*****************************************************************
C       Step 1 :
C
C       NROWO       : Number of coupling rows.
C       MAST        : Stage of the D-W algorithm that is currently being
C                     executed.  If 0 then executing step 1 of D-W phase 1.
C                     If 1 then executing step 2 of D-W phase 1 or D-W
C                     phase 2.  If 2 then executing the D-W  phase 3.
C       KFASE       : Normally set to 2.  Set to 1 if solving a sub-problem in
C                     D-W phase 2 and the sub-problem's solution is not
C                     feasible.
C       LSUB        : Index of the problem being treated.  Zero indicates the
C                     master problem.
```

```
C     IFFEZ          : Feasibility status of the problem. Feasible if 1 and not
C                      feasible if 0.
C     YA(*,*)        : Buffer in which the cost vector is set up.
C
C     - Initialize IFFEZ to 1 indicating that the status of the problem is feasible.
C       line 1]
C       Initialize KVEC to  6 indicating that the cost vector should be formed in
C       YA(*,6). [line 2].
C
C     - Case 1 : (MAST = 1)
C           Case 1.1  : (Solving sub-problem) In this case that the value of
C           KFASE can indicate the status (feasible or infeasible) of the
C           problem's solution.
C           - If KFASE = 1 then
C                 - Set KVEC to 5 to set up the cost vector in YA(*,5).
C                   [line 9]
C                 -Set YA(*,5) to the zero vector. [lines 11-13]
C                 - Proceed to step 2 to check if all infeasibilities have been
C                   removed and if so set the objective row's position in
C                   YA(*,*) to 1. Reference to subroutine NORMAL will
C                   show that in this case FORMC would be called again
C                   with KFASE set to 2. On the other hand if all
C                   infeasibilities have not been removed then step 2 would
C                   set the cost vector corresponding to infeasibilities.
C             Else if KFASE = 2 then
```

```
C                    - Reference to subroutine MASTER will show that
C                      (1,-π_0^k) will already have been stored in the first
C                      NROWO positions of YA(*,6). Since the required cost
C                      vector is (1,-π_0^k,0) set each of the elements in
C                      YA(NROWO,6) to YA(NROW,6) to zero. [lines 5-9]
C                End if
C
C              Case 1.2 : (Solving master problem)
C              - Zero YA(*,6). [lines 11-13]
C              - Go to step 2 to check for any infeasibilities failing which step 3
C                would set (1,0) as the cost vector.
C       Case 2 : (MAST = 0 or 2)
C       Same as case 1.2.
C*****************************************************************
        IFFEZ = 1                                                    1
        KVEC = 6                                                     2
        IF(MAST .NE. 1)GO TO 50                                      3
        IF(LSUB .LT. 1 .OR. KFASE .NE. 2)GO TO 25                    4
        MM = NROWO + 1                                               5
        DO 2500 I=NROWO+1                                            6
2500    YA(I,KVEC) = 0.                                              7
        GO TO 9000                                                   8
25      IF(KFASE .EQ. 1)KVEC = 5                                     9
50      MSTAT = QF                                                   10
        DO 100 I = 1,NROW                                            11
```

```
      YA(I,KVEC) = 0.                                                    12
100   CONTINUE                                                           13
      SUM = 0.                                                           14

C*****************************************************************
C     Step 2 :
C     For each row i do
C           - Set ICOL to the basic column of the i^th row. [line 16]
C           Case 1 : (non-binding row)
C           - ISTYPE(*) = 0 implies non-binding row. [line 18]
C           - Ignore and go to the next row
C
C           Case 2 : (structural or inequality logical)
C           - ICOL > NROW implies structural. [line 17]
C           - ICOL ≤ NROW and ISTYPE(*) = +1 implies inequality logic
C             [lines 17-18]
C           - If the X(i) is < 0 the store a +1.0 in YA(i,KVEC). [lines 24-25]
C           - Update the sum of infeasibilities, SUM. [line 26]
C           - Set IFFEZ to 0 indicating that the current solution is not
C             feasible. [line 27]
C
C           Case 3 : (equality logical)
C           - ICOL ≤ NROW and ISTYPE(*) = -1 implies equality logical.
C             [lines 17-18]
C           - If X(i) < 0 then YA(i,*) = +1
```

```
C               If X(i) > 0 then YA(i,*) = -1.  This is done because an artificial
C               has got to be brought to zero.  In either case set the sum of
C               infeasibilities, SUM and IFFEZ to  0 indicating that the current
C               solution is feasible.  [lines 19-23,27-28]
C       End For
C*********************************************************************
        DO 1000 I = 1,NROW                                          15
        ICOL = JH(I)                                                16
        IF (ICOL .GT. NROW) GO TO 500                               17
        IF (ISTYPE(ICOL)) 200,1000,500                              18
 200    IF (DABS(X(I)) .LE. ZTOLRJ) GO TO 1000                      19
        IF(X(I) .LT. 0.) YA(I,KVEC) = +1.                           20
        IF(X(I) .GT. 0.) YA(I,KVEC) = -1.                           21
        SUM = SUM + DABS(X(I))                                      22
        GO TO 510                                                   23
 500    IF(X(I) .GT. -ZTOLRJ) GO TO 1000                            24
        YA(I,KVEC) = +1.                                            25
        SUM = SUM - X(I)                                            26
 510    IFFEZ = 0                                                   27
        MSTAT = QI                                                  28
1000    CONTINUE                                                    29

C*********************************************************************
C       Step 3 :
C
```

```
C       Based on the result of step 2 there are two cases :
C
C       Case 1  :  (Feasible solution)
C       - IFFEZ = 1 implies feasible.  [line 31]
C       - Set the objective's position in YA to 1.0.  [line 32]
C
C       Case 2  :  (Not feasible)
C       - IFFEZ = 0 implies infeasible solution.  [line 31]
C       - RETURN.
C*************************************************************
        SUMINF = SUM                                          30
        IF (IFFEZ .LE. 0) GO TO 9000                          31
        YA(IOBJ,KVEC) = 1. 0                                  32
9000    RETURN                                                33
```

3.7. Subroutine FTRAN :

3.7.1. Major Task :

Performs the update $\mathbf{d_j} = \mathbf{E_k}(\ldots(\mathbf{E_2}(\mathbf{E_1 a_j})))$ where $\mathbf{a_j}$ is the original column and is $\mathbf{d_j}$ its update. Also gives the reduced cost if called with a suitable flag.

3.7.2. Inputs :

1. The eta file consisting of etas in the array A(*).
2. The column to be updated in YA(*,KVEC).
3. The type of FTRAN indicated by the parameter IPAR.

3.7.3. Outputs :

1. The updated column in YA(*,KVEC).
2. Also computes the reduced cost if the parameter IPAR is 3.

3.7.4. Called by :

NORMAL, INVERT

3.7.5. Calling :

Calls FORMC to form $\mathbf{c_b}$ if IPAR is 3 (flag to compute reduced cost).

3.7.6. Description :

FTRAN performs two functions. The first is to update a given column $\mathbf{a_j}$ to $\mathbf{d_j}$ by solving the system $\mathbf{Bd_j} = \mathbf{a_j}$. The second is to find the reduced cost if FTRAN is called with a flag.

Updating a column is done by transforming it with a set of elementary eta matrices as shown by (3.23)

$$\mathbf{d_j} = \mathbf{E_k}...\mathbf{E_1}\mathbf{a_j} \tag{3.23}$$

where

$$\mathbf{B} = \mathbf{F_1}...\mathbf{F_k} \tag{3.24}$$

and

$$\mathbf{F_i}^{-1} = \mathbf{E_i} \tag{3.25}$$

Before this transformation can be started, one should know the etas necessary for the required update and this depends on the type as indicated by IPAR. The first type arises when the column requiring the update has not been transformed by any of the etas generated thus far and this update is called full update. Therefore, in this case the transformation has to be done with all the etas generated so far. The second arises while the program is executing INVERT. The third type arises when a full update was done to the column during the previous multiple pricing pass. Therefore, for the third type of update the transformation is done only with

the etas generated during the previous pivot operation. Let $\mathbf{E_g}$ be the g^{th} eta and let the p^{th} column of $\mathbf{E_g}$ be the non-trivial column.

$$\mathbf{E_g} = [\ \ \mathbf{n_p}\ \ ] \tag{3.26}$$

where

$$\mathbf{n_p} = \begin{cases} -e_{1p}/e_{pp} & i \neq p \\ 1/e_{pp} & i = p \end{cases} \tag{3.27}$$

Without loss of generality consider the transformation by $\mathbf{E_g}$. Note that at this point the original column $\mathbf{a_j}$ has already been transformed to $\mathbf{d''_j}$ (say) by eta matrices $\mathbf{E_1},...,\mathbf{E_{g-1}}$. Therefore, the g^{th} stage of the trannsformation is :

$$\mathbf{E_g d''_j} = \begin{cases} d''_{ij} - e_{ip}d''_{pj}/e_{pp} & i \neq p \\ d''_{pj}/e_{pp} & i = p \end{cases} \tag{3.28}$$

Since the pivot element of the eta is stored first (cf. section 3.21), the row index of the first element gives the pivot row p of $\mathbf{E_g}$. Therefore, the p^{th} component (= d''_{pj}/e_{pp}) of the update can be computed easily. As can be seen from (3.28), computing each of the other elements of the update requires only one

multiplication and one addition.

As stated earlier, the reduced cost has also to be computed if FTRAN is called with the parameter IPAR set to three. To compute the reduced cost (cf. section 2.2.3)

$$\pi a_j = c_b B^{-1} a_j = c_b d_j \qquad (3.29)$$

one needs to multiply the appropriate cost vector (= c_b) to the updated column d_j just computed. This is done as follows :

Case 1 : (Solution feasible)

Case 1.1 : (Solving master problem) In this case the objective function is fixed and therefore the cost vector (cf. section 2.2.3) is (1,0,0). Hence, the reduced cost is the first element of d_j.

Case 1.2 : (Solving sub-problem in step 1 of D-W phase 1 or phase 3) Same as case 1.1.

Case 1.3 : (Solving sub-problem in D-W phase 2) The cost vector in this case is $(1,-\pi_0^k,0)$ and reference to subroutine MASTER shows that this is stored by subroutine MASTER in the sixth column of the named YA(*,*). The reduced cost is then computed by the dot product given by (3.29)

Case 2 : (Previous solution infeasible) In multiple pricing (cf. section 2.2.4) FORMC ia not called after the solution gets updated due to a basis change. Since the simplex method maintains feasibility, if the previous solution was feasible, then the solution after an update is also feasible. But if the previous solution was

infeasible then FTRAN has to call FORMC to ascertain the status of the current solution and therefore two cases would result subsequent to this FORMC call.

Case 2.1 : (Current solution still infeasible) The reduced cost is the dot product of the cost vector for infeasibilities just set up by FORMC and the updated column $\mathbf{d_j}$.

Case 2.2 : (Current solution feasible) FORMC is called again to set up the cost vector (1,0,0) for feasible state and then case 1 is executed.

FTRAN therefore comprises the following steps

3.7.7. Steps :

1. Find the etas necesary for the update.
2. Perform the update.
3. Find the reduced cost if flagged.

3.7.8. Detailed comments :

```
C*****************************************************************
C
C     Step 1 :
C
C     IPAR        : Determines the type of FTRAN to be performed. If 1
C                   then a full update is to be done. If 3 three then only the
C                   last eta is necessary.
C     NFE         : First eta for the update. If IPAR is 1 then NFE should
```

```
C                       1. If IPAR is 3 then NFE should be last eta generated.
C     NLE           : Last eta of the update and therefore NLE is the last eta
C                       generated.
C
C*****************************************************************
      CALL TICTAC(JTIM)                                          1
      GO TO (100,110,120),IPAR                                   2
  100 NFE = 1                                                    3
      NLE = NETA                                                 4
      GO TO 200                                                  5
  110 NFE = NLETA + 1                                            6
      NLE = NETA                                                 7
      GO TO 200                                                  8
  120 NFE = NETA                                                 9
      NLE = NETA                                                 10

C*****************************************************************
C
C     Step 2 :
C
C     LL            : First non-zero element of the non trivial column of the
C                     eta currently being considered.
C     KK            : Last non-zero element of the non trivial column of the
C                     eta currently being considered.
C     IA(*)         : Row position of the (*)th element of the data matrix A.
```

```
C     YA(*,*)      : Buffer one of whose columns requires the update.
C     KVEC         : The column of YA that is to be updated.
C     DY           : The pivot element of the updated column.
C     A(*)         : Has the eta file.
C
C     1. If there are no etas then RETURN. Else proceed. [line 11]
C     2. For each eta E_i do :
C
C           - Set LL and KK to the first and last position of E_i. [line 13-14]
C           - Since the pivot element of the eta is stored first the row index
C             of the first non-zero in E_i should be the pivot. Set IPIV to
C             indicate pivot row. If the pivot is close to zero then take up
C             the next eta. [line 15-16]
C           - DY is the pivot element of the updated column d_j since A(LL)
C             has the pivot element 1/e_pp of the eta and YA(IPIV,KVEC) has
C             the pivot element a_pj of the column a_j. [line 17]
C           - Set the pivot of the updated column in-place. [line 18]
C           - From (3.28) it is clear that if an eta element e_ip is zero then there
C             would be no change in the corresponding updated element
C             of d_j. So for each non-zero eta element j of E_i, find the row
C             index IR of the j^th element and update the IR^th position of
C             YA(*,*) as given by (3.28).
C
```

```
C        End For
C
C*************************************************************************
200   IF (NFE .GT. NLE) GO TO 9000                                    11
      DO 1000 IK = NFE,NLE                                            12
      LL = LE(IK)                                                     13
      KK = LE(IK+1) - 1                                               14
      IPIV = IA(LL)                                                   15
      IF(DABS(YA(IPIV,KVEC)) .LE. 1.D-15)GO TO 1000                   16
       DY = YA(IPIV,KVEC)*A(LL)                                       17
      YA(IPIV,KVEC) = DY                                              18
      IF (KK .LE. LL) GO TO 1000                                      19
       LL = LL + 1                                                    20
      DO 500 J = LL,KK                                                21
      IR = IA(J)                                                      22
      YA(IR,KVEC) = YA(IR,KVEC) - A(J)*DY                             23
 500  CONTINUE                                                        24
1000  CONTINUE                                                        25

C*************************************************************************
C
C     Step 3 :
C
C     MSTAT          : Denotes solution status. 'QF' for feasible, 'QI' for
C                      infeasible.
```

```
C     NROW      : Number of rows.
C     NROWO     : Number of coupling rows.
C     YA(*,*)   : Has the updated column whose reduced cost is needed.
C     KVEC      : The column of YA whose reduced cost is to be found.
C     KFASE     : Normally set to 2 in which case the cost vector is set
C                 up in YA(*,6). When a sub-problem is not feasible in
C                 D-W phase 2 then KFASE is set to 1 and the cost
C                 vector is set up in YA(*,5).
C     MAST      : Denotes the stage of the D-W algorithm being executed
C                 If 0 then executing step 1 of D-W phase 1. If 1 then
C                 executing step 2 of D-W phase 1 or D-W phase 2. If 2
C                 then executing D-W phase 3.
C     LSUB      : Problem index of the problem currently executing.
C                 Zero indicates MASTER problem.
C     IPAR      : If 3 then FTRAN is flagged to find the reduced cost.
C
C     1. If not flagged then RETURN. Else proceed. [line 26]
C     2. The cases cited below are the same as those given in "description"
C        Case 1 : (Solution feasible)
C
C             MSTAT = 'QF' implies feasible solution. [line 27]
C             Case 1.1 : (Solving master problem) Reduced cost is the first
C             element. [line 47]
C             Case 1.2 : (Solving sub-problem in step 1 of ph 1or D-W ph 3)
C             Same as case 1.1
```

```
C           Case 1.3 : (Sub-problem in D-W phase 2) Reduced cost is the
C           dot product of the sub-problem's D-W phase 2 cost vector in
C           YA(I,KK) and the updated column in YA(I,KVEC). [line 43-46]
C           MM in line 41 is set to NROWO since the sub-problem's phase 2
C           cost vector is zero beyond the NROWO^th position. [line 44]
C           KK is set to 6 since the phase 2 cost vector is in YA(*,6)
C
C     Case 2 : (Previous not feasible) MSTAT ≠ QF implies this. [line 27]
C     Call FORMC to ascertain current solution's status. [line 28]
C     After calling FORMC there are two cases. [line 29]
C
C           Case 2.1 : (Solution still infeasible) Reduced cost is now the
C           dot product of the cost vector for infeasibilities in YA(I,KK) and
C           the updated column YA(I,KVEC). [lines 44-46]
C           KK is set according to KFASE. [lines 31-32]
C           MM of line 44 is set to the number of rows NROW, since any of
C           the NROW positions of the solution vector X(*) can be
C           infeasible. [line 30]
C           Case 2.2 : (Solution feasible) If KFASE is 2 then the cost
C           vector is already set up in YA(*,6). Therefore, find reduced cost
C           as in case 1. [line 34]
C           If KFASE is 1 then in the previous iteration the sub-problem's
C           solution was infeasible and the cost would have been formed in
C           YA(*,5). Hence, set KFASE to 2 and call FORMC to set up the
C           cost vector in YA(*,6). Proceed to Case 1 to compute the
```

```
C                reduced cost. [lines 34-38]
C
C*****************************************************************
      IF(IPAR .LT. 3)GO TO 9000                                    26
      IF(MSTAT .EQ. QF)GO TO 1200                                  27
      CALL FORMC                                                   28
      IF(MSTAT .EQ. QF)GO TO 1100                                  29
      MM = NROW                                                    30
      KK = 6                                                       31
      IF(KFASE .EQ. 1)KK=5                                         32
      GO TO 1400                                                   33
1100  IF(KFASE .NE. 1)GO TO 1200                                   34
      KFASE = 2                                                    35
      CALL FORMC                                                   36
      DOB = 0                                                      37
      DO 1150 I=1,NROWO                                            38
1150  DOB = DOB - X(I)*YA(I,6)                                     39
1200  IF(LSUB .EQ. 0 .OR.MAST .NE. 1)GO TO 2000                    40
      MM = NROWO                                                   41
      KK = 6                                                       42
1400  DCMIN(KVEC) = 0                                              43
      DO 1500 I=1,MM                                               44
1500  DCMIN(KVEC) = DCMIN(KVEC) + YA(I,KK)*YA(I,KVEC)              45
      GO TO 9000                                                   46
2000  DCMIN(KVEC) = YA(IOBJ,KVEC)                                  47
```

```
900   CALL TICTAC(ITINV)                          48
      JTINV = JTINV + ITINV - JTIM                49
      RETURN                                      50
      END                                         51
```

3.8. Subroutine INDATA :

3.8.1. Major Task :

Executes Step 1 of D-W phase 1.

3.8.2. Called By :

Main Program

3.8.3. Calling :

1. INPUT to read the problem.
2. NORMAL to solve sub-problems.
3. VECTOR to write to and read from the direct access device (DAD).
4. PACK to pack the proposals.

3.8.4. Description :

INDATA executes step 1 of D-W phase 1. This is done in three stages. The first is the set-up stage the second is the proposal generation stage and the third stage sets up the master problem for step 2 of D-W phase 1.

To set up, INDATA has to first read the control parameters. Since DECOMP is meant for solving very large problems, all the data cannot be accomodated in core at the same time. Therefore, a direct access device (DAD)

consisting of a fixed number of records of fixed length is defined to store the data relevant to each problem. The organisation of DAD is shown in figure (3.1). The number of records required to write the problem's names (row and column names) and data is determined. Using this, the elements of a two dimensional pointer array NOLA(*,*) (cf. Appendix A) are computed. This concludes the set-up stage.

The second stage consists of execution of the following LMAXP1 (= number of sub-problems plus one) times.

- A problem (master or sub-problem) is read from the input file.

Case 1 : (Don't generate proposals)

- This case is met if the problem read is the master problem or if the user desires that no proposal be generated in step 1 of D-W phase 1.
- Since the master problem is not solved at this point, the problem's data is written to DAD.
- Each of the coupling rows is marked as non-binding.
- The number of rows (NROW) is initialized to the number of coupling rows (NROWO). This is done since the sub-problem structure (cf. § 2.6.1) includes the coupling rows as non-binding rows.

Case 2 : (Generate proposals)

- This is mutually exclusive of case 1.
- NORMAL is called to solve the sub-problem. If it is infeasible then the whole problem is infeasible. Otherwise, a column, which is the proposal modulo a minus sign, is generated and it resides in the first NROWO elements of the solution array X(*). This is copied to the proposal buffer Z(*,*).

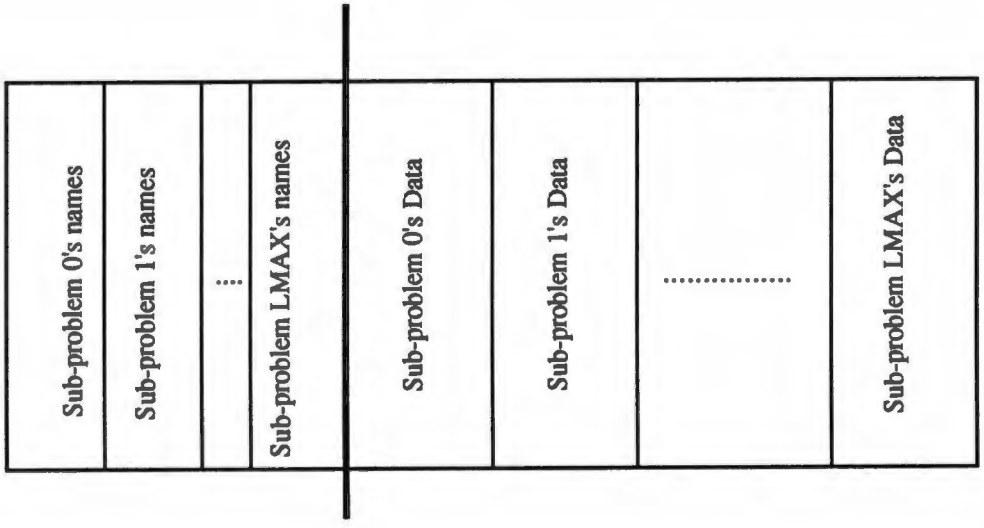


Figure 3.1 : Organization of the direct access file.

- The problem's data is written to DAD so as to free the space in core for reading and solving the next sub-problem.
- The number of rows (NROW) is set to the number of coupling rows (NROWO) for the same reason as cited in case 1 above.
- The names of the master problem are read from DAD. This is done to check the existence of row names given for data in $\mathbf{A1_r}$ (cf § 2.5.3).

The last stage is to read the master problem's data from DAD. PACK is called to incorporate the proposals into the data array of the master problem and now step 2 of D-W phase 1 can be executed.

INDATA therefore, comprises the following steps

3.8.5. Steps :

1. Set up
2. Solve all problems and generate proposals.
3. Pack proposals.

3.8.6. Detailed Comments :

```
C***********************************************************************
C
C     Step 1 :
C
C     A(*)          : Array containing the matrix elements and the eta file of
C                     a problem.
```

```
C     LPOINR    : Indicates where in A(*) the eta file of a problem begins.
C     NEMAX     : Maximum number of non-zero elements allowed in the
C                 array A(*) of the  problem.  Set to 10,000.
C     NRMAX     : Maximum number of rows in a problem.
C     NLAMAX    : Maximum number of columns in a problem.
C     NTMAX     : Maximum number of eta vectors in the eta file of a
C                 problem.
C     LTRAC     : Record length in bytes of a record in the direct access
C                 file.
C     MAXTRK    : Maximum number of records defined in the direct
C                 access file.
C     NTRAC2    : Number of records of the direct access file required to
C                 store the names (row and column) of any sub-problem
C                 of the given problem.
C     NTRAC1    : Number of records of the direct access file required to
C                 store all data relevant (other than names) to any
C                 sub-problem of the given problem.
C     ITRK      : Total number of records of the direct access file required
C                 by all sub-problems (including master) to store data
C                 relevant to it.
C     Z(*,*)    : Proposal buffer.
C     NPROS     : Column dimension of Z(*,*).
C     MAXCA     : Maximum number of proposals sent to the master by a
C                 particular problem in a given cycle.
C     MAXNCA    : Control parameter which gives the maximum number of
```

```
C                    proposals that can be sent to the master in a cycle.
C                    Default is 100 and maximum is 150.
C     KMULT          : Maximum number of columns to be priced for multiple
C                    pricing.
C
C     1. Define the unit 9 direct access file. [line 6].
C     2. Initialize problem parameters. [lines 7-37].
C     3. If parameters are supplied by the user then read them. [lines 40-42].
C     4. If the parameters are outside the allowable range then set them to the
C        appropriate values. [lines 43-45].
C     5. Compute NTRAC1 and NTRAC2. [lines 47-48].
C     6. Based on this, compute for each sub-problem (including master), the
C        record number of the direct access file at which the data relevant to the
C        sub-problem is to be written. Store this information in NOLA(*,*).
C        Also compute ITRK which indicates the total number of records
C        required for all sub-problems. [lines 49-60].
C     7. If ITRK exceeds the maximum number of tracks defined for the direct
C        access file (MAXTRK) then STOP. Otherwise proceed to step 2.
C        [lines 68-71].
C
C*****************************************************************
      NAMELIST/PARAM/,LPOINR,LMAX,IOBJ,USERHS,KINP,          1
     1KSTR,MAXNCA,MAXCA,NCANFR,INVFRQ,INVFMA,NPH,
     2KMULT,ITOL,ICPI,IRST,ICLAST,ISAVE,ISOL,RPERC,RSTOP
```

```
      NAMELIST /TOL/ ZTOLZE,ZTOLPV,ZTCOST,ZTOLRJ,                    2
     1ZTCAND
      DIMENSION NB(3612)                                             3
      EQUIVALENCE (ICNAM(1),A(5001)),(JH(1),NB(1))                   4
      DATA MAXTRK /101/                                              5
      DEFINE FILE 9( 78,3257,U,KD)                                   6
      REWIND 8                                                       7
      QPROP = QBL                                                    8
      IPROS = 0                                                      9
      MAST = 0                                                       10
      DO 90 I = 1,6                                                  11
   90 IPROP(I) = 0                                                   12
      LPOINR=5000                                                    13
      IOBJ = 1                                                       14
      USERHS = 1                                                     15
      KINP = 5                                                       16
      KSTR = 2                                                       17
      MAXNCA = 100                                                   18
      MAXCA = 10                                                     19
      NCANFR = 15                                                    20
      INVFRQ = 30                                                    21
      INVFMA = 20                                                    22
      NPH = 1                                                        23
      KMULT = 3                                                      24
      ITOL = 0                                                       25
```

```
      ICPI = 0                                                          26
      IRST = 0                                                          27
      ICLAST = 999                                                      28
      ISAVE = 1                                                         29
      ISOL = 0                                                          30
      RPERC = 40.                                                       31
      RSTOP = 0.01                                                      32
      ZTOLZE = 1.0E-07                                                  33
      ZTOLPV = 1.0E-06                                                  34
      ZTCOST = 1.0E-04                                                  35
      ZTOLRJ = 1.0E-05                                                  36
      ZTCAND = 1.0E-02                                                  37
C
      CALL TICTAC(ITIMIN)                                               38
      NROW = 0                                                          39
      OPEN (2,FILE='DECOMP.DAT',STATUS='OLD')                           40
      READ(2,PARAM)                                                     41
      IF (ITOL .NE. 0) READ(5,TOL)                                      42
      IF (KSTR .EQ. 5 .AND. MAXCA .GT. NPROS)MAXCA = NPROS              43
      IF(MAXNCA .GT. 150) MAXNCA = 150                                  44
      IF (KMULT .LE. 0 .OR. KMULT .GT. 5) KMULT = 1                     45
      LMAXP1 = LMAX + 1                                                 46
      NTRAC1 = (10*NEMAX + 4*(NRMAX + NLAMAX)
     1+2*NTMAX+LTRAC+19)/LTRAC                                          47
      NTRAC2 = (8*NLAMAX + LTRAC + 7)/LTRAC                             48
```

```
      NOLA(3,LMAXP1) = 1 + NTRAC2*LMAXP1                                49
      NOLA(4,LMAXP1) = 1                                                50
      IF (LMAX .EQ. 0) GO TO 105                                        51
      DO 100 I=1,LMAX                                                   52
      NOLA(3,I) = NOLA(3,LMAXP1) + NTRAC1*I                             53
  100 NOLA(4,I) = NOLA(4,LMAXP1) + NTRAC2*I                             54
      ITRK = NOLA(3,LMAX) + NTRAC1 - 1                                  55
      GO TO 110                                                         56
  105 ITRK = NTRAC2 + NTRAC1                                            57
  110 NOLA(1,LMAXP1) = ITRK + 1                                         58
      ITRK = ITRK + (2*NLAMAX  + 303)/LTRAC + 1                         59
      WRITE(6,6010)LMAX,LPOINR,IOBJ,USERHS,KINP,
     1KSTR,MAXNCA,MAXCA,NCANFR,INVFRQ,INVFMA,NPH,
     2KMULT,ITOL,ICPI,IRST,ICLAST,ISAV,ISOL,NTRAC1,
     3NTRAC2,ITRK,RPERC,RSTOP                                           60
 6010 FORMAT('1PROGRAM PARAMETERS'//'0LMAX',T8,I4/
     1LPOINR',T8,I4/' IOBJ',T8,I4/' USERHS',T8,I4/' KINP',T8,I4/
     2KSTR',T8,I4/' MAXNCA',T8,I4/' MAXCA',T8,I4/' NCANFR',T8,I4/
     3' INVFRQ',T8,I4/' INVFMA',T8,I4/' NPH',T8,I4/' KMULT',T8,I4/
     4' ITOL',T8,I4/' ICPI',T8,I4/' IRST',T8,I4/' ICLAST',T8,I4/
     5 ' ISAVE',T8,I4/' ISOL',T8,I4/' NTRAC1',T8,I4/
     6 ' NTRAC2',T8,I4/' ITRK',T8,I4/' RPERC  ',G10.3,' PERCENT'/
     7 ' RSTOP  ',G10.3,' PERCENT')                                     61
       IF (ZTCAND .LE. ZTCOST) ZTCAND = ZTCOST                          62
       WRITE (6,6015) ZTOLZE,ZTOLPV,ZTCOST,ZTOLRJ,ZTCAND                63
```

```
6015  FORMAT (' ZTOLZE ',G10.3/' ZTOLPV ',G10.3/
     1 ' ZTCOST ',G10.3/' ZTOLRJ ',G10.3/' ZTCAND ',G10.3)          64
       RCTEST = -1.0E+05*ZTCOST                                     65
       RPERC = 0.01*RPERC + 1.                                      66
       RSTOP = 0.01*RSTOP                                           67
       IF (ITRK .LE. MAXTRK) GO TO 112                              68
       WRITE (6,6020) ITRK                                          69
6020  FORMAT('0INSUFFICIENT STORAGE SPACE ON UNIT 9.',I5,
     1         ' TRACKS NEEDED')                                    70
       STOP                                                         71
 112   IF (IRST .NE. 0) GO TO 1500                                  72

C*********************************************************************
C
C     Step 2 :
C
C     LMAX       : Number of sub-problems.
C     LMAXP1     : Number of sub-problems plus one.
C     LSUB       : Index of the sub-problem currently considered.
C     Z(*,*)     : Proposal buffer.
C     NROW       : Number of rows in the sub-problem currently treated.
C     NROWO      : Number of coupling rows.
C     NPROS      : Column dimension of Z(*,*).
C     IPROS      : Column index in Z(*,*) where the next proposal is to be
C                  written.
```

```
C     NPH          : Control parameter only for step 1 of D-W phase 1  When
C                    one indicates each sub-problem is to be solved to
C                    optimality to generate proposals.
C     X(*)         : Array containing the solution  of the problem being
C                    considered.
C     MSTAT        : Status of the problem being considered.  QN indicates
C                    that the problem is infeasible.
C
C     A problem is read from the input file.  There are now two cases :
C
C     Case 1  : (Don't generate proposals)  If the problem read is the master
C     problem or if NPH is not one this case is met.  The data relevant to the
C     problem is written to the direct access  device.  [lines 82-85,101-103]
C     If the problem read is the master problem then mark the coupling rows as
C     "non-binding" and incorporate the coefficient of the logicals (ones) in the
C     array A(*).  [lines 106-108].
C     Set NROW to NROWO.  Since the sub-problem data contains the coupling
C     rows as non-binding rows, this is necessary to keep a correct tally for the
C     number of rows in a sub-problem.  [lines 110].
C     Read the names (row and column) of the master problem.  This is done to
C     check that the coupling rows in the sub-problem's data are actually given
C     in the problem.  [lines 112-113].
C     Case 2  : (Generate proposals)  NORMAL is called to solve the
C     sub-problem.  If the sub-problem is infeasible then the whole problem is
C     infeasible.  In this case STOP.  [lines 87-91].
```

```
C     Otherwise a proposal would have been generated.  In this case increment
C     IPROS, the column pointer in Z(*,*) at which the proposal is to be
C     written.  The first NROWO rows of X(*) contain the proposal, modulo a
C     minus sign.  Hence copy -X(*) to Z(*,IPROS).  [lines 92-98].
C     Write the data relevant to the problem to the unit 9 file.  [lines 101-103].
C     Set NROW to NROWO.  Also, read the names of the master problem.
C     These are done for the same reason as explained in case 1.
C
C*************************************************************************
      DO 1000 I=1,LMAXP1                                                  73
       IF (I .LE. LMAX) MSTATU(I) = QF                                    74
       LSUB = I - 1                                                       75
       CALL INPUT                                                         76
       ITCNT = 0                                                          77
       NT = 0                                                             78
       NX = 0                                                             79
       ITSINV = 0                                                         80
       KK = LSUB                                                          81
       IF(LSUB .EQ. 0)KK = LMAXP1                                         82
       KD = NOLA(4,KK)                                                    83
       WRITE(9'KD)NTEMP(1),NTEMP(2),ICNAM                                 84
       LCORE = LSUB                                                       85
       IF(LSUB .EQ. 0 .OR. NPH .NE. 1)GO TO 500                           86
       CALL NORMAL                                                        87
       IF(MSTAT .NE. QN)GO TO 300                                         88
```

```
      WRITE(6,200)LSUB,SUMINF                                          89
200   FORMAT(//,' SUBPROBLEM ',I4,'  HAS NO SOLUTION.                  90
     SUM OF INFEASIBILITIES = ',F16.8)
      STOP                                                             91
300   IPROS = IPROS + 1                                                92
      IF (IPROS .LE. NPROS) GO TO 330                                  93
      WRITE (8) Z                                                      94
       IPROS = 1                                                       95
330    DO 350 J=1,NROWO                                                96
350    Z(J,IPROS) = -X(J)                                              97
      Z(NROWO+1,IPROS) = 0.0                                           98
500   KD = NOLA(3,KK)                                                  99
      IF(NELEM.GT.(NEMAX-NRMAX).OR.                                    100
     ITSINV.GT.(4*INVFRQ)/5)NETA = 0
      NOLA(2,KK) = LE(NETA+1) - 1                                      101
      KIKA = 1                                                         102
      CALL VECTOR(KIKA,A,IA,NB,NOLA(2,KK))                             103
600   IF(LSUB .GT. 0)GO TO 800                                         104
      IF(LMAX .EQ. 0)GO TO 1000                                        105
      DO 750 J=1,NROWO                                                 106
      ISTYPE(J) = 0                                                    107
750   A(J) = 1.                                                        108
800   IF(LSUB .EQ. LMAX)GO TO 1000                                     109
      NROW = NROWO                                                     110
      IF (LSUB .EQ. 0 .OR. NPH .NE. 1) GO TO 1000                      111
```

```
      KD = NOLA(4,KK)                                                  112
      READ(9'KD) NTEMP(1),NTEMP(2),ICNAM                               113
1000  CONTINUE                                                         114

C*********************************************************************
C
C     Step 3 :
C
C     A(*)          : Array which has the data and eta elements of the problem
C     NOLA(*,*)     : NOLA(3,*) indicates the record number on the direct
C                     access file where tha data and the eta elements of the *th
C                     problem are stored.  NOLA(2,*) indicates the position in
C                     the array A(*) where the eta elements of the *th problem
C                     end.
C     KIKA          : One indicates that the data is to be written on the direct
C                     access file and two indicates that the data is to be read
C                     from the direct access file.
C     LPOINR        : Indicates the position in the array A(*) where the eta
C                     elements begin.
C     LE(1)         : Indicates where in the array A(*) the eta file of the
C                     problem begins.
C     NPH           : Control parameter used only in step 1 of D-W phase 1.
C                     When not one it indicates that no proposals are to be
C                     generated in step 1 of D-W phase 1.  Otherwise indicates
C                     that proposals are to be generated only at optimality.
```

C NCAND : Number of proposals generated in the current cycle.

C LMAX : Number of sub-problems.

C NROWO : Number of coupling rows.

C NCOLO : Number of coupling columns.

C JH(*) : Indicates the column that is basic in the $*^{th}$ row.

C KINBAS(*) : Indicates if $*^{th}$ column is basic and if so the row in

C which it is basic.

C

C 1. Obtain the record number of the direct access file where the problem's

C data and eta are stored. [line 115]

C 2. Set KIKA to 2 indicating that the data is to be read from the direct

C access device. [line 118]

C 3. Call VECTOR to read the data from the direct access device. [line 119]

C 4. If NPH is not one then no proposals would be generated and therefore

C the master problem is ready to be solved. In this case go to . Otherwise

C proceed as follows. [line 121].

C 5. Since problems are solved sequentially and since only one proposal is

C generated at optimality, the j^{th} proposal is from the j^{th} sub-problem.

C Hence assign J to NAME (J). Also set NCAND to LMAX since one

C proposal has been generated per sub-problem. Call PACK to

C incorporate the proposals in the array A(*). [lines 122-125].

C 6. The convexity constraint corresponding to each sub-problem has only

C one column and therefore it has to be basic

C Set JH(*) and KINBAS(*) to indicate this. [lines 129-134]

C

```
C*****************************************************************
      KD = NOLA(3,LMAXP1)                                          115
      LSUB = 0                                                     116
      LCORE = 0                                                    117
      KIKA = 2                                                     118
      CALL VECTOR(KIKA,A,IA,NB,NOLA(2,LMAXP1))                     119
      LE(1) = LPOINR                                               120
      IF(NPH .NE. 1 .OR. LMAX .EQ. 0) GO TO 1500                   121
      DO 1100 J=1,LMAX                                             122
1100  NAME(J) = J                                                  123
      NCAND = LMAX                                                 124
      CALL PACK                                                    125
      NCAND = 0                                                    126
      MM = NCOLO                                                   127
      NN = NROWO                                                   128
      DO 1200 J=1,LMAX                                             129
      NN = NN + 1                                                  130
      MM = MM + 1                                                  131
      JH(NN) = MM                                                  132
      KINBAS(NN) = 0                                               133
      KINBAS(MM) = NN                                              134
1200  CONTINUE                                                     135
1500  CALL TICTAC(JTIMIN)                                          136
      TIMER=(JTIMIN-ITIMIN)/1000.                                  137
      WRITE(6,2000)TIMER                                           138
```

```
2000  FORMAT(//'0TOTAL INPUT TIME = ',F8.2,'  SECONDS',/,1H1)   139
      IF(LMAX .GE. 1)MAST = 1                                    140
      RETURN                                                     141
      END                                                        142
```

3.9. Subroutine INPUT :

3.9.1. Major Task :

Read an LP in MPS format.

3.9.2. Input :

A file called DECOMP.DAT that contains the problem parameters and the problem data. The input file of a typical problem is given in the user's guide.

3.9.3. Called By :

INDATA.

3.9.4. Detailed Comments :

```
C*****************************************************************
C
C     LSUB        : Problem index of the problem that is to be read in.
C     NROW        : Number of rows in the problem currently read.
C     NROWO       : Number of coupling rows.
C     A(*)        : Data array.
C     KINBAS(*)   : Indicates if the *th column is basic and if so the row in
C                   which it is basic.
```

```
C     JH(*)        : Indicates the column that is basic in the *th row.
C     LMAX         : Number of sub-problems.
C     NCOL         : Number of columns (including rhs) in the problem
C                    currently read.
C     NELEM        : Number of non-zeros in the problem currently read.
C     USERHS       : Control parameter. DECOMP allows for the possibility
C                    of reading in many right hand sides and the USERHSth
C                    rhs is used as the right hand side of the problem.
C     IVIN         : Number of columns in the "columns" section of the
C                    problem currently read.
C
C     1. Read a card.
C     2. The following steps are executed provided the card just read is not
C        an "END" card.  In the case of an "END" card go to step 3.
C        (a) There are now several cases :
C              Case 1 :  (NAMES card)
C              - Record the name in NTEMP.
C              - Write the card.
C              Case 2 :  (ROWS card)
C              - Set L = 1 to flag the start the of the "ROWS" section.
C              - Write the card.
C              Case 3 :  (COLUMNS card)
C              - Set L = 2 to flag the start of the COLUMNS section.
C              - If reading the master problem then
C                    - The rows listed in the "rows" section of the master
```

C problem are the coupling rows. Hence set NROWO to
C NROW. [line 50]
C - An index is maintained to indicate the number of binding
C coupling rows. If there are none then the entire problem
C breaks up into LMAX independent problems. Hence, all
C sub-problems solved to optimality would specify the
C optimal solution and this can be achieved by setting
C KSTR to one. [lines 53-58]
C - Incorporate the convexity rows into the A(*). Also,
C mark the type of the convexity rows as binding equality
C rows. [lines 59,62-67]
C - Increment NROW by LMAX since the master problem
C has LMAX convexity conatriants. [lines 59,60]
C - Name the i^{th} convexity row QQi. [lines 68-69]
C End if
C - At this point the problem has only logicals. Therefore, it has as
C many columns and as many non-zero elemens as there are rows.
C Hence, set both NCOL and NELEM to NROW. [lines 71-72]
C *Case 4* : (RHS card)
C - Set L = 4 to signify the start of the RHS section.
C - Set IVIN to NCOL. Hence IVIN indicates the number column in
C the columns section of the problem. [line 134]
C - Write the card.
C *Case 5* : (L = 1 implying ROWS' section card)
C - Ignore coupling rows included in the ROWS' section of a

C sub-problem. [lines 25-27]

C - Increment row count (NROW). [line 28]

C - Record row name and row type. [lines 30-31]

C - Incorporate logical coefficient in A(*) according to row type.

C A(*) is -1 for ≥ row and 1 otherwise. [lines 32-49]

C - Record KINBAS(*) and JH(*).

C *Case 6* : (L = 2 or 4 implying COLUMNS' or RHS section card)

C - If the column name on the current card is not the same as that on

C the previous card then it implies the start of a new column.

C [line 80]

C In this case :

C - Test for split vector ((i.e) test if part of this column was

C inputted before) and if so print a message. [lines 79-84]

C - Increment the number of columns (NCOL). [line 85]

C - Record column name. [lines 86-87]

C - Set LA(NCOL) to NELEM. Hence, LA(NCOL)

C indicates where in A(*) the $NCOL^{th}$ column begins.

C [line 92]

C End If

C - If the row name on the current card matches with one of the

C names given in "ROWS" section of the master problem then the

C non-zero element on this card is part of the data. Hence

C incorporate the data into the array A(*) and increment NELEM.

C [lines 92-112].

C (b) Read a card.

```
C       2. Once the "END" card is read then
C               - Set LA(NCOL+1) to NELEM+1. This signifies the end of data
C                 in all NCOL columns. [line 134]
C               - Since the eta file for every problem except the master starts at the
C                 end of the problem's data LE(1) is set to NELEM+1. [line 135]
C               - IVIN indicates the number of columns after reading the data in
C                 the columns section. DECOMP allows for the possibility of
C                 choosing among various right hand sides. Hence, by setting
C                 RHSCOL to IVIN+USERHS the user can specify that the
C                 USERHS^th right hand side be used for the current run.
C                 [line 137]
C               - If master problem then initialize the origin of these columns.
C               - Print problem statistics.
C
C*****************************************************************
      EQUIVALENCE (ICNAM(1),A(5001))                              1
      CALL TICTAC(JTIM)                                           2
      WRITE(6,6000)                                               3
 6000 FORMAT(1H1)                                                 4
    5 READ(KINP,700)K1,K2,K3,K4,(NAME(I),I=1,4),                  5
      ATEMP1,NAME(5),NAME(6), ATEMP2
  700 FORMAT(4A1,2A4,2X,2A4,2X,G12.6,3X,2A4,2X,G12.6)             6
      IF(K1 .EQ. QBL)GO TO 50                                     7
      IF(K1 .EQ. QN)GO TO 100                                     8
      IF(K1 .EQ. QR .AND. K2 .EQ. QO)L=1                          9
```

```
      IF(K1 .EQ. QR .AND. K2 .EQ. QO)GO TO 150                    10
      IF(K1 .EQ. QC)L=2                                           11
      IF(K1 .EQ. QC)GO TO 300                                     12
      IF(K1 .EQ. QB .AND. K2 .EQ. QA)L=3                          13
      IF(K1 .EQ. QB .AND. K2 .EQ. QA)GO TO 150                    14
      IF(K1 .EQ. QR .AND. K2 .EQ. QH)L=4                          15
      IF(K1 .EQ. QR .AND. K2 .EQ. QH)GO TO 500                    16
      IF(K1 .EQ. QE)GO TO 600                                     17
 50   GO TO(210,320,410,320),L                                    18
100   NTEMP(1) = NAME(3)                                          19
      NTEMP(2) = NAME(4)                                          20
150   WRITE(6,710)K1,K2,K3,K4,NAME(1),NAME(2),                    21
      NAME(3),NAME(4)
710   FORMAT(3X,4A1,2A4,2X,2A4)                                   22
      GO TO 5                                                     23
C
210   IF (LSUB .LE. 0) GO TO 215                                  24
C
      DO 212 I = 1,NROWO                                          25
      IF (ICNAM(I,1) .NE. NAME(1) .OR.                            26
      ICNAM(I,2) .NE. NAME(2)) GO TO 212
      GO TO 5                                                     27
212   CONTINUE                                                    28
C
215   NROW = NROW + 1                                             29
```

```
      ICNAM(NROW,1) = NAME(1)                                    30
      ICNAM(NROW,2) = NAME(2)                                    31
C
C               TEST ROW TYPE
C
      IF(K2 .EQ. QL .OR. K3 .EQ. QL)GO TO 220                    32
      IF(K2 .EQ. QE .OR. K3 .EQ. QE)GO TO 230                    33
      IF(K2 .EQ. QG .OR. K3 .EQ. QG)GO TO 240                    34
      IF(K2 .EQ. QN .OR. K3 .EQ. QN)GO TO 250                    35
      GO TO 230                                                  36
  220 ISTYPE(NROW) = 1                                           37
      GO TO 255                                                  38
  230 ISTYPE(NROW) = -1                                          39
      GO TO 255                                                  40
  240 ISTYPE(NROW) = 1                                           41
      A(NROW) = -1.                                              42
      GO TO 260                                                  43
  250 ISTYPE(NROW) = 0                                           44
  255 A(NROW) = 1.                                               45
  260 IA(NROW) = NROW                                            46
      LA(NROW) = NROW                                            47
      JH(NROW) = NROW                                            48
      KINBAS(NROW) = NROW                                        49
      GO TO 5                                                    50
C
```

```
C       MATRIX ELEMENTS
C
 300  IF(LSUB .GT. 0 .OR. LMAX .EQ. 0)GO TO 310                    51
      NROWO = NROW                                                 52
      J = 0                                                        53
      DO 305 I=1,NROW                                              54
      IF(ISTYPE(I) .EQ. 0)GO TO 305                                55
      J = J + 1                                                    56
 305  CONTINUE                                                     57
      IF(J .EQ. 0)KSTR = 1                                         58
      DO 308 I=1,LMAX                                              59
      NROW = NROW + 1                                              60
      ISTYPE(NROW) = - 1                                           61
      A(NROW) = 1.                                                 62
      IA(NROW) = NROW                                              63
      LA(NROW) = NROW                                              64
      ICOLS(NROW) = 0                                              65
      JH(NROW) = NROW                                              66
      KINBAS(NROW) = NROW                                          67
      ICNAM(NROW,1) = QQ                                           68
      ICNAM(NROW,2) = I                                            69
 308  CONTINUE                                                     70
 310  NCOL = NROW                                                  71
      NELEM = NROW                                                 72
      ICS1 = QQ                                                    73
```

```
      ICS2 = QQ                                                    74
      GO TO 150                                                    75
  320 J = 3                                                        76
      K = 4                                                        77
      IF (NAME(1) .EQ.ICS1 .AND. NAME(2) .EQ.ICS2) GO TO 328       78
C
C     TEST FOR SPLIT VECTOR
C
      DO 325 I = 1,NCOL                                            79
      IF (NAME(1) .NE. ICNAM(I,1)) GO TO 325                       80
      IF (NAME(2) .NE. ICNAM(I,2)) GO TO 325                       81
      WRITE(6,8250) NAME(1),NAME(2)                                82
 8250 FORMAT(14H0SPLIT VECTOR ,2A4)                                83
  325 CONTINUE                                                     84
      NCOL = NCOL + 1                                              85
      ICS1 = NAME(1)                                               86
      ICS2 = NAME(2)                                               87
      ICNAM(NCOL,1) = ICS1                                         88
      ICNAM(NCOL,2) = ICS2                                         89
      LA(NCOL) = NELEM + 1                                         90
      KINBAS(NCOL) = 0                                             91
C
C     TEST FOR ROW MATCH
C
  328 IF (DABS(ATEMP1) .GT. ZTOLZE) GO TO 330                      92
```

```
      J = 5                                                    93
      K = 6                                                    94
      ATEMP1 = ATEMP2                                          95
  330 DO 340 I = 1,NROW                                        96
      IF (DABS(ATEMP1) .LE. ZTOLZE) GO TO 335                  97
      IF(NAME(J) .NE. ICNAM(I,1) .OR.                          98
      NAME(K) .NE. ICNAM(I,2))GO TO 340
      IF (L .EQ. 4 .AND. LSUB .GT. 0 .AND. I .LE.              99
     1NROWO)GO TO 335
      NELEM = NELEM + 1                                        100
      IA(NELEM) = I                                            101
      A(NELEM) = ATEMP1                                        102
  335 IF (K .GT. 5) GO TO 5                                    103
      IF(DABS(ATEMP2) .LE. ZTOLZE)GO TO 5                      104
      J = 5                                                    105
      K = 6                                                    106
      ATEMP1 = ATEMP2                                          107
      GO TO 330                                                108
  340 CONTINUE                                                 109
      WRITE(6,8300) NAME(J),NAME(K)                            110
 8300 FORMAT(18H0NO MATCH FOR ROW ,2A4)                        111
      STOP                                                     112
C
C     BASIS CARDS
C
```

```
410   DO 420 I = 1,NCOL                                          113
      IF(NAME(1) .NE. ICNAM(I,1) .OR.                            114
      NAME(2) .NE. ICNAM(I,2)) GO TO 420
      IBVEC = I                                                  115
      GO TO 425                                                  116
420   CONTINUE                                                   117
      WRITE(6,8400)NAME(1),NAME(2)                               118
8400  FORMAT(21H0NO MATCH FOR VECTOR ,2A4)                       119
      GO TO 5                                                    120
425   DO 430 I = 1,NROW                                          121
      IF(NAME(3) .NE. ICNAM(I,1) .OR.                            122
      NAME(4) .NE. ICNAM(I,2)) GO TO 430
      IBROW = I                                                  123
      GO TO 440                                                  124
430   CONTINUE                                                   125
      WRITE(6,8300)NAME(3),NAME(4)                               126
      GO TO 5                                                    127
440   JH(IBROW) = IBVEC                                          128
      KINBAS(IBROW) = 0                                          129
      KINBAS(IBVEC) = IBROW                                      130
      GO TO 5                                                    131
C
C     RHS
C
500   IVIN = NCOL                                                132
```

```
      GO TO 150                                                    133
C
C       END OF INPUT
C
  600 LA(NCOL+1) = NELEM + 1                                       134
      LE(1) = NELEM + 1                                            135
      NETA = 0                                                     136
      RHSCOL = IVIN + USERHS                                       137
      NUMRHS = NCOL - IVIN                                         138
      IF(LSUB .GT. 0)GO TO 615                                     139
      DO 605 I=1,NCOL                                              140
  605 ICOLS(I) = 0                                                 141
      MM = IVIN + 1                                                142
      DO 610 I=MM,NCOL                                             143
  610 KINBAS(I) = NROW + 1                                         144
      NCOLO = NCOL                                                 145
      NSCOL = IVIN - NROW                                          146
      GO TO 620                                                    147
  615 NCOL = IVIN                                                  148
      NSCOL = NCOL - NROW                                          149
  620 NELEM = LA(IVIN+1) - 1 - NROW                                150
      RELEM = NELEM                                                151
      RDENS = 100.                                                 152
      IF(NSCOL .EQ. 0)GO TO 625                                    153
      RDENS = RELEM/(NROW*NSCOL)*100.                              154
```

```
625   WRITE(6,710)K1,K2,K3,K4,NAME(1),NAME(2)                          155
      WRITE(6,8500)NROW,NSCOL,NELEM,RDENS                              156
8500  FORMAT(19H0PROBLEM STATISTICS/1H ,I5,5H ROWS/1H ,I5,             157
     1  19H STRUCTURAL COLUMNS /1H I5,18H NON-ZERO
      ELEMENTS/ 211H DENSITY = ,F6.2,' PERCENT')
      CALL TICTAC(ITIM)                                                158
      TIMER=(ITIM-JTIM)/1000.                                          159
      WRITE(6,8900)NTEMP(1),NTEMP(2),TIMER                             160
8900  FORMAT(  '0INPUT TIME SUBPROBLEM ',2A4,' = ',F6.2,'              161
      SECONDS')
      RETURN                                                           162
      END                                                              163
```

3.10. Subroutine INVERT

3.10.1. Major Task :

Computes the L-U factorization of the basis.

3.10.2. Inputs :

1. The array JH(I) which specifies the column that is basic in the i^{th} row.
2. The array KINBAS(I) which tells if the i^{th} column is basic and if so the row in which it is basic.
3. The basic columns which reside in the array A(*).

3.10.3. Outputs :

1. Eta vectors which define the factors L and U of the basis **B**.
2. The updated solution vector X(*).

3.10.4. Description :

INVERT applies a series of symbolic permutations to the basis. Roughly speaking, this changes **B** so that it is "almost" lower triangular. Thereby, only a small sub-matrix of **B** would have to be factored. INVERT consists of five steps :

1. A pivot sequence is decided for all row and column singletons. A row (or column) singleton is a row (or column) having just one non-zero element after the

removal of rows and columns corresponding to all previously assigned pivots.

2. The resulting basis matrix is then permuted to an easily factorable form.
3. At the end of steps 1 and 2, it will be seen later that only a sub-matrix **B2** of **B** (called the bump) would have to be factored. Step 3 consists of computing the L-U factors of the bump **B2.**
4. Form etas.
5. Compute the solution.

The matrix is conceptually partitioned into four. The first partition called part 1 contains row singletons. The second and third partitions called part 3 and part 4 respectively contain column singletons. The unnamed partition is the bump **B2.** Step 1 is known as pre-assigned pivot procedure and is based on Hellerman and Rarick's method (1972).

Step 1 Pre-assigned Pivot sequence : Step1 is done in three stages.

1.1 Put logicals in part 4 : A scan is conducted through all rows as follows. Consider the i^{th} row. The basic variable of each row is tested to see if it is a logical. If so, rows and columns are permuted so that the logical appears on the diagonal of part 4. Otherwise, the column belongs tentatively to part 1 and hence the number of columns in part 1 is incremented. At the end of step 1.1 a general basis as shown in figure (3.2) has the structure as shown in figure (3.3). Note that **permutations are never carried out physically.** Instead, the effect of a permutation is achieved by changing the order of pivoting. This is done through two arrays MREG(i) and VREG(i) which indicate that the i^{th} pivot will be in row MREG(i) and column VREG(i). For

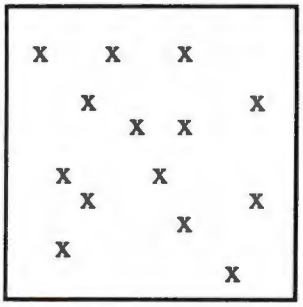
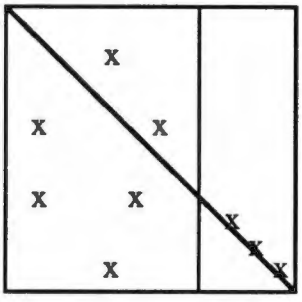


Figure 3.2 : Structure of a general basis before inversion.



Tentative Part 1 Part 4

Figure 3.3 : Structure of the basis after step 1.1

example, consider the matrix in figure (3.4). A logical would be detected while considering the third row. MREG(5) is set to 4 and VREG(5) is also set to 4 indicating that the 5^{th} pivot will be in row 4 and column 4. This is shown in figure (3.5).

1.2 Establish row counts : First, rows and columns corresponding to all singleton elements of part 4 are marked as "unavailable" (assigned for pivot). Then all columns of the tentative part 1 are scanned as follows. Without loss of generality, consider the j^{th} part 1 column. The row index (say i) of a non-zero in the column is obtained. If the i^{th} row was not assigned for pivot in part 4, then the number of non-zeros in that row is updated. On the other hand, if the i^{th} row was assigned for pivot then it is ignored. This is repeated for all non-zeros in that column. If the column has only one non-zero in unassigned rows, then a column singleton has been detected and in this case rows and column are permuted so that the singleton element appears on the diagonal of part 3. Thereby, at the end of the scan one can obtain the non-zero count of each row. At the end of step 1.2, the matrix would have the structure as shown in figure (3.6). For the example given in step 1.1, column 5 is a column singleton. Therefore, MREG(4) is set to 3 and VREG(4) is set to 5 to give the effect of a permutation. This is shown in figure (3.7).

1.3 Put row singletons in part 1 and new column singletons in part 3 : This

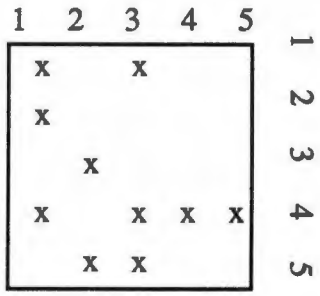


Figure 3.4 : An example basis

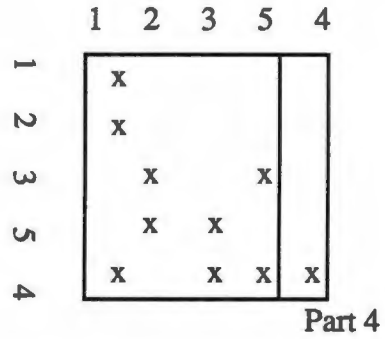


Figure 3.5 : Structure of the example basis after step 1.1

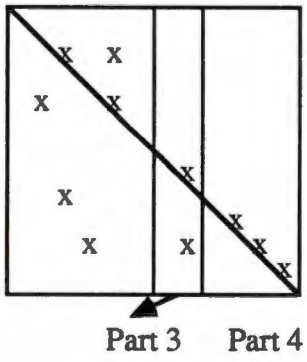


Figure 3.6 : Structure of the basis after step 1.2

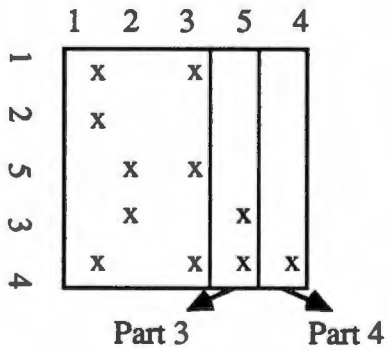


Figure 3.7 : Structure of the example basis after step 1.2

step is a multi-pass scan through part 1 columns until all row and column singletons have been removed. Everytime a pass is completed, an additional pass is undertaken provided atleast one row or column singleton was detected during the previous pass. In each pass all part 1 columns are scanned to put row singletons in part 1 and column singletons in part 3 as explained below. Consider the j^{th} part 1 column during the p^{th} pass. The row index (say i) of a non-zero in the column is obtained. If row i has been assigned for pivot it is ignored. If the row was not assigned and is a row singleton (recall step 1.2 gives the non-zero count of all part 1 rows), then

(a) rows and columns are permuted so that the singleton element appears on the diagonal of part 1.

(b) the row and column corresponding to the singleton element is marked as "unavailable" (assigned for pivot) and then the next ($(j+1)^{st}$) part 1 column is taken up.

This procedure is repeated for all non-zeros in the j^{th} column . If no row singletons were detected and if the number of non-zeros in unassigned rows of the column is one, then a column singleton has been detected. In this case, rows and columns are permuted so that the singleton element appears on the diagonal of part 3. Again, the row and column corresponding to the singleton element is marked as "unavailable". At the end of this step the matrix would have the structure as shown in figure (3.8). For the example, the structure looks as shown in figure (3.9).

Step 2 *Bringing the basis to an easily factorable form* : This is illustrated in the

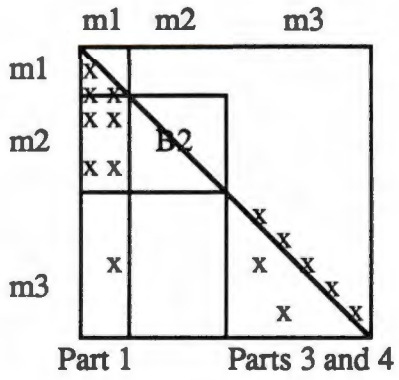


Figure 3.8 : Structure of the basis
after step 1.3

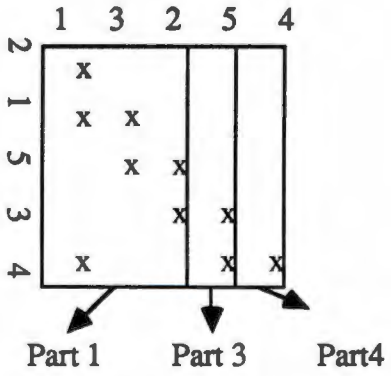


Figure 3.9 : Structure of the example
basis after step 1.3.

figures (3.10) - (3.12) below. Knowing the L-U factors of the bump **B2** (= **L2'U2'** say), the L-U factors of the matrix given in figure (3.12) is shown in figure (3.13).

Step 3 Factorizing the bump **B2** : First a pivot sequence has to be established to minimize the amount of fill-ins. Then the factors are obtained using Gaussian elimination and the corresponding etas are written to the array A.

3.1 Pivot Sequence for the bump : This is based on the following heuristic. For each column, a "meritcount" is computed. The meritcount m(j), of j^{th} column, is the sum of the number of non-zeros in rows having non-zeros in the j^{th} column. Thus

$$m(j) = \sum_{i} (r_i f_{ij})$$

where

r_i is the number of non-zeros in row i and

f_{ij} is 1 if row i has a non-zero in column j and 0 otherwise.

For example, the merit count of column j is seven for the matrix shown in

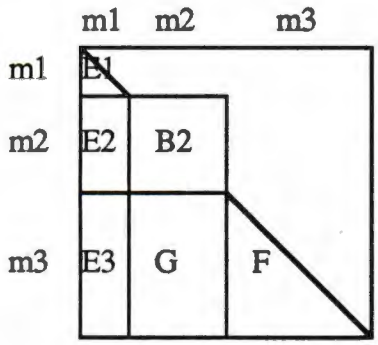


Figure 3.10 : Structure of the basis after step 1.

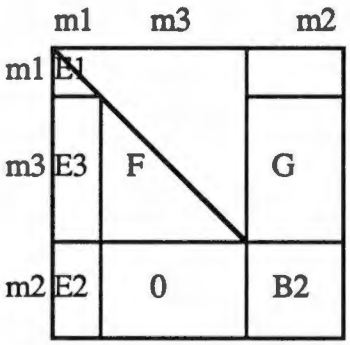


Figure 3.11 : Illustration of step 2.

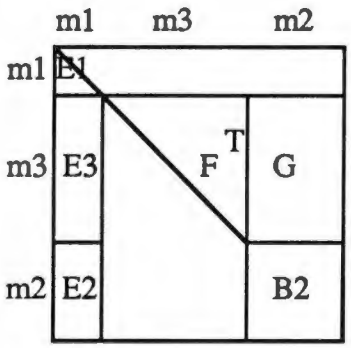


Figure 3.12 : Structure of the basis after step 2.

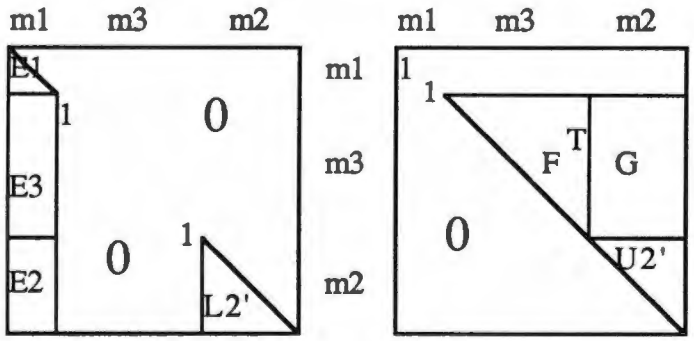


Figure 3.13 : L-U factors of the basis

figure (3.14). Then columns are sorted in the ascending order of their meritcounts using shell sort.

3.2 L-U factors of the bump : This is done by Gaussian elimination. The jth step of Gaussian elemenation which consists of pre-multiplying **B2** by the elementary matrix $\mathbf{L2''_j}$ is as shown in figure (3.15). Thus, at the end of m2 (since size of **B2** is m2 by m2) Gaussian elimination steps we have :

$$\mathbf{L2''_1\ L2''_2 \ldots L2''_{m2}\ B2} = \mathbf{U2'} \qquad (3.30)$$

$$\mathbf{B2} = (\mathbf{L2'_{m2} \ldots L2'_1})\ (\mathbf{U2'_1 \ldots U2'_{m2}}) \qquad (3.31)$$

where $\mathbf{U2'_j}$ (j = 1,...,m2) is the elementary matrix whose j^{th} column is the j^{th} column of **U2'** and $\mathbf{L2'_j = L2''_j{}^{-1}}$ (j = 1,...,m2).

Step 4 Form Etas : From (3.31) and figure (3.13) the etas of the basis are given by :

$$\mathbf{L1\ L2\ U2\ U3} \qquad (3.32)$$

where

$$\mathbf{L1 = L1_1 \ldots L1_{m1}},$$

$$\mathbf{L2 = L2_1 \ldots L2_{m2}},$$

$$\mathbf{U2 = U2_{m2} \ldots U2_1},$$

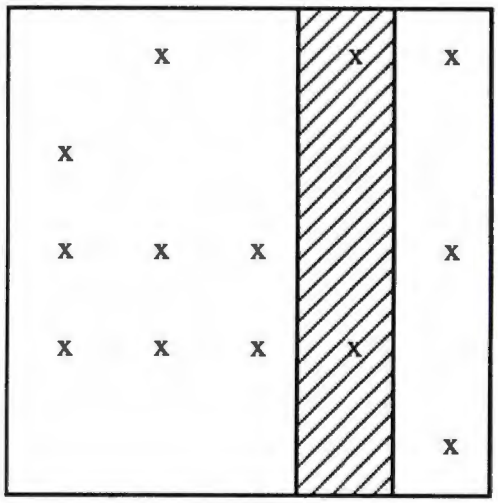


Figure 3.14 : Illustration of merit count

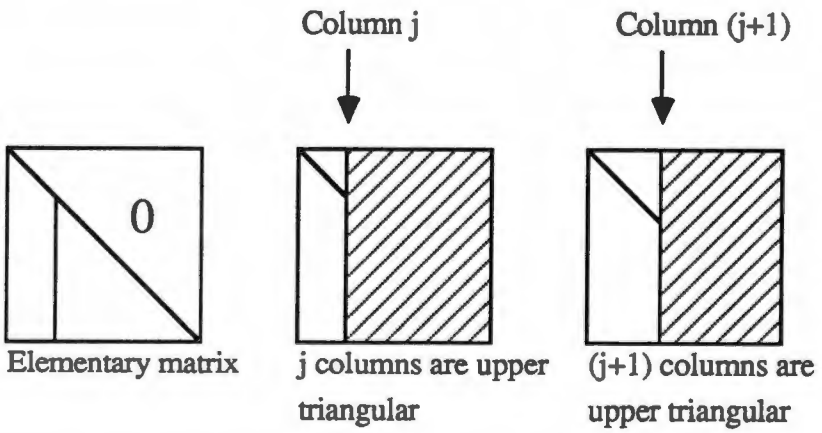


Figure 3.15 : A step of Gaussian elimination.

$\mathbf{L1_j}$ is an elementary matrix of size m x m whose j^{th} column is the j^{th} column of $[\mathbf{E1^t},\mathbf{E3},\mathbf{E2}]^t$,

$\mathbf{L2_j}$ is an elementary matrix of size m x m whose $(m1+m3+j)^{th}$ column is the j^{th} column of $[\mathbf{0},\mathbf{0},\mathbf{L2'^t_j}]^t$,

$\mathbf{U2_j}$ is an elementary matrix of size m x m whose $(m1+m3+j)^{th}$ column is the j^{th} column of $[\mathbf{0},\mathbf{G},\mathbf{U2'^t_j}]^t$,

$\mathbf{U3_j}$ is an elementary matrix of size m x m whose $(m1+j)^{th}$ column is the j^{th} column of $[\mathbf{0},\mathbf{F},\mathbf{0}]^t$.

Step 5 Update solution : This is done by copying the original rhs of the given LP and then calling FTRAN to compute the solution $\mathbf{B^{-1}b}$.

3.11. Subroutine MASTER :

3.11.1. Major Task :

Performs step 2 of D-W phase 1 and phase 2.

3.11.2. Description :

MASTER is the driver routine for step 2 of D-W phase 1 and D-W phase 2. Occasionally, the decomposition algorithm has to be restarted from a cycle saved in a precious run. In this case, all the data relevant to the problem is read from the unit 9 direct access device (DAD). MASTER then executes a series of cycles till the stopping criterion is met or the maximum limit on the number of cycles has been exceeded. Every cycle begins by first solving the master problem. POLICY is then called provided the previous cycle was not an interrupted cycle (cf. § 2.6.2). Whenever POLICY is called it sets the policy for the current cycle as follows.

1. Sets KK to the number of problems to be solved.
2. Sets the order in which the problems are to be solved.
3. Sets the number of proposals to be generated by each sub-problem in the current cycle.

Subsequently, the (KK-1) sub-problems identified by POLICY are solved except when a cycle gets interrupted. In this case, assume that the cycle was interrupted while solving the i^{th} sub-problem ($1 \leq i \leq KK-1$). A new cycle is then begun by solving the master problem. The first sub-problem to be solved in this cycle is

either the i^{th} sub-problem or the $(i+1)^{st}$ sub-problem depending on the strategy chosen by the user. Note that, this cycle may itself be interrupted again in which case the same procedure is repeated. Otherwise, a new cycle would be started by solving the master problem whenever all the remainder of the (KK-1) sub-problems have been solved. POLICY is then called and the new cycle is continued by solving sub-problems. Master and sub-problems are solved as follows :

Case 1 : (Master problem) First the lower bound of the objective value is computed and then NORMAL is called to solve the master problem. If the maximum cycle limit has been exceeded or if the problem is unbounded then the present run is terminated. Otherwise, the relative difference in the objective value (RELDIFF) is computed. If RELDIFF is less than the tolerance of the stopping criterion (RSTOP) then the solution procedure is continued. Then the dual variables of the master problem are stored in YA(*,6) and the appropriate sub-problems are solved to generate proposals.

Case 2 : (Sub-problem) The dual variables of the convexity rows of the master problem are stored for performing the candidacy test later and then NORMAL is called to solve the sub-problem. The number of proposals generated by the sub-problem during the current cycle is recorded. The proposals, if any, are written from the proposal buffer Z(*,*) to the unit 8 work file, provided the user chooses a strategy of solving problems to optimality and sending only q_{max} proposals (cf. § 2.6.2). A check is performed to determine if the cycle should be interrupted and based on this, the next sub-problem is solved or a new cycle is started.

MASTER therefore comprises the following steps.

3.11.3. Steps :

Refer to the psuedo code for MASTER in section 2.7

3.11.4. Detailed Comments :

```
C*********************************************************************
C
C       Step 1 :
C
C       KD              : Points to DAD for accessing the problem's data
C       IRST            : Flag. When set to 1 then the solution procedure is
C                         continued from a cycle saved in a previous run.
C
C       - Initialize variables. [lines 1-9].
C       - If IRST is 1 then set KD so that it points to the DAD for restart
C         information. Read the problem's relevant data. [lines 11-12].
C
C*********************************************************************

        ICPI1 = ICPI + 1                                        1
        INCO = 0                                                2
        CALL TICTAC(INCIAL)                                     3
```

```
      TIMER2 = 0.0                                                    4
      NCAND = 0                                                       5
      ICAND = 0                                                       6
      LCOUNT = -1                                                     7
      KK = 1                                                          8
      LS(1) = 0                                                       9
      IF (IRST .EQ. 0) GO TO 200                                      10
      KD = NOLA(1,LMAXP1)                                             11
      READ(9'KD)  DOLD,DEOB,TIMER2,NOLA,KCYC,                         12
                  NCOLO,NROWO,ICOLS,MSTATU
      LCORE = -1                                                      13
      ITEMP = ICPI                                                    14
      ICPI = KCYC                                                     15
      GO TO 310                                                       16
200   MAJOR = 0                                                       17
      DOLD = -1.0D30                                                  18
      GO TO 310                                                       19
300   CALL POLICY(KK,LCOUNT)                                          20
310   IF(KK .LE.0)GO TO 10000                                         21
      DO 1000 KKK=1,KK                                                22
315   LSUB = LS(KKK)                                                  23
320   IF(LSUB .GE. 1)GO TO 500                                        24
```

```
C*****************************************************************
C
C     SOLVMSTR :
C
C     LS(i)       : Gives the order in which problems are to be solved.
C                   Set by POLICY. If LS(2) = 5 then the 2nd problem to
C                   be solved in the cycle is problem with index 5.
C     KK          : Number of problems to be solved before POLICY can
C                   be called again.
C     INDEX       : Flag normally set to 1. When 0 indicates that the
C                   previous cycle was interrupted while solving the i^th
C                   sub-problem and the first sub-problem of the new cycle
C                   is the i^th sub-problem.
C     NCAND       : Number of proposals made in the current cycle.
C     MSTAT       : Status of the problem.
C     NOLA        : Pointer to DAD for accessing a problem's data.
C     DOLD        : Lower bound on the optimal solution.
C     KCYC        : Cycle number
C     MAJOR       : Cycle number.
C     ICPI        : Cycle number at which special prices are to be read.
C     ICLAST      : Maximum limit on the number of cycles.
C     LMAX        : Number of sub-problems.
C     LMAXP1      : Number of sub-problems plus 1.
C     ISAVE       : Flag. When 1 indicates that the problem is to be saved
C                   DAD for future restarts.
```

```
C       YA(*,*)      : Contains the master dual variables.
C       KSTR         : Strategy parameter.
C       LCOUNT       : Flag used when KSTR is 3. In this case when
C                      LCOUNT ≥ -1 indicates that in every cycle subsequen
C                      to the first, sub-problems are to be solved in a cyclic
C                      fashion.
C
C       - Print massages. {lines 34-37].
C       - Update the lower bound on the optimal solution as given by eqn (2.75)
C         of § 2.5.3. [lines 39-44].
C       - Call NORMAL to solve the master problem. [lines 45-46].
C       - If the maximum limit on the number of cycles has been exceeded then
C         save the problem's data on DAD for future restarts if ISAVE is 1.
C         Else proceed. [lines 47-58].
C       - If MSTAT is 'QU' (problem unbounded) then RETURN since the
C         whole problem is unbounded. [line 60].
C         Else proceed.
C       - Compute the relative difference in the objective value (RELDIFF) and if
C         it is less than RSTOP then STOP. Else proceed. [lines 64-67]
C       - Read the dual variables if the cycle number (MAJOR) is ICPI.
C         [lines 68-71]
C       - Set each sub-problem's status feasible and start solving the appropriate
C         sub-problem. [lines 78-81]
C
C*************************************************************************
```

```
      INDEX = 1                                                    25
      IF(NCAND .GE. 1 .OR. LCOUNT .EQ. -1)GO TO 350                26
      GO TO 1000                                                   27
350   IVIN = IOBJ                                                  28
      MAJOR = MAJOR + 1                                            29
      CALL TICTAC(IFINAL)                                          30
      TIMER1=(IFINAL-INCIAL)/1000.                                 31
      TIMER2 = TIMER2 + TIMER1                                     32
      INCIAL = IFINAL                                              33
      WRITE(6,355)MAJOR                                            34
355   FORMAT(///,40X,'BEGINNING MAJOR CYCLE ',I4)                  35
      WRITE(6,358)TIMER1,TIMER2                                    36
358   FORMAT(//,20X,'TIME FOR PREVIOUS CYCLE = ',                  37
      F6.2,' SEC.',10X,'TIMIN MASTER  =',F8.2,' SEC')
      IF (MAJOR .EQ. ICPI1) GO TO 380                              38
      DNEW = 0.                                                    39
      DO 360 I=1,LULO                                              40
      IF(MSTATU(I) .NE. QBL)GO TO 380                              41
360   DNEW = DNEW + DEOB(I)                                        42
      IF(DNEW .LE. DOLD)GO TO 380                                  43
      DOLD = DNEW                                                  44
380   CALL NORMAL                                                  45
      ICPI = ITEMP                                                 46
      IF (KCYC .NE. ICLAST) GO TO 384                              47
```

```
      KCYC = KCYC -1                                                    48
      IF (ISAVE .EQ. 0) GO TO 10000                                     49
      KD = NOLA(1,LMAXP1)                                               50
      NOLA(2,LMAXP1) = LE(NETA+1) - 1                                   51
      WRITE(9'KD)                                                       52
      DOLD,DEOB,TIMER2,NOLA,KCYC,
      NCOLO,NROWO,ICOLS,MSTATU
      KD = NOLA(3,LMAXP1)                                               53
      IAUX = NOLA(2,LMAXP1)                                             54
      KIKA = 1                                                          55
      CALL VECTOR(KIKA,A,IA,NB,IAUX)                                    56
      WRITE (6,6020)                                                    57
6020  FORMAT(//'0PROBLEM SAVED FOR RESTART')                            58
      GO TO 10000                                                       59
 384  IF(MSTAT .EQ. QU)GO TO 10000                                      60
      DEOB(LULO) = -X(IOBJ)                                             61
      MSTATU(LULO) = MSTAT                                              62
      IF(MSTAT .NE. QBL .OR. DOLD .LE. -1.E10)GO TO 388                 63
      DNEW =(DEOB(LULO) - DOLD)/DABS(X(IOBJ))                           64
      WRITE(6,385)DOLD,DNEW                                             65
 385  FORMAT(//,3X,'LOWER BOUND  = ',F16.8,//1H ,'                      66
      RELATIVE DIFFERENCE = ',F16.8)
      IF(DNEW .LT. RSTOP)GO TO 10000                                    67
 388  IF (MAJOR .EQ. ICPI) READ(5,5000) (YA(I,6),I=1,NROW)              68
5000  FORMAT(6G12.6)                                                    69
```

```
      WRITE (6,6000)                                                    70
6000  FORMAT(/'0MASTER DUAL VARIABLES'/)                                71
      WRITE (6,6010) ((I,YA(I,6)),I=1,NROW)                             72
6010  FORMAT(6(1X,I3,1X,G12.6))                                         73
      IF(LMAX .EQ. 0)GO TO 1000                                         74
      IF(INCO .NE. -1)GO TO 390                                         75
      INCO = 0                                                          76
 390  IF(ITCNT .EQ. ITMA .AND. ITCNT .NE. 0)GO TO 1000                  77
      DO 400 I=1,LMAX                                                   78
 400  MSTATU(I) = QF                                                    79
      IF(INDEX .EQ. 0)GO TO 315                                         80
      GO TO 1000                                                        81
```

```
C*********************************************************************
C
C     SOLVSUB :
C
C     YTEMP(i)    : Contains dual variable of the i^th convexity constraint
C     LSUB        : Problem index of the sub-problem being solved.
C     NCASUB      : Number of proposals made by the sub-problem being
C                   solved.
C     ICASUB(i)   : Number of proposals made by the i^th sub-problem in
C                   the current cycle.
C     MAXCA       : Maximum number of proposals that can be made by a
```

```
C                    sub-problem in a cycle.
C     NCAND          : Number of proposals generated in the current cycle.
C     MAXNCA         : Once NCAND ≥ MAXNCA the current cycle is
C                      interrupted.
C     ICAND          : Pointer toward NCAND for the proposal index of the
C                      last proposal generated by the sub-problem previously
C                      treated.
C     KSTR           : Strategy parameter. When 5 indicates that the
C                      sub-problem would be solved to optimality and only the
C                      last MAXCA proposals are sent. When 4 indicates that
C                      a sub-problem is allowed to solve upto MAXCA
C                      proposals. If the sub-problem is still not optimized a
C                      new cycle is begun with the same sub-problem as the
C                      first sub-problem.
C     INDEX          : Flag used when KSTR is 4. In this case, a cycle is
C                      interrupted when INDEX is set to 0.
C     MSTAT          : Status of the problem.
C     DEOB(i)        : Objective value for sub-problem i minus the dual
C                      variable of the i^th convexity constraint.
C
C     - Variables are initialized. [lines 84-88]
C     - The dual variabel of the convexity constraint is saved in YTEMP(*).
C       [lines 82-83]
C     - NORMAL is called to solve the sub-problem and generate proposals.
C       [line 89]
```

```
C       - The number of proposals generated by the sub-problem is recorded
C         [line 90]
C       - If there are any unbounded proposals then to discover other unbounded
C         columns, NORMAL would have set MSTAT to 'QF' (feasible). Reset
C         MSTAT to 'QU' (unbounded).
C       - If KSTR is 5 then the proposals, if any, are written on an unit 8 work
C         file. [lines 97-100]
C       - If the sub-problem is infeasible then STOP since the whole problem is
C         infeasible. [line 101]
C         Otherwise proceed.
C       - The current cycle would be interrupted :
C               - If KSTR is 4 and the sub-problem is not optimized then INDEX
C                 is set to zero. [lines 106,108]
C                 A new cycle is started by solving the master problem.
C                 [lines 109-110]
C                 Now, since INDEX is zero, after solving the master problem,
C                 the same sub-problem would be the first sub-problem of the
C                 new cycle. [line 80]
C               - If the number of proposals generated in the current cycle is ≥
C                 MAXNCA the cycle is interrupted except if the current
C                 sub-problem is the last sub-problem. [line 107]
C
C*****************************************************************

 500  DO 520 I=1,LMAX                                          82
```

```
520   YTEMP(I) = YA(NROWO+I,6)                                  83
      NCASUB = 0                                                84
      DOBMA = 0.0                                               85
      ISUNB = 0                                                 86
      NCUNB = 0                                                 87
      ITSLCA =-5                                                88
      CALL NORMAL                                               89
      ICASUB(LSUB) = NCASUB                                     90
      IF (ISUNB .LE. 0) GO TO 530                               91
C
      DO 525 I=1,ISUNB                                          92
      KINBAS(ICUNB(I)) = 0                                      93
525   ICUNB(I) = 0                                              94
C
      MSTAT = QU                                                95
530   CONTINUE                                                  96
      IF (KSTR .NE. 5 .OR. NCASUB .EQ. 0) GO TO 550             97
      IPROS = 0                                                 98
      WRITE (8) ((Z(I,J),I=1,NZMAX),J=1,NCASUB)                 99
      ICAND = ICAND + NCASUB                                    100
550   IF (MSTAT .EQ. QN) GO TO 10000                            101
      MSTATU(LSUB) = MSTAT                                      102
      DEOB(LSUB) = DOB + YTEMP(LSUB)                            103
      DO 620 I=1,LMAX                                           104
620   YA((NROWO+I),6) = YTEMP(I)                                105
```

```
      IF(KSTR .EQ. 4 .AND. MSTAT .NE. QBL)GO TO 700                106
      IF(NCAND.LT.MAXNCA .OR. KKK.EQ.(KK - 1))GOTO 1000 107
  700 IF(KSTR .EQ. 4 .AND. MSTAT .NE. QBL)INDEX = 0                108
      LSUB = 0                                                     109
      GO TO 350                                                    110
 1000 CONTINUE                                                     111
      GO TO 300                                                    112
10000 IF (LMAX .EQ. 0) GO TO 9999                                  113
      WRITE(6,6990) IPROP                                          114
 6990 FORMAT('1PROPOSAL GENERATION STATISTICS'/                    115
     '0CRITERION',10X,'R   F   A   B   O   U'/                     116
     ' NO PROPOSALS  ',6I5)                                        117
 9999 IF (ISOL .NE. 1) STOP                                        118
      RETURN                                                       119
      END                                                          120
```

3.12. Subroutine NORMAL

3.12.1. Major Task :

Solve an LP using the RSM.

3.12.2. Inputs :

The index of the problem (or sub-problem) number to be solved.

3.12.3. Description :

NORMAL is called by INDATA or MASTER whenever a problem has to be solved.

Since it is not possible to store all the problems in core at the same time, all the data relevant to the problem are stored on an unit 9 direct access device (DAD). Whenever a problem has to be solved NORMAL calls CHANGE. If the problem is not in core then CHANGE first writes the problem currently in core to DAD and reads the problem that needs to be solved from DAD. Also, if there is no storage available in the eta file, CHANGE sets the number of etas (NETA) to zero. Once the problem is in core NORMAL calls INVERT if

(a) the number of etas NETA is zero **or**

(b) the infinity norm of the residual rhs exceeds 10^{-4} (cf. § 3.4). The infinity norm is computed by calling CHSOL.

If the problem being solved is the master problem then PACK is called to

incorporate the proposals into the data matrix. Also, since solving the master problem implies the beginning of a new cycle, one should check to ensure that the maximum cycle has not been exceeded before proceeding further. A set of candidate columns can then be selected for multiple pricing as follows:

Case 1 : (Solving master problem) The cost vector is formed in the buffer YA(*,6) by calling FORMC. Since proposals by definition have good reduced costs, a set of columns is selected in PACK as the proposals get incorporated. Thereby, a full scale pricing operation is avoided. However, the user can specify a cycle number at which a complete pricing is desired to start solving the master problem. In this case PRICE is called following BTRAN to select columns for multiple pricing.

Case 2 : (Solving sub-problem in D-W phase 1 or D-W phase 3) The cost vector is formed in YA(*,6) by calling FORMC. Then BTRAN and PRICE are called to select columns for multiple pricing.

Case 3 : (Solving sub-problems in D-W phase 2) Recall, that the sub-problem's feasible cost vector (cf. § 2.6.1) is :

$$[1, -\pi_0^k, 0] \tag{3.33}$$

and the corresponding dual vector is

$$[1. -\pi_0^k, -\pi_r^k] \tag{3.34}$$

Reference to subroutine MASTER will show that at the beginning of a major cycle the master sets up the cost vector given by (3.33) in a buffer

named YA(*,6). If during the course of a major cycle, a sub-problem's solution becomes infeasible, then to set up the cost vector for infeasibilities FORMC would set YA(*,6) to the zero vector. Hence, the cost vector would not be available for solving subsequent sub-problems. To avoid this difficulty a variable KFASE is maintained in D-W phase 2 to indicate the status of the sub-problem's solution and the cost vector is set up accordingly. When a sub-problem's solution is infeasible in D-W phase 2 then KFASE is set to 1 and cost vector is set up in YA(*,5). Otherwise KFASE is set to 2 and the cost vector is formed in YA(*,6). The sequence of calls to select columns for multiple pricing in case 3 is as follows:

(a) Initialize KFASE to 1 and call FORMC.

(b) If the solution is infeasible then the cost vector would have been formed in YA(*,5) since KFASE is 1. Go to d.

(c) If the solution is feasible then KFASE is set to 2 and FORMC is called again to set the third component $-\pi_r^k$ to the zero vector.

(d) Call PRICE following BTRAN to select columns for multiple pricing.

If no columns can be selected then the problem is infeasible if the current solution is infeasible and optimal otherwise. In the latter case CHECK is called to see if a proposal can be made. On the otherhand if columns have been selected then multiple pricing can be started.

Multiple pricing is begun by unpacking the selected columns in a buffer YA and the following is done to each one of those columns. FTRAN is first called to update each candidate column. CHUZR then

(a) determines the allowable increase if the variable corresponding to the candidate column enters,

(b) determines the resulting change in the objective value and

(c) finds out whether the column is unbounded.

Case 1 : (CHUZR identifies the column as unbounded) If the current problem is the master problem then the whole problem is unbounded. If step 1of D-W phase 1 then the proposal would be written to the proposal buffer Z(*,*) in the subroutine INDATA. If the problem is in D-W phase 3 then a proposal need not be formed. Otherwise CHECK is called to ascertain the possibility of making a proposal. CHECK may or may not be able to make a proposal without exceeding the maximum limit on the number of proposals that can be generated by a sub-problem. In the latter case the solution of the sub-problem is interrupted and control is returned back to NORMAL. In the former case, the column is fixed at zero and the optimization of the restricted sub-problem is pursued from the start of the simplex cycle. This is done (cf. § 2.6.2) to enable other extreme rays to be discovered.

Case 2 : (Not an unbounded column) The best change in the objective value is updated. This procedure is repeated for the next column until all columns in the buffer YA(*,*) have been considered.

Therefore, at the end of one pass through all candidate columns in YA(*,*), the one that provided the best change DY in the objective value would be the entering column provided none of the columns were unbounded. To find the pivot row, CHUZR is called again to pick the maximum pivot among those columns which provide the same change DY in the objective value. The solution is updated and a basis change is effected by marking the leaving column as non-basic and the entering column as basic. Also, the column that just entered the basis is marked to

prevent it from being considered for entering the basis during the next multiple pricing pass. CHECK is called again and two cases result form this call.

Case 1 : (Don't continue with multiple pricing) A proposal can be generated provided some tests (cf. § 2.6.2) are met by a sub-problem in D-W phase 2 and a strategy of generating intermediate proposals (cf. § 2.6.2 and users guide) is chosen by the user. If the maximum limit on the number of proposals has been exceeded in the process of generating a proposal then the solution of the problem has to be terminated. CHECK sets a flag KOUT to signify this condition. Often an INVERT may be done in order to generate a more accurate proposal. In this case it is better to interrupt the multiple pricing cycle and start from the beginning of the simplex cycle. CHECK sets the flag IRTN to denote this condition. If both KOUT and IRTN have not been set then multiple pricing is continued as follows after a proposal is made.

Case 2 : (Continue with multiple pricing) Conceivably the dual price after the basis change done before the call to CHECK would not be very different from that of the previous basis. Therefore, all columns in YA(*,*) except the ones which entered the basis may still price out good. To verify this, one has to update the column with the lastest eta and find the reduced cost. This is done efficiently be calling FTRAN with the parameter IPAR set to three and the next multiple pricing pass can be started.

If all the columns selected in the buffer YA(*,*) have either unfavorable reduced costs or have already been pivoted in, then a fresh simplex iteration is started.

NORMAL therefore, comprises the following steps

3.12.4. Steps :

1. Read the problem to core.
2. Invert if necessary.
3. Pack proposals if necessary and select a pool of columns for multiple pricing. If no columns can be selected decide if the problem is optimal or unbounded and go to step 6
4. Perform multiple pricing.
5. Go to step 2 to begin the next simplex iteration.
6. If the problem is optimal then call CHECK. RETURN.

3.12.5. Deatailed Comments :

```
C*****************************************************************
C
C       Step 1 :
C
C       LSUP            : Index of the problem that needs to be in core.
C
C       1. Set LSUP to the index of the problem that needs to be solved. [line 6]
C          Initialize other varaibles. [lines 1-5,8-11].
C       2. Call CHANGE to write into DAD the problem currently in-core and
C          read the LSUP^th problem into core from DAD. [line 7]
C
C*****************************************************************
```

```
      IPAR = 1                                                    1
      ITEMP = 0                                                   2
      INVF = INVFRQ                                               3
      IF (LSUB .EQ. 0) INVF = INVFMA                              4
      CALL TICTAC(ITIM)                                           5
      LSUP = LSUB                                                 6
      CALL CHANGE(LSUP)                                           7
      KFASE = 2                                                   8
      NCASUB = 0                                                  9
      JTINV = 0                                                   10
      ITMA = ITCNT                                                11

C*********************************************************************
C
C     Step 2 :
C
C     NETA        : Number of etas
C     ERMAX       : Residual rhs ∂b.
C     YA(*,3)     : Contains b to start with and contains ∂b after CHSOL.
C     NROWO       : Number of coupling rows.
C
C     If NETA is 0 then call INVERT. [lines 12,30].
C     Otherwise, if the residual rhs is ≥ 10**-4 then call INVERT. The residual
C     rhs ∂b is computed by CHSOL which needs as inputs the rhs and the
C     computed solution x+∂x (cf. 3.10 in § 3.4). These inputs are
```

```
C       provided as follows.
C               - The rhs supplied by the user are stored in YA(*,3) by calling
C                 UNPACK.  [line 13]
C                 If it is the master problem then 1's are to be stoed in rhs of the
C                 convexity rows.  [lines 16-18]
C                 If the program is executing phase 3 then the allocation for the
C                 sub-problem, computed by (2.74), is stored in the buffer
C                 Z(*,1).  This is copied to the appropriate positions of YA in this
C                 case.  [lines 20-22]
C               - Call FTRAN to compute x+∂x and copy it to the array X.
C                 [lines 22-26].
C       Call CHSOL and then call INVERT if necessary.  [lines 27-28]
C
C*****************************************************************
        IF(NETA .EQ. 0)GO TO 1000                                   12
        CALL UNPACK(RHSCOL,3)                                       13
         IF(LMAX .EQ. 0)GO TO 4100                                  14
        IF(LSUB .GT. 0 )GO TO 4000                                  15
        DO 3800 I=1,LMAX                                            16
3800    YA((NROWO+I),3) = 1.                                        17
        GO TO 4100                                                  18
4000    IF(MAST .NE. 2)GO TO 4100                                   19
        DO 4050 I=1,NROWO                                           20
4050    YA(I,3) = Z(I,1)                                            21
4100    DO 4200 I=1,NROW                                            22
```

```
4200  YA(I,2) = YA(I,3)                                          23
      CALL FTRAN(1,2)                                            24
      DO 4400 I=1,NROW                                           25
4400  X(I) = YA(I,2)                                             26
      CALL CHSOL                                                 27
      IF(ERMAX .LE. 1.E-4) GO TO 1060                            28
1000  NT = NT + JTINV                                            29
      CALL INVERT                                                30
      ITSINV = 0                                                 31

C*****************************************************************
C
C     Step 3 :
C
C     MAST      : Indicates the stage of the problem.  If 0 then executes
C                 step 1 of D-W phase 1.  If 1 then executes step 2 of
C                 phase 1 or D-W phase 2.  If 2 then executes phase 3
C     NCAND     : Number of proposals generated in the current cycle.
C     KCYC      : Cycle number
C     ICLAST    : Maximum cycle limit
C     KFASE     : Normally set to 2 and in this case the cost vector is
C                 formed in YA(*,6).  When a sub-problem's solution
C                 becomes infeasible in D-W phase 2 then KFASE is set
C                 to 1 and now the cost vector is formed in YA(*,5).
C     MSTAT     : Status of the problem.  (cf. variable dictionary)
```

```
C
C     1. If there are proposals and the master problem is being solved then call
C        PACK to incorporate the proposals into the data array A.
C        [lines 33,36-37]
C     2. If the cycle number has exceeded the maximum cycle limit then
C        RETURN. [lines 38,42]
C     3. Candidate columns for multiple pricing are selected as follows
C              Case 1 : (Master problem)
C              - Call FORMC to form the cost vector in YA(*,6)
C              - Candidate columns for multiple pricing are selected by PACK
C                except if special prices are to be computed. In this case BTRAN
C                and PRICE are called. [lines 46-47,58-59].
C              Case 2 : (Sub-problem in D-W phase 1 or phase 3)
C              - Call FORMC, BTRAN, PRICE. [line 55-56,58-59]
C              Case 3 : (Sub-problem in D-W phase 2)
C              Refer to case 3 of § 3.12.3.
C              (a) [lines 48-49]
C              (b) [line 50]
C              (c) [lines 51-55]
C              (d) [lines 58-59]
C     4. If no candidate columns were selected for multiple pricing and the
C        solution is infeasible then set MSTAT to 'QN' and RETURN.
C        [lines 60-61]
C        If no candidate columns were selected for multiple pricing and the
C        solution is feasible then the problem is optimal. Hence, set MSTAT to
```

```
C       'QBL' and call CHECK to see if a proposal can be made.
C       [lines 60-63,138].
C       Else go to step 4.
C
C*********************************************************************
      IF(ITCNT .NE. ITMA)GO TO 1070                                  32
1060  IF(LSUB .EQ. 0 .AND. NCAND .GE. 1)GO TO 1070                   33
      CALL ITEROP(0,IYIV)                                            34
1070  IF(MAST .NE. 1)GO TO 1500                                      35
      IF(LSUB .GE. 1)GO TO 1100                                      36
      IF(NCAND .NE. 0)GO TO 1075                                     37
      IF (KCYC .LT. ICLAST) GO TO 1500                               38
      RETURN                                                         39
1075  CALL PACK                                                      40
      NCAND = 0                                                      41
      IF (KCYC .LT. ICLAST) GO TO 1080                               42
      RETURN                                                         43
1080  CALL ITEROP(0,IYIV)                                            44
      CALL FORMC                                                     45
      IF (KCYC .EQ. ICPI) GO TO 1800                                 46
      GO TO 1900                                                     47
1100  KFASE = 1                                                      48
      CALL FORMC                                                     49
      IF(MSTAT .EQ. QI)GO TO 1800                                    50
1200  KFASE = 2                                                      51
```

```
      DOB = 0.                                                          52
      DO 1400 I=1,NROWO                                                 53
1400  DOB = DOB - X(I)*YA(I,6)                                          54
1500  CALL FORMC                                                        55
      IF(KFASE .NE. 1)GO TO 1800                                        56
      IF(MSTAT .EQ. QF)GO TO 1200                                       57
1800  CALL BTRAN                                                        58
      CALL PRICE                                                        59
1900  IF(MULT .GT. 0)GO TO 3000                                         60
      IF (MSTAT .EQ. QI   ) GO TO 2000                                  61
      MSTAT = QBL                                                       62
      GO TO 5810                                                        63
2000  MSTAT = QN                                                        64
      GO TO 6000                                                        65

C*********************************************************************
C
C     Step 4 :
C
C     YA(*,*)       : Buffer containing candidate entering columns.
C     JCOLP(i)      : Column index in the data matrix A of the column
C                      YA(*,i)
C     LSUB          : Problem index of the problem being solved.
C     MAST          : Indicates the stage of the problem.
C     MAXCA         : Maximum limit on the number of proposals that can be
```

```
C                made by a sub-problem in a cycle.
C     KSTR       : Strategy parameter. When 5 indicates that only the last
C                  maxca proposals are sent to the master problem.
C     NCASUB     : Number of proposals generated by the LSUB^th
C                  sub-problem in the current cycle.
C     KVEC       : Candidate column of YA that provides the best change
C                  among all the column considered.
C     MSTAT      : Status of the problem.
C     KTYPE,NTYPE: Ratio type. (cf. subroutine CHUZR)
C     DE         : Allowable increase in the variable corresponding to the
C                  candidate column.
C     DY         : Allowable change in the objective value if the variable
C                  corresponding to the candidate column enters.
C     IROWP      : Pivot row for the candidate column that provides the
C                  best change in the objective value.
C     IPAR       : Parameter indicating type of FTRAN
C     JENTRY     : Parameter indicating type of CHUZR.
C     KOUT       : Flag set by CHECK to indicate if he solution of the
C                  problem is to be terminated for the current cycle.
C     IRTN       : Flag set by CHECK to indicate if the multiple pricing
C                  cycle is to be interrupted.
C
C     (a) Unpack all candidate columns selected for multiple pricing in the
C         buffer YA. [lines 66-68].
C     (b) If the candidate column has already entered the basis (KINBAS(i) ≠ 0)
```

C then it cannot enter the basis again. Hence, take up the next column.
C Otherwise call FTRAN to update the candidate column beingconsidered.
C [lines 74-76].
C (c) If the reduced cost of the column is not good then it cannot enter.
C Hence, take up the next column. Otherwise call CHUZR to determine
C the allowable increase if the column corresponding to the candidate
C column enters. CHUZR also computes the resulting change in the
C objective value and find out if the call is unbounded. In addition
C CHUZR finds the type of the pivot row if the column were to enter.
C [lines 77-78]
C *Case 1* : (Unbounded Column identified)
C - Call CHECK if a sub-problem is being solved in D-W phase 2
C [lines 79-82].
C - If a proposal is made without exceeding the limit on the number
C of proposals generated by a sub-problem then reset MSTAT to
C QU and fix the column at zero. Interrupt the multiple pricing
C cycle and start a new Simplex iteration. [lines 83-92]
C Otherwise RETURN.
C *Case 2* : (Not an unbounded column)
C - Update the best change in the objective value. [lines 95-96]
C - Update the column index of the candidate column that yielded
C this change in the objective value. [line 99]
C - Update the allowable change if the variable corresponding to
C the candidate column enters. [lines 100]
C (d) Take up the next column till all columns in YA have been considered.

```
C
C     (e) Set KVEC to the column of YA providing the best change in the
C         objective value. [line 104].
C         If the ratio type (KTYPE) is not 2 then call CHUZR with JENTRY set
C         to 2 to find the maximum pivot row. [lines 105,108].
C         Update the colution by calling UPBETA. [line 111]
C     (f) Update the basis. [lines 112-117].
C     (g) Call CHECK to see if a proposal can be made. [line 118]
C             Case 1 : (Don't continue with multiple pricing)
C                     Case 1.1 : (terminate the solution of the problem)
C                     - If KOUT is 1 then the solution of the problem is to be
C                       terminated for the current cycle. {lines 120,129].
C                     - If there is no space in the eta file then write the etas
C                       (Call WRETA) if an INVERT was done in CHECK.
C                       RETURN. [lines 129-130].
C                     Case 1.2 : (Terminate multiple pricing)
C                     - If IRTN is 1 then an INVERT was done in CHECK.
C                       Hence, start a new simplex cycle. [lines 120-121].
C             Case 2 : (Continue with multiple pricing)
C             - Call WRETA to write the etas generated after the last basis
C               update. [line 123].
C             - Set IPAR to 3 so that the reduced cost would also be computed.
C               [line 125]
C     (h) Go to (b) to take up the ramaining candidate columns of YA.
C     (i) If all columns in YA do not have good reduced cost or have already
```

```
C          entered the basis then go to step 2.  [lines 134-136].
C
C*****************************************************************
 3000  DO 3200 I=1,MULT                                          66
       J = JCOLP(I)                                              67
 3200  CALL UNPACK(J,I)                                          68
       IPAR = 1                                                  69
       DO 5000 K=1,MULT                                          70
       DELTA = 1.5                                               71
       DO 4500 I=1,MULT                                          72
       ITEMP = I                                                 73
       IF(JCOLP(I) .EQ. 0)GO TO 4500                             74
       IF(KINBAS(JCOLP(I)) .NE. 0)GO TO 4500                     75
       CALL FTRAN(IPAR,I)                                        76
       IF(DCMIN(I) .GE. -ZTCOST)GO TO 4500                       77
       CALL CHUZR(I,NTYPE,1)                                     78
       IF (MSTAT .NE. QU) GO TO 4460                             79
       IF (LSUB .EQ. 0) GO TO 5800                               80
       IF (MAST .NE. 1) GO TO 5800                               81
       CALL CHECK(KOUT,ITEMP,IRTN)                               82
       IF (KSTR .EQ. 5 .AND. NCUNB .EQ. MAXCA) GO TO 4450        83
       IF(KSTR .NE. 5 .AND. NCASUB .GE. NC(KKK))GO TO 4450       84
       IF (ISUNB .GE. 7) GO TO 4450                              85
       ISUNB = ISUNB + 1                                         86
       ICUNB(ISUNB) = JCOLP(I)                                   87
```

```
      KINBAS(JCOLP(I)) = -1                                    88
      JCOLP(I) = 0                                             89
      MSTAT = QF                                               90
      IF (IRTN .EQ. 1) GO TO 1800                              91
      GO TO 1500                                               92
4450  IYIV = ITEMP                                             93
      GO TO 6000                                               94
4460  IF(DELTA .LE. DY)GO TO 4500                              95
      DELTA = DY                                               96
      KTYPE = NTYPE                                            97
      MROWP = IROWP                                            98
      IYIV = I                                                 99
      DTEMP = DE                                               100
4500  CONTINUE                                                 101
      IF(DELTA .GT. .1)GO TO 5200                              102
      DE = DTEMP                                               103
      KVEC = IYIV                                              104
      IF(KTYPE .NE. 2)GO TO 4600                               105
      IROWP = MROWP                                            106
      GO TO 4700                                               107
4600  CALL CHUZR(KVEC,KTYPE,2)                                 108
4700  IVIN = JCOLP(IYIV)                                       109
      IVOUT = JH(IROWP)                                        110
      CALL UPBETA(IYIV)                                        111
      KINBAS(IVIN) = IROWP                                     112
```

```
      KINBAS(IVOUT) = 0                                          113
      JH(IROWP) = IVIN                                           114
      ITCNT = ITCNT + 1                                          115
      ITSINV = ITSINV + 1                                        116
      ITEMP = IYIV                                               117
      CALL CHECK(KOUT,ITEMP,IRTN)                                118
      CALL ITEROP(IPAR,IYIV)                                     119
      IF(KOUT .EQ. 1)GO TO 4800                                  120
      IF (IRTN .EQ. 1) GO TO 1800                                121
      IF(NELEM .GT. (NEMAX - NROW))GO TO 1000                    122
      CALL WRETA(IYIV)                                           123
      JCOLP(IYIV) = 0                                            124
      IF(IPAR .EQ. 1)IPAR = 3                                    125
      GO TO 5000                                                 126
4800  IF (NELEM .GT. NEMAX-NROW) GO TO 4850                      127
      IF (IRTN .EQ.1) GO TO 6010                                 128
      CALL WRETA(IYIV)                                           129
      GO TO 6010                                                 130
4850  NETA = 0                                                   131
      GO TO 6010                                                 132
5000  CONTINUE                                                   133
5200  IF(NELEM .GT. (NEMAX - NROW))GO TO 1000                    134
      TIMER = JTINV                                              135
      IF(ITSINV - INVF)1500,1000,1000                            136
5800  IYIV = ITEMP                                               137
```

```
5810  CALL CHECK(KOUT,ITEMP,IRTN)                    138
6000  CALL ITEROP(IPAR,IYIV)                         139
6010  NT = NT + JTINV                                140
      RETURN                                         141
      END                                            142
```

3.13. Subroutine PACK :

3.13.1. Major Task :

Packs a proposal into the data matrix and finds the reduced cost of KMULT (cf variable dictionary) of these columns.

3.13.2. Inputs :

1. Proposal buffer called henceforth the **Z(*,*).**
2. The unit 8 work file having proposals that cannot be accomodated in Z(*,*).

3.13.3. Called By :

INDATA and NORMAL.

3.13.4. Description :

In order to solve the master problem all proposal columns generated in the current cycle should be processed by incorpoarting them in the data array A(*) of the master problem.

Proposals are incorporated one at a time from a proposal buffer called Z. Let NPROS be the maximum number of columns that can be accomodated by Z. Z is set up depending on the strategy chosen. If the user chooses a strategy of permitting partial cycles (cf. section 2.6.2) then proposals are stored in and

processed from Z one problem at a time as explained below. Initially Z contains the proposals from the problem that first generated proposals after a cycle interruption. Even if partial cycles are not permitted, it possible for a problem to generate more than one proposal. Therefore, it is not difficult to imagine a situation when the number of proposals generated exceeds the dimensional limit of NPROS imposed by Z. In this case the current contents of Z are written to the end of the unit 8 work file and the first NPROS proposals are then read into Z from the begining of the work file. Once these proposals are incorporated, the next NPROS proposals are read from the work file and this continues until all proposals have been processed. Incorporating a proposal is done as axplained below.

Before a proposal is incorporated, one should check for availability of space, failing which some of the "unwanted" columns in A should be purged. A column is labelled as "wanted" if it is basic or if it is non-basic with reduced cost "better" that a threshold value or if it has been generated in the current cycle. A column that is not wanted is "unwanted". Purging involves scanning through all columns in A to determine if a certain column is "wanted" or "unwanted". An unwanted column in A is purged by overwriting on it the next wanted column as illustrated in the figure (3.16). If the space created by the current pass of purging is not enough, then the threshold reduced cost used to label a column as unwanted is tightened further and the process is repeated again.

When there is space to pack the proposal, it is scaled if it is an extreme point proposal and then incorporated into the data array A(*) of the master problem.

If the maximum number of columns that can be chosen for multiple pricing

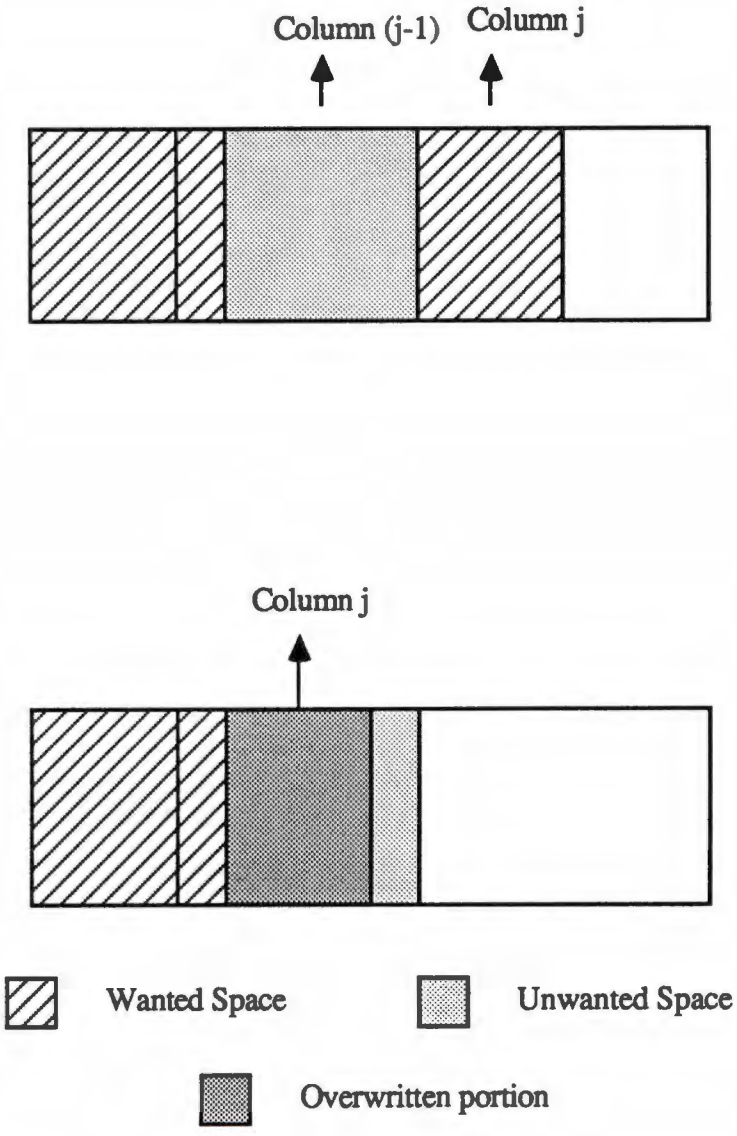


Figure 3.16 : Data array before and after purging.

(cf. section 2.2.4) has not been exceeded, then the current column is selected for multiple pricing provided its reduced cost is "better" than zero. Otherwise it is chosen if its reduced cost is "better" than the "worst" reduced cost of columns already chosen for multiple pricing.

The steps in PACK are therefore as listed below.

3.13.5. Steps :

1. Set up the Z-buffer.
2. For each proposal do
   (a) Check if purging is required and if so goto step 2b. Else goto step 2c.
   (b) Purge if required.
   (c) If the contents of the current Z-buffer have been processed then read some more co;umns from the work file.
   (d) Scale the column if necessary and incorporate it into the data matrix.
   (e) Select columns for multiple pricing.

   End For.

3.13.6. Detailed Comments :

```
C*****************************************************************
C
C     Step 1 :
C
C     KSTR          : When 5 indicates that a partial cycle strategy is adopted.
```

C Here a major cycle is interrupted once the total number
C of proposals exceeds the supplied limit. The next major
C cycle is started by solving the next sub-problem. For
C example, if the limit is exceeded while solving the i^{th}
C sub-problem, then in the next cycle sub-problem 1
C through i would not have generated any proposal.
C MAST : When 0 implies that the program is executing step 1 of
C D-W phase 1 and during this phase each problem
C generates only one proposal.
C
C *Case 1* : (KSTR = 5 and MAST ≠ 0) In this case a cycle interruption
C could have occured and some problems would have generated more than
C one proposal.
C - Scan through all problems to see which one first generated proposals
C after a cycle interruption. Let it be the L^{th} problem. [lines 11-13].
C - Set JEND to the number of proposals generated by thr L^{th} problem.
C [line 14].
C - Read JEND proposals into Z. [line 15]
C - Set ICASUB(L) to zero to ensure that the next time the proposal from
C the same problem is not read again. [line16].
C - Set IPROS to zero indicating that the first proposal in Z can now be
C processed if there is space in A. [line17]
C *Case 2* : (Otherwise.) If NCAND > NPROS then more proposals have
C been generated than can be accomodated in Z. Therefore, write the
C contents of Z to the end of the unit 8 work file. [lines 4-5].

```
C       - Read into Z the first NPROS proposals.  [lines 6-7]
C       - Set IPROS to zero, indicating that the first proposal in Z can now be
C         processed provided there is space in A.  [line 17]
C
C*****************************************************************
      MULT = 0                                                    1
      DCOST = - ZTCOST                                            2
      IF (KSTR .EQ. 5 .AND. MAST .NE. 0) GO TO 90                 3
      IF (NCAND .LE. NPROS) GO TO 100                             4
      WRITE (8) Z                                                 5
      REWIND 8                                                    6
      READ (8) Z                                                  7
      GO TO 100                                                   8
 90   L = 1                                                       9
      REWIND 8                                                    10
 92   IF (ICASUB(L) .NE. 0) GO TO 94                              11
      L = L+ 1                                                    12
      GO TO 92                                                    13
 94   JEND = ICASUB(L)                                            14
      READ (8) ((Z(I,J),I=1,NZMAX),J=1,JEND)                      15
      ICASUB(L) = 0                                               16
100   IPROS = 0                                                   17
      DO 2000 K=1,NCAND                                           18
```

```
C*********************************************************************
C
C       Step 2a :
C
C       - If there is no space in the data array A(*)  or if the inclusion of the
C         proposal  means exceeding the maximum number of columns then go to
C         step 2b to purge unwanted columns.  Else proceed.  [line 19]
C       - Increment IPROS since one more column can be processed.  [line 20]
C
C*********************************************************************
      IF ((LA(NCOL+1) .GT. LE(1)-NROW) .OR.                          19
     1(NCOL+1 .GT. NLAMAX)) GO TO 1000
  200 IPROS = IPROS + 1                                              20

C*********************************************************************
C
C       Step 2c :
C
C       Case 1 : (KSTR = 5 and MAST ≠ 0)  If all proposals in Z have not been
C         processed (IPROS ≤ JEND) then go to step 2d to process the proposal
C         currently pointed at in Z.
C       - If IPROS > JEND then all proposals in Z have been processed.  So
C         read into Z the proposals from the problem that next generated proposals
C         as explained before.  [lines 22-28].
C       Case 2 : (Otherwise)  If IPROS  < NPROS then all proposals in Z have
```

```
C       been incorporated.  So read into Z the next NPROS proposals.
C       [lines 29-30].
C
C*********************************************************************
        IF (KSTR .NE. 5 .OR. MAST .NE. 1) GO TO 208                    21
        IF (IPROS .LE. JEND) GO TO 210                                 22
  202   L = L + 1                                                      23
        IF (ICASUB(L) .EQ.0) GO TO 202                                 24
        JEND = ICASUB(L)                                               25
        ICASUB(L) = 0                                                  26
        READ(8) ((Z(I,J),I=1,NZMAX),J=1,JEND)                          27
        GO TO 209                                                      28
  208   IF (IPROS .LE. NPROS) GO TO 210                                29
        READ (8) Z                                                     30
  209   IPROS = 1                                                      31

C*********************************************************************
C
C       Step 2d :
C
C       Case 1 : (Extreme point implied by NAME(*) > 0)
C       (a) The 2-norm of the proposal vector is found.  [lines 34-36].
C           Using this 2-norm the scaling factor DSY is fixed.  [lines 38-39]
C       (b) Each element in the proposal is first scaled and tested to see if it is not
C           zero.  If non-zero, NELA, the number of non-zero elements in A is
```

```
C          incremented and element is then incorporated into the data A(*).
C          [lines 48-50]
C       (c) The sub-problem origin of the proposal is marked so that one can keep
C          track of  the number of proposals generated by a certain problem.
C          [line 55].
C       (d) Since it is an extreme point proposal there should be a 1 in the
C          convexity row's position.  Incorporate the scaled value of 1 into A.
C          [lines 56-59]
C       (e) In CHECK Z(NROW+1,*) was set to the reduced cost of the column.
C          Set DSUM to the scaled value of the reduced cost so that a decision
C          can be made in step 2e about selecting the column for multiple pricing.
C          [line 61]
C       Case 2 : (extreme ray implied by NAME(*) ≤ 0)
C       (a) Here since scaling is not done the value of the scaling factor DSY is 1.
C       (b) Each non-zero entry in the proposal is incorporated as explained in
C          case 1.  [lines 43-51]
C       (c) Mark the sub-problem origin of the proposal.  [line 53].
C       (d) Same as (e) of case1.
C
C*************************************************************************
  210   DSUM = 1.                                                    32
        IF(NAME(K) .LE. 0)  GO TO 312                                33
        DO 300 I=1,NROWO                                             34
  300   DSUM = DSUM + Z(I,IPROS)*Z(I,IPROS)                          35
        DSUM = DSQRT(DSUM)                                           36
```

```
      IF (NAME(K) .LE. 0 ) GO TO 312                    37
      DSY = 10./DSUM                                    38
      IF (DSUM .GT. 1000.) DSY = 0.01                   39
      GO TO 315                                         40
312   DSY = 1.                                          41
      IF (DSUM .LT. 1.) DSY = 10./DSUM                  42
315   NELA = LA(NCOL+1) - 1                             43
      DO 500 I=1,NROWO                                  44
      DSA = Z(I,IPROS)*DSY                              45
      IF(DABS(DSA) .LE. 1.E-15)GO TO 500                46
      NELA = NELA + 1                                   47
      IA(NELA) = I                                      48
      A(NELA) = DSA                                     49
500   CONTINUE                                          50
      NCOL = NCOL + 1                                   51
      IF(NAME(K) .GE. 0)GO TO 600                       52
      ICOLS(NCOL) = -NAME(K)                            53
      GO TO 800                                         54
600   ICOLS(NCOL) = NAME(K)                             55
      NELA = NELA + 1                                   56
      IA(NELA) = NROWO + NAME(K)                        57
      A(NELA) = DSY                                     58
800   LA(NCOL+1) = NELA + 1                             59
      KINBAS(NCOL) = 0                                  60
      DSUM = Z(NROWO+1,IPROS)*DSY                       61
```

```
C*****************************************************************
C
C     Step 2e :
C
C     DCOST      : Threshold reduced cost.  Initialized to zero tolerance.
C     DSUM       : The reduced cost of the column just processed.
C     MULT       : The numbar of columns selected for multiple pricing.
C     KMULT      : Set by the user and indicates the maximum number of
C                  columns that can be selected for multiple pricing.
C     DCMIN      : Array containing the reduced cost of the columns
C                  selected for multiple pricing.
C     JMIN       : The column index of the column with the "worst"
C                  reduced cost among those selected for multiple pricing.
C     JCOLP(i)   : Column index of the i^th column selected for multiple
C                  pricing.
C
C     Case 1 : (Enough columns have not been selected) Check to see if the
C     reduced cost DSUM is < 0 and if so proceed. [line 62].
C     This is done since, as long as enough (KMULT) columns have not been
C     selected any column with reduced cost < 0 is good enough to be selected
C     for multople pricing.  Increment MULT and select this column for
C     multiple pricing. [lines 72-75].
C     If after selecting a column, there are enough columns selected (MULT =
```

```
C       KMULT) then also do case 2.
C       Case 2 : (Enough columns have been selected) In this case a threshold
C       reduced cost of 0 is not good enough since there are KMULT columns
C       with reduced costs "better" than 0. So the threshold reduced cost DCOST
C       is updated to the value of the "worst" reduced cost from among the set of
C       columns selected for multiple pricing. [lines 66-71].
C       A column is selected only if its reduced cost is "better" than the previously
C       updated threshold value DCOST and it replaces the column with the
C       worst reduced cost. [lines 61-65].
C
C*************************************************************************
        IF(DSUM .GE. DCOST)GO TO 2000                                    62
        IF(MULT .LT. KMULT)GO TO 850                                     63
        DCMIN(JMIN) = DSUM                                               64
        JCOLP(JMIN) = NCOL                                               65
  820   DCOST = DSUM                                                     66
        DO 830 J=1,KMULT                                                 67
        IF(DCOST .GE. DCMIN(J))GO TO 830                                 68
        DCOST = DCMIN(J)                                                 69
        JMIN = J                                                         70
  830   CONTINUE                                                         71
        GO TO 2000                                                       72
  850   MULT = MULT + 1                                                  73
        DCMIN(MULT) = DSUM                                               74
        JCOLP(MULT) = NCOL                                               75
```

```
      IF(MULT .LT. KMULT)GO TO 2000                                    76
      JMIN = MULT                                                      77
      GO TO 820                                                        78

C*********************************************************************
C
C     Step 2b :
C
C     DE          : Threshold reduced cost and is initialized to 10.
C     KK          : The total number of columns in A before a fresh pass of
C                   purging begins. Since there are NCOL columns to
C                   start with , KK is initialized to NCOL.
C     NCOL        : The number of columns in A "wanted" at any point.
C                   Since all coupling rows are wanted NCOL is initialized
C                   to the number of coupling rows NCOLO. [line 82]
C     NELA        : The number of elements in A that are wanted. NELA is
C                   therefore initialized to the number of elements in the
C                   coupling columns. [lines 83].
C
C     For each column from NCOLO+1 to KK :
C          Case 1 : (cannot be purged)
C          - If the column is basic [line 88]
C            or if it is generated in this cycle [line 89]
C            or if the reduced cost being computed is "better" than the
C            threshold value [lines 90-93]
```

```
C                then it cannot be purged.
C              - Starting from NELA overwrite the wanted column on an
C                unwanted column, duly updating NELA to reflect the number of
C                wanted elements in A.  [lines 94-97]
C              - Since one more column has been added, increment the number
C                of columns NCOL.  [lines 98].
C              - Since the number of elements and the column index of a column
C                changes due to purging, update LA,ICOLS, KINBAS and JH.
C                [lines98-102].
C              - If the column is selected for multiple pricing update the column
C                index in JCOLP.  [lines104-107].
C              Case 2 : (can be purged)  A column can be purged if it does not
C              meet the tests described in case 1.  Overwriting on it a wanted
C              column effectively purges it.
C       End For.
C       If purging done is not sufficient then tighten the reduced cost threshold
C       and start a fresh purging pass.  [lines 111-113].
C
C*************************************************************************
 1000   DE = 10.                                                        79
        DY = LA(NCOL+1) - LA(NCOLO+1)                                   80
 1050   KK = NCOL                                                       81
        NCOL = NCOLO                                                    82
        NELA = LA(NCOLO+1) - 1                                          83
        MM = NCOLO + 1                                                  84
```

```
      DO 1500 I=MM,KK                                              85
      MU = LA(I+1) - 1                                             86
      NN = LA(I)                                                   87
      IF(KINBAS(I) .NE. 0)GO TO 1150                               88
      IF(I .GT. (KK + 1 -K))GO TO 1150                             89
      DP = 0.                                                      90
      DO 1100 J=NN,MU                                              91
1100  DP = DP + A(J)*YA(IA(J),6)                                   92
      IF(DP .GE. DE)GO TO 1500                                     93
1150  DO 1200 J=NN,MU                                              94
      NELA = NELA + 1                                              95
      IA(NELA) = IA(J)                                             96
1200  A(NELA) = A(J)                                               97
      NCOL = NCOL + 1                                              98
      LA(NCOL + 1) = NELA + 1                                      99
      ICOLS(NCOL) = ICOLS(I)                                       100
      KINBAS(NCOL) = KINBAS(I)                                     101
      IF(KINBAS(NCOL) .GT. 0)JH(KINBAS(NCOL))=NCOL                 102
      IF(I .LE. (KK - K + 1))GO TO 1500                            103
      IF(MULT .EQ. 0)GO TO 1500                                    104
      DO 1400 JA=1,MULT                                            105
      IF(JCOLP(JA) .NE. I)GO TO 1400                               106
      JCOLP(JA) = NCOL                                             107
      GO TO 1500                                                   108
1400  CONTINUE                                                     109
```

```
1500  CONTINUE                                                    110
      DP = (LA(NCOL+1) -LA(NCOLO+1))/DY                           111
      IF(NELA .LT. NEMAX .AND. DP .LT. 0.75)GO TO 200             112
      DE = 0.1*DE - ZTCOST                                        113
      GO TO 1050                                                  114
2000  CONTINUE                                                    115
      IF (NCAND .GT. NPROS .OR. KSTR .EQ. 5) REWIND 8             116
      IPROS = 0                                                   117
      ICAND = 0                                                   118
      RETURN                                                      119
      END                                                         120
```

3.14. Subroutine POLICY :

3.14.1. Major Task :

Sets proposal generation policy.

3.14.2. Outputs :

Sets policy if necessary. Setting policy involves :

1. Setting limts on the maximum number of proposals that can be generated by a sub-problem in a cycle.
2. Setting the order in which sub-problems are to be solved.

3.14.3. Called By :

Called by MASTER for each uninterrupted cycle.

3.14.4. Description :

POLICY first checks to see if policy has to be set and this check is done as follows. If there are no sub-problems then policy need not be set. If all sub-problems are optimal then the policy that was set during a previous call is still valid. Otherwise policy is set based on the value of the control parameter KSTR.

POLICY, therefore consists of the following steps.

3.14.5. Steps :

1. Decide if policy has to be set and if so go to step 2. Else RETURN.

2. Set policy.

3.14.6. Detailed Comments :

```
C*****************************************************************
C
C     Step 1 :
C
C     LMAX          : Maximum number of sub-problems.
C     MSTAT(*)      : Status of the *th sub-problem. QBL indicates optimal,
C                     and QN indicates infeasible.
C     KK            : Maximum number of problems to be solved in a cycle.
C     LS(*)         : Index of the *th sub-problem to be solved. Zero
C                     indicates master problem.
C
C     1. If LMAX is 0 then there are no sub-problems. Therefore, if the master
C        is optimal or infeasible then we are done. Otherwise, KK is set to 1
C        and LS(1) is set to zero indicating that only the master problem is to be
C        solved. If LMAX is greater than 0 then proceed. [lines 2,8-12].
C     2. If all problems are optimal then policy need not be set. Otherwise go to
C        step 2. [lines 3-7].
C*****************************************************************
```

```
      KK = -1                                                    1
      IF(LMAX .EQ. 0)GO TO 500                                   2
      DO 200 I=1,LMAX                                            3
      IF(MSTATU(I) .EQ. QBL)GO TO 200                            4
      GO TO 990                                                  5
  200 CONTINUE                                                   6
      GO TO 10000                                                7
  500 IF(MSTAT .EQ. QBL .OR. MSTAT .EQ. QN)GO TO 10000           8
      KK = 1                                                     9
      LS(1) = 0                                                  10
      KRIT(1) = 1                                                11
      GO TO 10000                                                12

C*********************************************************************
C
C     Step 2 :
C
C     LMAX        : Maximum number of sub-problems.
C     LS(*)       : Index of the *th sub-problem to be solved. Zero
C                   indicates master problem.
C     NC(*)       : Number of proposals to be generated by the *th
C                   sub-problem.
C     KRIT(*)     : Indicates stopping criterion when solving sub-problem
C                   LS(*). When one it indicates that one proposal is to be
C                   generated at optimality or unboundedness. Two means
```

```
C                    that NC(*) proposals are to be generated.
C     KK             : Maximum number of sub-problems to besolved in a
C                      cycle.
C     MAXCA          : Maximum number of proposals that can be generated
C                      in a cycle by any sub-problem.
C     KSTR           : Control parameter based on which policy is set.
C     LCOUNT         : Used when KSTR is 3. Its exact significance will be
C                      explained later.
C
C     Case 1 : (KSTR = 1) In this case all sub-problems are solved and one
C     proposal is to be generated at optimality or unboundedness. Therefore,
C     LS(i) is i, NC(i) is one, KRIT(i) is one. [lines 15-17].
C     Since all sub-problems are solved, a total of LMAX+1 problems are
C     solved (including master) and the (LMAX+1)st problem is the master
C     problem. [lines 9-10].
C     Case 2 : (KSTR = 2) In this case every sub-problem is solved and a
C     maximum of MAXCA proposals can be generated in a cycle by any
C     sub-problem. Therefore, LS(i) is i, NC(i) is set to MAXCA and
C     KRIT(i) is set to 2. [lines 25-27].
C     Again a total of LMAX+1 problems are solved (KK is set to LMAX+1)
C     and the last problem is the master problem. [lines 28-30].
C     Case 3 : (KSTR = 3) In this case, during the first cycle, every
C     sub-problem is solved and a maximum of MAXCA proposals can be
C     by any sub-problem. Subsequently, only one sub-problem per cycle is
C     solved in a cyclic manner. Consider an example with five sub-problems.
```

```
C     If the fifth sub-problem was solved in a certain cycle then in the
C     next cycle only the first sub-problem is solved. The variable LCOUNT is
C     kept to indicate the sub-problem/sub-problems to be solved in a cycle.
C     When LCOUNT is -1 then it indicates the first cycle and all sub-problems
C     are solved to generate a maximum of MAXCA proposals. Hence when
C     LCOUNT is -1 it is just like the case when KSTR is 2. [lines 32,22].
C     Otherwise, only the LCOUNT^th sub-problem is solved during a cycle.
C     [lines 36-40].
C     To maintain the cyclic order of solving sub-problems, LCOUNT is
C     updated to LCOUNT mod LMAX at the end of every cycle.
C     [lines 33-34].
C     Case 4 : (KSTR = 4 and 5). As far as POLICY is concerned this is the
C     same KSTR = 2. [lines 13].
C
C*************************************************************************
990   GO TO(1000,2000,3000,2000,2000),KSTR                          13
1000  DO 1200 I=1,LMAX                                              14
      LS(I) = I                                                     15
      KRIT(I) = 1                                                   16
1200  NC(I) = 1                                                     17
      LS(LMAX+1) = 0                                                18
      KRIT(LMAX+1) = 1                                              19
      KK = LMAX + 1                                                 20
      GO TO 10000                                                   21
1900  LCOUNT = LCOUNT + 1                                           22
```

```
2000  DO 2200 I=1,LMAX                                    23
      ICASUB(I) = 0                                       24
      LS(I) = I                                           25
      KRIT(I) = 2                                         26
2200  NC(I) = MAXCA                                       27
      LS(LMAX+1) = 0                                      28
      KRIT(LMAX+1) = 1                                    29
      KK = LMAX + 1                                       30
      GO TO 10000                                         31
3000  IF(LCOUNT .EQ. -1)GO TO 1900                        32
3100  LCOUNT = LCOUNT + 1                                 33
      IF(LCOUNT .GT. LMAX)LCOUNT = 1                      34
      IF(MSTATU(LCOUNT) .EQ. QBL)GO TO 3100               35
      LS(1) = LCOUNT                                      36
      KRIT(1) = 2                                         37
      NC(1) = MAXCA                                       38
      LS(2) = 0                                           39
      KRIT(2) = 1                                         40
      KK = 2                                              41
10000 RETURN                                              42
      END                                                 43
```

3.15. Subroutine PRICE :

3.15.1. Major Task :

Selects columns for multiple pricing.

3.15.2. Input :

1. The array A(*).
2. The dual prices in the buffer YA(*,KVEC).
3. KMULT indicating the maximum number of columns that can be selected for multiple pricing. This is a control parameter specified by the user.

3.15.3. Outputs :

1. Sets MULT to the number of columns selected for multiple pricing.
2. The column index of the columns selected for multiple pricing is stored in JCOLP(*).
3. The reduced costs of each of these columns is stored in DCMIN(*).

3.15.4. Called By :

NORMAL

3.15.5. Description :

PRICE examines each non-basic column as follows. First, the reduced cost of the column is computed and then based on this the column may or may not be selected for multiple pricing. The reduced cost of the column $\mathbf{a_j}$ (cf. § 2.2.3) is computed using $\pi\mathbf{a_j}$ where π is the dual vector.

A column can be chosen for multiple pricing provided its reduced cost is "better" than a threshold value. The threshold value is initially set to zero and is updated according to the following cases.

Case 1 : (Number of columns chosen < KMULT). Since enough columns have not yet been selected any column with reduced cost less than zero is good enough. Therefore, the threshold reduced cost remains at zero.

Case 2 : (Number of columns selected = KMULT). In this case the threshold reduced cost is updated to the value of the "worst" reduced cost of columns already chosen for multiple pricing. The column just chosen replaces the column with the "worst" reduced cost and thereby the number of columns selected for multiple pricing never exceeds the user specified limit KMULT.

Once a column is selected for multiple pricing, its column index in the matrix **A** is stored in the array JCOLP(*). The reduced costs of each of these columns is stored in the array DCMIN(*).

PRICE therefore, comprises the following steps.

3.15.6. Steps :

1. Initialize

2. For each column i do

(a) Compute the reduced cost.

(b) Decide if the column should be selected for multiple pricing.

End for.

3.15.7. Detailed Comments :

```
C***********************************************************************
C
C     Step 1 :
C
C     MULT       : Number of columns selected for multiple pricing.
C     DCOST      : Threshold value of the reduced cost.
C     ZTCOST     : Zero tolerance for reduced cost.
C     KVEC       : Column index of the buffer YA(*,*) that contains the
C                  dual vector π.  When KFASE is one the dual vector is
C                  in YA(*,5) and otherwise it is in YA(*,6).
C
C***********************************************************************
      IYIV = 0                                                    1
      MULT = 0                                                    2
      DCOST = -ZTCOST                                             3
      KVEC = 6                                                    4
      IF(KFASE .EQ. 1)KVEC = 5                                    5
```

```
C*****************************************************************
C
C     Step 2a :
C
C     KINBAS(*)     : Zero indicates that the *th column is non-basic.
C     YA(*,KVEC)    : Contains the dual vector.
C     A(*)          : Contains the columns for which the reduced cost is
C                     to be computed.
C     DSUM          : Indicates reduced cost of the column being considered.
C     LL            : Position in A(*) of the first non-zero of the column
C                     being considered.
C     KK            : Position in A(*) of the last non-zero of the column
C                     being considered.
C     ISTYPE(*)     : Indicates the type of the *th row. Minus one indicates
C                     equality row. One indicates inequality row.
C
C     1. Ignore basic columns. Also ignore equality logicals since they are not
C        allowed to enter the basis. [lines 7-9].
C     2. Compute the dot product πa_j which is the reduced cost. [lines 11-14].
C
C*****************************************************************
      DO 1000 J=1,NCOL                                          6
      IF (J .GT. NROW) GO TO 499                                7
      IF (ISTYPE(J) .NE. 1) GO TO 1000                          8
  499 IF (KINBAS(J) .NE. 0) GO TO 1000                          9
```

```
      DSUM = 0.                                                    10
      LL = LA(J)                                                   11
      KK = LA(J + 1) - 1                                           12
      DO 500 I=LL,KK                                               13
  500 DSUM = DSUM + YA(IA(I),KVEC)*A(I)                            14

C*********************************************************************
C
C     Step 2b :
C
C     DSUM        : Indicates the reduced cost of the column considered.
C     DCOST       : Indicates the thershold value of the reduced cost.
C                   Initialized to zero.
C     JCOLP(*)    : Index of the *th column selected for multiple pricing.
C     DCMIN(*)    : Reduced cost of the *th column selected for multiple
C                   pricing.
C     MULT        : Number of columns selected for multiple pricing.
C
C     1. If the reduced cost of the column just considered is greater than the
C        threshold value then it cannot be selected for multiple pricing. Hence
C        go to step 2a. Otherwise there are two cases.
C             Case 1 : (MULT < KMULT) Increment MULT to indicate the
C             number of columns selected for multiple pricing. Store the
C             reduced cost and the column index of the column. [lines 26-28].
C             If the column just selected causes the number of columns selected
```

```
C           for multiple pricing to equal KMULT then the threshold reduced
C           cost is updated.  [lines 29-31,19-24]
C           Case 2 : (MULT = KMULT)  Update the value of the threshold
C           reduced cost.  [lines 19-24].
C
C*****************************************************************
      IF(DSUM .GE. DCOST)GO TO 1000                              15
      IF(MULT .LT. KMULT)GO TO 700                               16
      DCMIN(JMIN) = DSUM                                         17
      JCOLP(JMIN) = J                                            18
  550 DCOST = DSUM                                               19
      DO 600 I = 1,KMULT                                         20
      IF(DCOST .GE. DCMIN(I))GO TO 600                           21
      DCOST = DCMIN(I)                                           22
      JMIN = I                                                   23
  600 CONTINUE                                                   24
      GO TO 1000                                                 25
  700 MULT = MULT + 1                                            26
      DCMIN(MULT) = DSUM                                         27
      JCOLP(MULT) = J                                            28
      IF(MULT .LT. KMULT)GO TO 1000                              29
      JMIN = MULT                                                30
      GO TO 550                                                  31
```

```
1000  CONTINUE                                                    32
      RETURN                                                      33
      END                                                         34
```

3.16. Subroutine RESULT :

3.16.1. Major Task :

Reconstructs the primal solution vector.

3.16.2. Calling :

FORMC, UNRAVL, BTRAN, INVERT, NORMAL, UNPACK.

3.16.3. Called By :

Main Program.

3.16.4. Description :

RESULT first checks if the master problem or one of the sub-problems is infeasible. In this case phase 3 is abandoned. Otherwise, the feasible vector to be reconstructed (cf § 2.5.3) is :

$$(x_0^*, x_1^*, \ldots, x_R^*) \qquad (3.35)$$

First, the vector x_0^* is obtained from the second component of the solution to the master problem (cf. § 2.5.3). Subsequently, variable names are read from DAD

(direct access device) and the subroutine UNRAVL outputs the vector $\mathbf{x_0}^*$.

Next, each $\mathbf{x_r}^*$ (r = 1,..,R) is reconstructed by solving (cf § 2.5.3) the sub-problem :

$$
\begin{aligned}
&\text{Min} \quad \mathbf{c_r x_r} \\
&\text{ST} \\
&\qquad \mathbf{A1_r x_r} = \mathbf{y_r} \\
&\qquad \mathbf{A2_r x_r} = \mathbf{b_r} \qquad (3.36) \\
&\qquad \mathbf{x_r} \geq \mathbf{0}
\end{aligned}
$$

where $\mathbf{A1_r}$ is m_0 by n_r and $\mathbf{A2_r}$ is m_r by n_r and all other vectors and matrices are of suitable dimension. This is done in two steps. The first step is to compute $\mathbf{y_r}$ and the second step is to actually solve the sub-problem (3.36) defined above.

1. Compute $\mathbf{y_r}$: The vector $\mathbf{y_r}$, termed as allocation vector, contains the allocation of each resource to sub-problem r. Consider an example with three coupling rows and two sub-problems. Assume that the problem was optimal after two cycles and that each sub-problem generated one proposal per cycle. Let q^k_{ij} denote the j^{th} element of the kth proposal generated by sub-problem i. Let v_i and u_i indicate the weight on the i^{th} proposal from sub-problems 1 and 2 respectively. Then, the constraints of the master problem at the end of two cycles are given in (3.37) below :

$$
\begin{array}{lllllll}
s_1 & \mathbf{a1_{o1}x_o} & + q_{11}{}^1v_1 + q_{11}{}^2v_2 + q_{21}{}^1w_1 + q_{21}{}^2w_2 = d_{o1} & & & & \\
s_2 & \mathbf{a1_{o2}x_o} & + q_{12}{}^1v_1 + q_{12}{}^2v_2 + q_{22}{}^1w_1 + q_{22}{}^2w_2 = d_{o2} & & & & \\
s_3 & \mathbf{a1_{o3}x_o} & + q_{13}{}^1v_1 + q_{13}{}^2v_2 + q_{23}{}^1w_1 + q_{23}{}^2w_2 = d_{o3} & & & & (3.37) \\
s_4 & & + v_1 & + v_2 & & = 1 & \\
s_5 & & & & + w_1 + w_2 & = 1 &
\end{array}
$$

where s_i denotes the logical for the i^{th} constraint, $\mathbf{b_{oi}}$ denotes the right hand side (resource availability) of the i^{th} coupling constraint and $\mathbf{a1_{oi}}$ denotes the i^{th} row of the coupling columns $\mathbf{A1_o}$, (cf § 2.5.3) of the problem. The allocation vector $\mathbf{y_r}$ (r = 1,2) is

$$\mathbf{y_1} = (q^1{}_{11}v_1, q^1{}_{12}v_1, q^1{}_{13}v_1)^t + (q^2{}_{11}v_2, q^2{}_{12}v_2, q^2{}_{13}v_2)^t \qquad (3.38)$$

for sub-problem 1 and

$$\mathbf{y_2} = (q_{21}{}^1w_1, q_{22}{}^1w_1, q_{23}{}^1w_1)^t + (q_{21}{}^2w_2, q_{22}{}^2w_2, q_{23}{}^2w_2)^t \qquad (3.39)$$

for sub-problem 2. For each sub-problem r, $\mathbf{y_r}$ is computed and written to the unit 8 work file. To compute $\mathbf{y_r}$, a scan is performed through all rows (including convexity) of the master problem as follows. For each row, the column that is basic in that row is determined. If the column was generated by the r^{th} sub-problem then

the allocation vector y_r is updated by adding the contribution from the column. Otherwise, the next row is taken up.

After computing y_r, an array named XRHS (i) is initialized to :

$$b_{oi} - A1_{oi}x_o \quad \text{for all } i = 1,...,m_o \tag{3.40}$$

where $a1_{oi}$ is the i^{th} row of $A1_o$. Therefore, at this stage, XRHS denotes the amount of resource left after subtracting the resource consumed by the master problem.

2. *Solve sub-problems* : At the end of D-W phase 2, let π_r^* be the dual vector corresponding to constraints $A2_r x_r = b_r$. Without loss of generality, the reconstruction of x_r^* can be stated as follows :

2.1. The sub-problem is brought to core by calling CHANGE and the allocation vector y_r is read from the unit 8 work file. INVERT is then called to begin the solution of the sub-problems.

2.2. The dual vector π_r^* which is also to be output is not readily available at this stage. Also, the dual vector $\pi'_r{}^*$ for the constraints $A2_r x_r = b_r$ obtained by solving (3.36) would not in general be the same as π_r^*. Therefore, FORMC and BTRAN are called to recompute π_r^* before going to (2.3).

2.3. The type of the rows, corresponding to the constraints $A1_r x_r = y_r$, is changed from "non-binding" to "binding equality". NORMAL is called to solve the problem (3.36).

2.4. The array XRHS is updated so that after treating sub-problem r, it contains :

$$b_{oi} \quad — \quad \sum_{j=0}^{r} Al_{ji}\, x_j \qquad \text{for all } i = 1,\ldots,m_o \qquad (3.41)$$

In other words, after solving the r^{th} sub-problem the amount of resource consumed by that sub-problem is subtracted from what was left after treating sub-problems 1 through (r-1).

2.5. UNRAVL is called to output the solution.

RESULT therefore consists of the following steps.

3.16.5. Steps :

1. Check if phase 3 is to be abandoned and if so GO TO (4). Else proceed.
2. Compute allocation.
3. Solve sub-problems.
4. STOP.

3.16.6. Detailed Comments :

```
C*********************************************************************
C
C     Step 1 :
C
C     MSTAT          : Status of the problem. (cf. variable dictionary)
```

```
C     LCORE          : Flag to indicate if the LCORE^th problem is in core.
C     LSUB           : Index of the problem being solved. Zero indicates that the
C                      master problem is being solved.
C     LMAXP1         : Number of sub-problems plus one.
C     NOLA(4,LSUB): Pointer toward the direct access device (DAD) where the
C                      names (row and column) of the LSUB^th problem are
C                      stored.
C
C     - Phase 3 is to be abandoned if :
C           - One of the sub-problems is infeasible or [line 1,6-13].
C           - The master problem is either unbounded or infeasible.
C             [lines 2,4,14-16].
C       Else proceed [line 3].
C           - Solve the master problem if the solution is not already in core.
C             [lines 17-19].
C           - Read master problem's names.  [lines 20-21].
C           - Call UNRAVL to output x_0*.  [line 22].
C
C*********************************************************************
      IF(MSTAT .EQ. QN .AND. LCORE .GE. 1)GO TO 1000                  1
      IF(MSTAT .EQ. QU)GO TO 1500                                     2
      IF(MSTAT .EQ. QBL)GO TO 2000                                    3
      IF(MSTAT .EQ. QN)GO TO 1000                                     4
      STOP 1234                                                       5
 1000 WRITE(6,1200)LSUB                                               6
```

```
1200  FORMAT(1H1,'SUBPROBLEM ',I4,'  HAS NO SOLUTION')        7
      MM = LSUB                                               8
      IF(LSUB .EQ. 0)MM = LMAXP1                              9
      KD = NOLA(4,MM)                                         10
      READ(9'KD)NAME(1),NAME(2),ICNAM                         11
      CALL UNRAVL                                             12
      GO TO 10000                                             13
1500  WRITE(6,1600)                                           14
1600  FORMAT(1H1,' MASTER PROBLEM IS UNBOUNDED')              15
      GO TO 10000                                             16
2000  IF(LCORE .EQ. 0)GO TO 2050                              17
      LSUB = 0                                                18
      CALL NORMAL                                             19
2050  KD = NOLA(4,LMAXP1)                                     20
      READ(9'KD)NAME(1),NAME(2),ICNAM                         21
      CALL UNRAVL                                             22


C*********************************************************************
C
C     Step 2 :
C
C     LMAX        : Number of sub-problems.
C     MAST        : Indicates the phase of the D-W algorithm that is being
C                   executed.  If 2 then it indicates that a phase 3 operation
```

```
C                    (compute y_r or solve sub-problems to reconstruct x_r*) is
C                    being executed.
C     NROW           : Number of rows in the problem being solved.
C     NROWO          : Number of coupling rows.
C     NCOLO          : Number of coupling columns. That is, the number of
C                      columns in A1_0 (cf. § 2.5.3).
C     JH(*)          : Indicates which column is basic in the *th row.
C     ICOLS(*)       : Indicates the sub-problem origin of the *th proposal.
C     Z(*,*)         : Normally the proposal buffer. In RESULT it is used as a
C                      buffer to compute the coupling rhs of each sub-problem.
C     A(*)           : Data array.
C     X(*)           : Solution array.
C     YA(*,6)        : Contains dual variable of the D-W phase 2 coupling rows.
C
C     (a) Set MAST to 2.Indicates that a phase 3 operation is being done.
C         [line 24].
C     (b) Initialize the Z(*,*) buffer. [lines 26-27].
C     (c) Consider a row and obtain the column that is basic in this row. [line 29].
C     (d) Obtain the sub-problem origin of a master problem's column. [line 30].
C     (e) If the column was generated by the sub-problem currently treated then
C             - Compute the allocation due to this column for all resources.
C               [lines 31-34].
C         Else goto (c) to take up the next row unless all rows have beenconsidered.
C     (f) Write the allocation on the unit 8 work file. [line 36].
C     (g) Call UNPACK to obtain the rhs of the master in YA(*,1). [line 39].
```

```
C     (h) Copy the dual variables of the coupling into XYZ(*) and the master's rhs
C         into XRHS(*).  [lines 40-43].
C     (i) Subtract the amount of resource consumed by the master's coupling
C         columns A1_0.  This is done according to (3.6).  [lines 46-56].
C
C*********************************************************************
      IF(LMAX .EQ. 0)GO TO 10000                                    23
      MAST = 2                                                      24
      DO 2500 K=1,LMAX                                              25
      DO 2100 I=1,NROW                                              26
 2100 Z(I,1) = 0.                                                   27
      DO 2400 L=1,NROW                                              28
      IV = JH(L)                                                    29
      IF(ICOLS(IV) .NE. K)GO TO 2400                                30
      MM = LA(IV+1) - 1                                             31
      NNN = LA(IV)                                                  32
      DO 2300 I=NNN,MM                                              33
 2300 Z(IA(I),1) = Z(IA(I),1) + A(I)*X(L)                           34
 2400 CONTINUE                                                      35
      WRITE(8)(Z(N,1),N=1,NROWO)                                    36
 2500 CONTINUE                                                      37
      REWIND 8                                                      38
      CALL UNPACK(RHSCOL,1)                                         39
      DO 2520 I=1,NROW                                              40
      IF(I .LE. NROWO)ITEMP(I) = - 1                                41
```

```
      XYZ(I) = YA(I,6)                                          42
2520  XRHS(I) = YA(I,1)                                         43
      ITEMP(IOBJ) = 0                                           44
      DO 2550 I=1,NROW                                          45
      IV = JH(I)                                                46
      IF(IV .GT. NROWO)GO TO 2530                               47
      IF(X(I) .LE. ZTCAND)GO TO 2530                            48
      ITEMP(IV) = 0                                             49
2530  IF(IV .GT. NCOLO)GO TO 2550                               50
      LL = LA(IV)                                               51
      KK = LA(IV+1) - 1                                         52
      DO 2540 J=LL,KK                                           53
      IR = IA(J)                                                54
2540  XRHS(IR) = XRHS(IR) - A(J)*X(I)                           55
2550  CONTINUE                                                  56
      DO 2560 I=1,NROWO                                         57
2560  XYZ(I) = YA(I,6)                                          58

C*************************************************************************
C
C     Step 3 :
C
C     LSUB        : Index of the sub-problem being solved.
C     LMAX        : Number of sub-problems.
C     NROW        : Number of rows in the sub-problem being solved.
```

```
C     NROWO       : Number of coupling rows.
C     YA(*,6)     : Contains dual vector. The first NROWO elements of
C                   YA(*,6) contain the dual vector of either the phase 2
C                   coupling rows or the phase 3 coupling rows.
C     XYZ(*)      : Contains the dual vector of the D-W phase 2 coupling row
C                   in the first NROWO elements. When sub-problem i is
C                   being considered, its subsequent elements contain the D-W
C                   phase 2 dual vector, of the non-coupling rows of
C                   sub-problem i.
C     ISTYPE(*)   : Indicates the type of the *th constraint. One indicates
C                   equality logicals.
C     MAST        : Indicates the stage of the algorithm being executed.
C     XRHS(*)     : The amount of each resource left after the treatment of the
C                   sub-problem currentky solved.
C
C     Reference will be made to "solve sub-problems" part of "description". Note
C     the reconstruction of just one x_r* is described below in steps a-e.
C
C     (a) Read the problem and call INVERT to update the solution. [lines 62-66].
C     (b) Now π_r* (cf. step 2.2) is to be reconstructed. After INVERT,
C       π_0^k, which was residing in the first NROWO elements of YA(*,6) has
C       been destroyed and this required to recompute π_r*. These are copied from
C       XYZ(*). [lines 68-69].
C       Call FORMC and BTRAN to compute π_r*. [lines 70-71].
```

```
C        Save the dual vector [π0*,πr*] in XYZ(*). [lines 72-73].
C        Note that since this is not a phase 3 step, set MAST to 1. [line 67].
C     (c) Now 2.3 is to be executed. Hence, reset MAST back to 2. [line 74].
C        Set type of the constraints A1rxr=yr as equality logicals.[lines 41,75-76].
C        Call NORMAL to reconstruct xr*. [line 78].
C     (d) Update XRHS(*) as per (3.6). [lines 81-90].
C     (e) Call UNRAVL to output xr*. [line 91].
C     (f) Compute the infinity norm of XRHS(*) and print it. [lines 98-103].
C
C*************************************************************************
*

      NTEMP(9) = NT                                                    59
      NTEMP(10) = NX                                                   60
      DO 3000 I=1,LMAX                                                 61
      LSUB = I                                                         62
      LSUP = I                                                         63
      CALL CHANGE(LSUP)                                                64
      READ(8)(Z(N,1),N=1,NROWO)                                        65
      CALL INVERT                                                      66
      MAST = 1                                                         67
      DO 2570 JK=1,NROWO                                               68
 2570 YA(JK,6) = XYZ(JK)                                               69
      CALL FORMC                                                       70
```

```
      CALL BTRAN                                          71
      DO 2580 JK=1,NROW                                   72
2580  XYZ(JK) = YA(JK,6)                                  73
      MAST = 2                                            74
      DO 2600 J=1,NROWO                                   75
      ISTYPE(J) = ITEMP(J)                                76
2600  CONTINUE                                            77
      CALL NORMAL                                         78
      KD = NOLA(4,LSUB)                                   79
      READ(9'KD)NAME(1),NAME(2),ICNAM                     80
      DO 2800 JK=1,NROW                                   81
      IV = JH(JK)                                         82
      LL = LA(IV)                                         83
      KK = LA(IV+1) - 1                                   84
      DO 2700 JRHS=LL,KK                                  85
      IR = IA(JRHS)                                       86
      IF(IR .GT. NROWO)GO TO 2700                         87
      XRHS(IR) = XRHS(IR) - A(JRHS)*X(JK)                 88
2700  CONTINUE                                            89
2800  YA(JK,6) = XYZ(JK)                                  90
      CALL UNRAVL                                         91
      NTEMP(9) = NTEMP(9) + NT                            92
      NTEMP(10) = NTEMP(10) + NX                          93
3000  CONTINUE                                            94
      LSUP = 0                                            95
```

```
      CALL CHANGE(LSUP)                                                  96
      ERMAX = 0.0                                                        97
      DO 4000 I=1,NROWO                                                  98
      IF(ISTYPE(I) .EQ. 0)GO TO 4000                                     99
      IF(DABS(XRHS(I)) .GT. ERMAX)ERMAX = DABS(XRHS(I))                  100
4000  CONTINUE                                                           101
      WRITE(6,4500)ERMAX                                                 102
4500  FORMAT(' MAXIMUM ROW ERROR                                         103
     1IN COMMON ROWS  = ',E14.5)
      TIMER1 = NTEMP(9)/1000.                                            104
      TIMER2 = NTEMP(10)/1000.                                           105
      WRITE(6,5000)TIMER1                                                106
5000  FORMAT(' TOTAL TRANSFORMATION                                      107
     1TIME  = ',F8.2,'  SECONDS')
      WRITE(6,6000)TIMER2                                                108
6000  FORMAT(' TOTAL INPUT/OUTPUT TIME  = ',F8.2,'  SECONDS')            109
10000 RETURN                                                             110
      END                                                                111
```

3.17. Subroutine UNPACK :

3.17.1. Major Task :

Unpacks a column stored in standard column-wise packing.

3.17.2. Inputs :

1. KVEC, giving the column position in the buffer YA(*,*) into which the data column is to be unpacked. Note that throughout *this sub-routine* the word data column will be used to indicate the column in the array A(*) that is to be unpacked.
2. IV, giving the index of the data column that is to be unpacked.

3.17.3. Called By :

NORMAL, INVERT, UNRAVL

3.17.4. Comments :

```
C*****************************************************************
C
C     YA(*,*)        : Buffer into which the data column is to be unpacked
C     KVEC           : Unpacks the data column into the KVEC^th column of
C                      YA(*,*).
```

```
C     A(*)          : Data array.
C     IA(*)         : Indicates the row index of the *th element of A(*).
C     LA(*)         : Indicates the position in A(*) where the first non-zero
C                     element of the *th data column resides.
C
C     1. Store the zero vector in YA(*,KVEC). [lines 1-3]
C     2. Set LL to the starting position in A(*) of the data column  and set KK
C        to the ending position in A(*) of the data column. [lines 4-5].
C     3. If there are no elements in the data column then RETURN.  Else
C        proceed. [line 6]
C     4. For each non-zero element i in the data column do :
C              - Set IR to IA(i). Thereby, IR contains the row index to which
C                the i th element belongs. Assign A(i) to YA(IR,KVEC).
C                [lines 8-9].
C        End For
C
C*********************************************************************
      DO 100 I = 1,NROW                                          1
      YA(I,KVEC) = 0.                                            2
  100 CONTINUE                                                   3
      LL = LA(IV)                                                4
      KK = LA(IV+1) - 1                                          5
      IF(KK .LT. LL)GO TO 500                                    6
      DO 200 I = LL,KK                                           7
      IR = IA(I)                                                 8
```

```
      YA(IR,KVEC) = A(I)                                        9
200   CONTINUE                                                  10
500   RETURN                                                    11
      END                                                       12
```

3.18. Subroutine UNRAVL :

3.18.1. Major Task :

Outputs the primal solution vector $\mathbf{x}_r$ (r = 0,...,R) corresponding to the r^{th} problem.

3.18.2. Inputs :

1. The index of the sub-problem
2. The solution vector $\mathbf{x}_r$ in the array X(*).
3. The dual vector in the buffer YA(*,6).

3.18.3. Outputs :

1. The row names.
2. The column names or in case the column is a proposal the sub-problem origin of the proposal.
3. The dual vector, the rhs vector and the primal solution vector.

3.18.4. Called By :

RESULT.

3.18.5. Calling :

UNPACK to unpack the rhs vector.

3.18.6. Description :

The outputs given in "outputs" are obtained as explained below. The solution vector and the dual vector are provided by subroutine RESULT. The rhs vector is obtained by calling UNPACK and in case of the master problem, ones are stored in positions corresponding to convexity constraints. Note that the type (integer, character and so on) of each of the above is always the same and therefore the format statement needed to output these can be easily written.

On the otherhand, the format statement for the names (row and column) is more difficult to infer and this is explained below. Each row and column name consists of two parts. The *type* of the first part never changes and is always in character form while the *type* of the second part changes. In general any name that is specified by the user is read in character form while for names that are assigned by the program the second part is of type integer. Therefore, in the case of sub-problems all names are in characters form (cf. Subroutine INPUT) but fot the master some of the names are assigned by the program and in these cases the second part is an integer. Thus, just one format statement would be sufficient for outputting the sub-problem's "results" whereas the format statement for the master is different in different cases. To explain these cases let NCOLO denote the number of coupling columns, NROWO denote the number of coupling rows and NROW denote the number of rows (including convexity) in the master problem. Also, let IRNAM1 and IRNAM2 denote the two parts of a row's name and ICNAM1 and

ICNAM2 denote the two parts of a column's name. The master problem is partitioned into six regions as shown in the figure (3.17). The following remarks about the names of the master then apply.

1. Since the name of the i^{th} convexity constriant is QQi (cf. Subroutine INPUT), IRNAM2 is an integer for any row whose row index is exceeds NROWO (implies convexity row). Therefore, to specify the row index of a variable in regions 2 or 4 or 6, IRNAM2 is an integer.
2. Since the row names of coupling constraints (row index $i \leq$ NROWO) are specified by the user IRNAM2 is of type character. Therefore, to specify the row index of a variable in regions 1 or 3 or 5, IRNAM2 would be in character form.
3. Due to the input mechanism the names of the i^{th} logical columns (column index $i \leq$ NROW) is the same as the i^{th} row. Using this statement and remarks (1) and (2) it can be concluded that ICNAM2 would be : (a) an integer to specify the column index of a variable in regions 3 or 4 and (b) in character form to specify the column index of a variable in regions 1 or 2.
4. A column in regions 5 or 6 is a proposal column. Hence, the column index of a variable in regions 5 or 6 is identified by giving the index of the sub-problem that generated the proposal.

These remarks are summarised in table (3.1).

3.18.7. Comments :

```
C**********************************************************************
C
C     NROW          : Number of rows in the problem being considered.
```

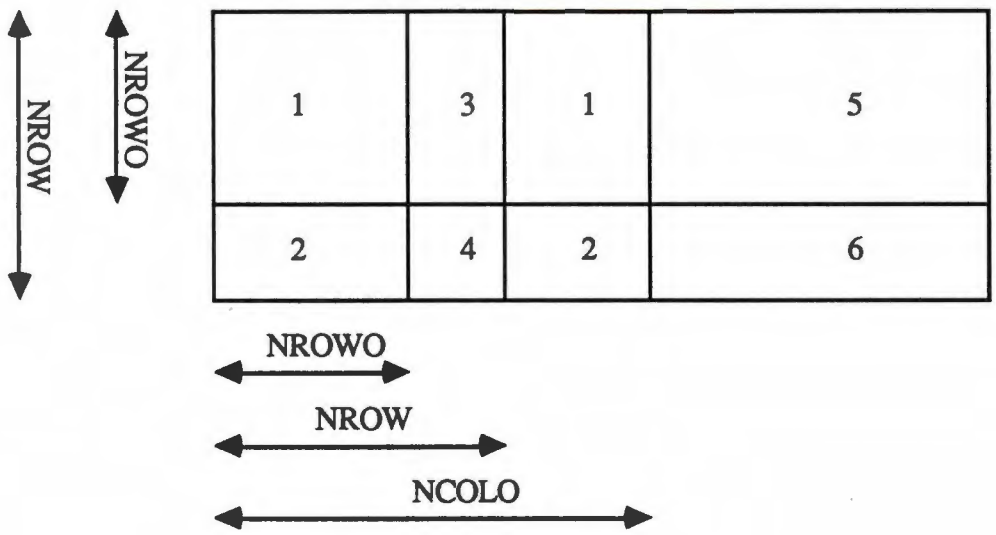


Figure 3.17 : Master problem name's data type

Table 3.1 : Tabular representation of master problem name's data type

Variable Region	Column Name's Type		Row Name's Type	
	Part 1	Part 2	Part 1	Part 2
1	Character	Character	Character	Character
2	Character	Character	Character	Integer
3	Character	Integer	Character	Character
4	Character	Integer	Character	Integer
5	Integer	Integer	Character	Character
6	Integer	Integer	Character	Integer

```
C     NROWO      : Number of coupling rows.
C     NCOLO      : Number of coupling columns.
C     YA(*,6)    : Contains the dual vector.
C     YA(*,1)    : Contains the rhs vector.
C     X(*)       : Contains the solution vector that is to be output.
C     JH(*)      : Indicates the column that is basic in the *th row.
C     IV         : Column index of the column that is basic in the row
C                  being considered.
C     LMAX       : Number of sub-problems.
C     LSUB       : Index of the provblem being considered. Zero indicates
C                  master problem.
C     RHSCOL     : The column that is used as the rhs of the problem.
C
C     1. Call UNPACK to unpack the rhs. If LSUB indicates the master then
C        store ones in positions corresponding to convexity constriants. If
C        LSUB indicates sub-problem then copy the allocation rhs y_r.
C        [lines 6,9-10,13-14]
C     2. The output is then done as follows. Lines 23 to 25 output the results
C         sub-problems and for variables in region 1 of the master. Lines 35 and
C        36 output results for region 2 of the master. Lines 28 and 29 output
C        results for variables in region 3 of the master. Lines 32 and 33 output
C        results for variables in region 4 of the master. Lines 39 and 40 output
C        results for variables in region 5 of the master. And finally lines 42 and
C        43 output results for variables in region 6 of the master.
C*****************************************************************
```

```
      EQUIVALENCE (ICNAM(1),A(5001))                                    1
      WRITE(6,8000)(NAME(I),I = 1,2)                                    2
 8000 FORMAT(1H1,'SUBPROBLEM ',2A4//)                                   3
      WRITE(6,8100)                                                     4
 8100 FORMAT(6H JH(I),16X,5HVALUE,9X,9HROW                              5
     1NAMES,9X,5HPI(I),12X,3HRHS )
      CALL UNPACK(RHSCOL,1)                                             6
      IF(LMAX .EQ. 0 .OR. MAST .EQ. 0)GO TO 500                         7
      IF(LSUB .GT. 0)GO TO 200                                          8
      DO 100 I=1,LMAX                                                   9
  100 YA((NROWO+I),1) = 1.                                              10
      GO TO 500                                                         11
  200 IF(MAST .NE. 2)GO TO 500                                          12
      DO 300 I=1,NROWO                                                  13
  300 YA(I,1) = Z(I,1)                                                  14
  500 DO 1000 I = 1,NROW                                                15
      IV = JH(I)                                                        16
      IRNAM1 = ICNAM(I,1)                                               17
      IRNAM2 = ICNAM(I,2)                                               18
      IF(LSUB .EQ. 0 .AND. IV .GT. NCOLO)GO TO 800                      19
      ICNAM1 = ICNAM(IV,1)                                              20
      ICNAM2 = ICNAM(IV,2)                                              21
      IF(LSUB .EQ. 0 .AND. MAST .NE. 0)GO TO 600                        22
  550 WRITE(6,8200)ICNAM1,ICNAM2,X(I),IRNAM1,IRNAM2,                    23
```

```
     1YA(I,6),YA(I,1)
8200 FORMAT(1H ,A4,A4,8X,G16.8,4X,A4,A4,4X,2G16.8 )                    24
     GO TO 1000                                                        25
 600 IF(I .GT. NROWO)GO TO 700                                         26
     IF(IV .LE.NROWO .OR. IV .GT. NROW)GO TO 550                       27
     WRITE(6,8300)ICNAM1,ICNAM2,X(I),IRNAM1,IRNAM2,                    28
     1YA(I,6),YA(I,1)
8300 FORMAT(1H ,A4,I4,8X,G16.8,4X,A4,A4,4X,2G16.8 )                    29
     GO TO 1000                                                        30
 700 IF(IV .LE. NROWO .OR. IV .GT. NROW)GO TO 750                      31
     WRITE(6,8400)ICNAM1,ICNAM2,X(I),IRNAM1,IRNAM2,                    32
     1YA(I,6),YA(I,1)
8400 FORMAT(1H ,A4,I4,8X,G16.8,4X,A4,I4,4X,2G16.8 )                    33
     GO TO 1000                                                        34
 750 WRITE(6,8500)ICNAM1,ICNAM2,X(I),IRNAM1,IRNAM2,                    35
      1YA(I,6),YA(I,1)
8500 FORMAT(1H ,A4,A4,8X,G16.8,4X,A4,I4,4X,2G16.8 )                    36
     GO TO 1000                                                        37
 800 IF(I .GT. NROWO)GO TO 850                                         38
     WRITE(6,8600)IV,ICOLS(IV),X(I),IRNAM1,IRNAM2,                     39
     1YA(I,6),YA(I,1)
8600 FORMAT(1H ,2I4,8X,G16.8,4X,A4,A4,4X,2G16.8 )                      40
     GO TO 1000                                                        41
 850 WRITE(6,8700) IV,ICOLS(IV),X(I),IRNAM1,IRNAM2,                    42
     1YA(I,6),YA(I,1)
```

```
8700  FORMAT(1H ,2I4,8X,G16.8,4X,A4,I4,4X,2G16.8 )                    43
1000  CONTINUE                                                        44
      TIMER = NT/1000.                                                45
      WRITE(6,8800)TIMER                                              46
8800  FORMAT (4H EOF,/,'0TRANSF TIME =' ,F8.2,8H SECONDS)             47
      TIMER = NX/1000.                                                48
      WRITE(6,9000)TIMER                                              49
9000  FORMAT    (' INPUT/OUTPUT TIME  = ',F8.2,'  SECONDS'/'1')       50
      RETURN                                                          51
      END                                                             52
```

3.19. Subroutine UPBETA :

3.19.1. Major Task :

Updates the solution.

3.19.2. Input :

Column index of the entering column.

3.19.3. Called By :

NORMAL

3.19.4. Comments :

```
C*********************************************************************
C
C     YA(*,*)      : Buffer containing the entering column..
C     KVEC         : The column index of the buffer YA(*,*) that contains
C                    the entering column.
C     IVOUT        : Index of the leaving column.
C     KINBAS(*)    : If the *th column is basic it indicates the row in which it
C                    is basic.
C     X(*)         : Contains the solution vector.
```

```
C     DP          : Gives the allowable increase in the entering variable.
C     IROWP       : Indicates the pivot row.
C     DE          : Value of the IROWP^th element of the current rhs.
C     DCMIN(*)    : Reduced cost of the *^th column of the buffer YA(*,*)
C     DOB         : Chenge in the objective value due to the column just
C                   wanting to enter.
C
C     1. Compute DP. [lines 1-3].
C     2. The value of the new basic variable for the pivot row IROWP should
C        be allowable increase in the entering variable. Hence set X(IROWP)
C        to DP. [line 4]
C     3. For each non-pivot row do
C              - Set X(i) <-- X(i) - YA(i,KVEC)*X(IROWP). If |X(i)| is less
C                than the zero tolerance then set it to zero. [lines 7-9]
C        End For
C     4. Compute the increase in the objective due to this pivot change provided
C        the problem currently treated is the sub-problem. Racall that this
C        would determine the reduced cost of the proposal provided the current
C        solution is sent as a proposal. [lines 11-13].
C
C*********************************************************************
      IROWP = KINBAS(IVOUT)                                          1
      DE = X(IROWP)                                                  2
      DP = DE/YA(IROWP,KVEC)                                         3
      X(IROWP) = DP                                                  4
```

```
      DO 1000 I = 1,NROW                                        5
      IF (I .EQ. IROWP) GO TO 1000                              6
      DE = X(I)                                                 7
      X(I) = DE - YA(I,KVEC)*DP                                 8
      IF (DABS(X(I)) .LE. 1.0D-15) X(I) = 0.0D0                 9
1000  CONTINUE                                                  10
      IF(KFASE .EQ. 1 .OR. LSUB .EQ. 0)GO TO 2000               11
      IF (MAST .EQ. 0)GO TO 2000                                12
      DOB = DOB + DCMIN(KVEC)*DP                                13
2000  RETURN                                                    14
      END                                                       15
```

3.20. Subroutine VECTOR :

3.20.1. Major Task :

Reads in a specified problem into core from DAD or writes a problem in core to DAD.

3.20.2. Input :

The parameter K1. K1 indicates a write if its value is one and it indicates a read if its value is two.

3.20.4. Called By :

CHANGE,INDATA,NORMAL

3.20.5. Comments :

```
C*****************************************************************
C
C     K1          : Parameter. One indicates write and two indicates read.
C     AX(*)       : Equivalent to A(*).
C     IAX(*)      : Equivalent to IA(*).
C     NBX(*)      : Equivalent ro NB(*).
C*****************************************************************
```

```
      GO TO (100,200),K1                    1
100   WRITE (9'KD) AX,IAX,NBX               2
      GO TO 1000                            3
200   READ (9'KD) AX,IAX,NBX                4
1000  RETURN                                5
      END                                   6
```

3.21. Subroutine WRETA :

3.21.1. Major Task :

Writes etas to the data matrix.

3.21.2. Input :

KVEC, giving the column position of YA(*,*) containing the eta vector.

3.21.3. Called By :

NORMAL

3.21.4. Description :

WRETA writes in the array A(*) the eta vector generated due to the basis change. Suppose the p^{th} column of the matrix **A** (= $[a_{1p},a_{2p},....,a_{pp},....,a_{np}]^t$) is the entering column. Then the eta vector generated due to this column is :

$$[-a_{1p}/a_{pp},....,1/a_{pp},....,-a_{np}/a_{pp}]^t \qquad (3.42)$$

The pivot element of the eta vector $1/a_{pp}$ is stored first. This avoids having to explicitly store the pivot row of each eta vector since the row index of the first

element of the eta vector specifies the required pivot row. Then, each of the remaining non-zero element of $\mathbf{a_p}$ (p^{th} column of **A**) is stored.

3.21.5. Comments :

```
C*************************************************************************
C
C     A(*)          : Array containing the data and eta elements.
C     IA(*)         : Indicates the row index in the matrix A of the *th
C                     element of the array A(*).
C     NELEM         : Number of non-zero elements in the array A(*).
C     YA(*,KVEC)    : Contains the pivot column or entering column a_p.
C     IROWP         : Pivot row of the column.
C     NETA          : Numbre of etas.
C
C     1. Incorporate the eta element 1/a_pp into the array A(*). [lines 2-4]
C     2. Incorporate each non-zero element a_ip (for all i ≠ p) into the array
C        A(*). [lines 6-10]
C     3. Increment NETA, the number of etas. [line 12]
C     4. Mark the end position of this eta. [line 13]
C
C*************************************************************************
      DATA ZTETA/1.D-15/                                              1
      NELEM = NELEM + 1                                               2
```

```
      IA(NELEM) = IROWP                                          3
      A(NELEM) = 1./YA(IROWP,KVEC)                               4
      DO 1000 I = 1,NROW                                         5
      IF (I .EQ. IROWP) GO TO 1000                               6
      IF (DABS(YA(I,KVEC)) .LE. ZTETA) GO TO 1000                7
      NELEM = NELEM + 1                                          8
      IA(NELEM) = I                                              9
      A(NELEM) = YA(I,KVEC)                                      10
1000  CONTINUE                                                   11
      NETA = NETA + 1                                            12
      LE(NETA+1) = NELEM + 1                                     13
      RETURN                                                     14
      END                                                        15
```

CHAPTER 4

Portability Issues

In this chapter issues concerning portability of the code are addressed. Since the code is written in FORTRAN IV it is easily portable. However, there are some parts of the code that are machine dependent and therefore they would require modification when installed on different machines. The following paragraphs identify these parts.

The usefulness of decomposition is felt mainly for problems whose data is very large. This implies that all the data cannot be accomodated in core at the same time. Hence, direct access I/O becomes a necessary part of any robust implementation. Since direct access I/O is machine dependent it is likely to create problems.

The statement defining the direct access device (DAD) normally contains a field for specifying the record length. On the IBM/370 this field is specified in terms of bytes. But on the VAX-VMS the record length should be in terms of four byte words and on a UNIX based machine the same should be in terms of bits. Also, on the IBM programming direct access I/O is easy since any message can be written (or read) just by specifying the starting record number in the statement : WRITE(Unit number'Starting record number) "Message." But on the VMS and on the UNIX based systems this would cause an error whenever the

size of the message exceeds the record length.

In DECOMP these machine-dependent differences require major changes in the sub-routine VECTOR. The pseudo-code to perform some of the functions of VECTOR on a VAX-VMS like system is as follows.

```
KD <-- Starting record number.
RECL <-- Record length in bytes.
DATSIZ <-- Size of the message in bytes.
LAST <-- Last component of A(*) that is to be written
A(*) <-- Array containing the message.
C
C It is assumed for simplicity that each component of A(*) is a byte long.
C KD gets incremented after the execution of a write.
C
      J <-- 0
      DO WHILE (DATSIZ > RECL)
            WRITE(Unit Number'KD) A(J+1:J+RECL)
            DATSIZ <-- DATSIZ - RECL
            J <-- J + RECL
      END
      WRITE (Unit Number'KD) A(J+1:LAST)
```

It is to be noted that the program to perform the function of VECTOR on a VAX-VMS like system is more complicated than what can be inferred from the psuedo-code. The psuedo-code given above would be correct only if all the elements to be written to DAD are contiguous components of the array A(*). This is not true for the master problem since data elements are followed by a gap to

accomodate the expansion of the master due to column generation. See figure (4.1). Therefore, the pseudo-code to perform the function of VECTOR in the case of the master problem has to be modified as follows.

```
        KD <-- Starting record number.
        RECL <-- Record length in bytes.
        DATSIZ <-- Size of the message (not including etas).
        ETASIZ <-- Size of the eta file.
        A(*) <-- Array containing the message to be written.
        LN <-- Indicates where in A(*) the eta file begins.
        LAST <-- Last element of A(*) that is to be written.
        J <-- 0
C
C For simplicity each component of A(*) is assumed to be a byte long.
C KD gets incremented every time a WRITE is executed.
C
        J <--- 0
        DO WHILE (DATSIZ ≥ RECL)
            WRITE(Unit Number'KD) A(J+1:J+RECL)
            DATSIZ <-- DATSIZ - RECL
            J <-- J + RECL
        END
        IF (DATSIZ ≠ 0) THEN
            L <-- RECL - DATSIZ
```

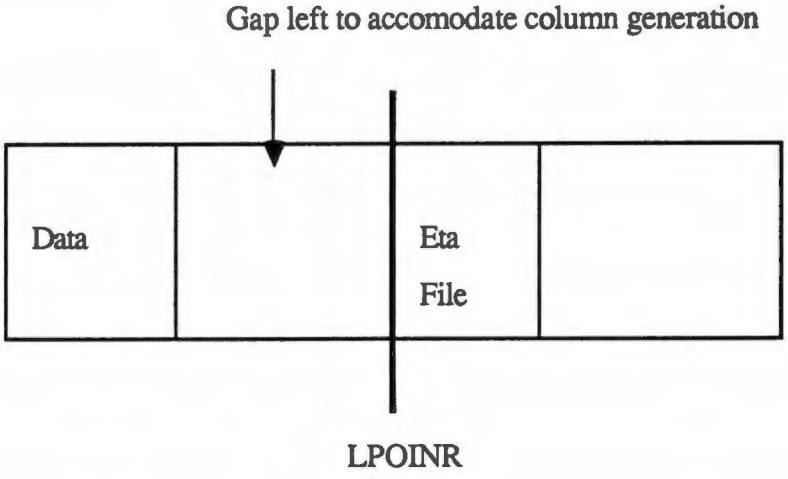


Figure 4.1 : Organization of the data array.

```
        WRITE(Unit Number'KD) A(J+1:J+DATSIZ),A(LN+1:LN+L)
        ETASIZ <-- ETASIZ - L
    END
    J <-- LN + L
    DO WHILE (ETASIZ > RECL)
        WRITE(Unit Number'KD) A(J+1:J+RECL)
        ETASIZ <-- ETASIZ - RECL
        J <-- J + RECL
    END
    WRITE(Unit Number'KD) A(J+1:LAST)
```

The next modification that may be necessary concerns input format for NAMELIST statements rather than the code itself. For example, the input format on the VAX-VMS for the statements :

```
    NAMELIST/NM/ X,Y
    READ(Unit Number,*) NM
```

should be :

```
    NM
        X = Value of X
        Y = Value of Y
    $END
```

On a UNIX based machine the values of X and Y need to separated by blank spaces.

BIBLIOGRAPHY

Bibliography

V. Chvatal, *Linear Programming,* W. H. Freeman, New York, (1983).

G. B. Dantzig, *Linear Programming and extensions,* (Princeton University Press, Princeton, (1963)).

G. B. Dantzig and P. Wolfe, "The decomposition principle for linear programs, " *Operations Research,* 8 (1960) 101-111.

J. J. H. Forrest and J. A. Tomlin, "Updating Triangular Factors of the Basis to Maintain Sparsity in Product Form Simplex Method, " *Mathematical Programming,* 2 (1972) 263-278.

E. Hellerman and D. Rarick, "The Partitioned Preassigned Procedure (P^4), " in D. J. Rose and R. A. Willoughby, eds., *Sparse matrices and their applications* (Plenum, New york, 1972) pp. 67-76.

J. K. Ho, "Implementation and Application of a Nested Decomposition Algorithm, " in W. W. White, ed., *Computers and mathematical programming* (National Bureau of Standards, 1978) pp. 21-30.

J. K. Ho and E. Loute, "An Advanced Implementation of the Dantzig-Wolfe Decomposition Algorithm for Linear Programming, " *Mathematical Programming,* 20 (1981) 303-326.

J. K. Ho and E. Loute, "Computational Experience with Advanced Implementation of Decomposition Algorithms for Linear Programming, " *Mathematical Programming,* 27 (1981) 282-290.

J. K. Ho and E. Loute, "Computational Aspects of DYNAMICO : A Model of Trade and Development in World Economy, " *R.A.I.R.O. Recherche operationelle / Operations Research,* 18 (1984) 403-414.

IBM, "IBM Mathematical Programming System Extended/370 (MPSX/370) Program Reference Manual, " SH19-1095-02, (1979).

L. S. Lasdon, *Optimization Theory for Large Systems,* McMillan, London, (1970).

B. A. Murtagh, *Advanced Linear Programming : Computation and Practice,*

McGraw Hill, New York, (1981).

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

APPENDICES

APPENDIX A

Users Guide

DECOMP is a FORTRAN IV code of the Dantzig-Wolfe algorithm for solving linear programming problems of the form

$$\text{Min} \quad z = \sum (c_r x_r)$$

st

$$\sum (A1_r x_r) = b_0$$

$$A2_r x_r = b_r \qquad r = 1,\ldots,R$$

$$x_r \geq 0 \qquad r = 1,\ldots,R$$

where c_r is 1 by n_r, b_r is m_r by 1 and all other vectors and matrices are of suitable dimensions. DECOMP has been developed by C.Winkler at Systems Optimization Laboratory (SOL). It has been subsequently transformed by J.K.Ho and E.Loute ata CORE, University of Louvain.

A.1. Dimension of Arrays :

The arrays within the program are dimensioned to accomodate up to ten sub-problems, possibly none. Each sub-problem can have up to 400 rows (including the m_0 coupling rows and the objective row), 1000 columns (including logicals, structurals, and rhs columns) and up to 10,000 non-zero elements in the sub-problem (including sub-problem data and eta file). These restrictions apply to master problem with the following remarks :

1. The master problem can have up to 99 rows (including the objective and coupling rows) besides the convexity rows;
2. The number of columns (including logicals and proposals) must be less than 1000;
3. Up to 25 proposals can be generated during a master cycle.

When compiled with FORTRAN H compiler (with option "OPT = 2") DECOMP requires less than 230 K of core storage including the buffers for the three external files. One of the ecternal files is a FORTRAN direct access file of 106 records of 13028 bytes used to store the master problem and sub-problem data. A FORTRAN sequential file is needed for proposal buffering and phase 3 of D-W algorithm. The data of the whole problem can be read on a sequential file

A typical set of JCL cards for executing DECOMP on an IBM/370 in an OS environment is as follows :

```
//JOB Card
//STEP1 EXEC PGM=%%%,TIME=8,REGION=230K
//STEPLIB DD DSN= %%%,VOL=REF=%%%
//FT06F001 DD SYSOUT=A
//FT08F001 DD UNIT=SYSDA,DSN=&SYSUT3,
```

```
//                DISP=(NEW,DELETE),SPACE=(1612,(100,10)),
//                DCB=(RECFM=VBS,LRECL=804,BLKSIZE=1612)
//FT09F001 DD DSN=%%%,VOL=REF=%%%,
//                DISP=(OLD,KEEP),SPACE=(13028,78)
//FT02F001 DD DSN=DECOMP.DAT,VOL=REF=%%%,
//                DISP=(OLD,KEEP)
```

DECOMP is written in FORTRAN IV with some IBM FORTRAN features. It uses a timer routine TICTAC with an INTEGER*4 output argument. The difference of the arguments corresponding to two calls gives the CPU time spent betwween the two calls. If it is not available the user can replace it by a dummy routine :

```
SUBROUTINE TICTAC(ITIME)
INTEGER*4 ITIME
ITIME = 0
RETURN
END
```

A.2. Input Data :

The input data consists of two sections : (a) the control parameters and (b) the master and sub-problemn data. They are read on an unit 2 file called DECOMP.DAT. The control parameters are as follows :

LMAX : is the number of problems and it must be specified.

LPOINR : is the start of the eta file in the master file,

IOBJ : is the number of the objective row to be used among the

common non-binding rows. (default is one),

USERHS : is the number of the rhs column to be used among the rhs columns (default is one),

KINP : is the unit number of the input unit (default is two),

KSTR : is a strategy parameter for D-W algorithm with default value two.

When KSTR = 1, each sub-problem sends one proposal at optimality and two at unboundedness.

When KSTR = 2, each sub-problem is solved until MAXCA proposals have been sent to the master, unless optimality has been reached before. With this strategy a major cycle is interrupted once a total of MAXNCA proposals has been reached. The master problem is then solved and another major cycle is started with the next sub-problem.

When KSTR = 3, then in the first major cycle all sub-problems until they have provided up to MAXCA proposals and together less than MAXNCA proposals and in subsequent major cycles only one sub-problem is solved per cycle in a cyclic manner.

When KSTR = 4, then in a major cycle a sub-problem is solved allowing for up to MAXCA proposals; if the sub-problem is not optimized then a cycle is interrupted and new one is started with the sub-problem as the first sub-problem.

When KSTR = 5, the strategy is similar to KSTR = 2 except that now sub-problems are solved to optimality with only the last MAXCA proposals sent to the master.

MAXNCA : is the maximum number of proposals that can be sent to the master in a major cycle. Maximum is 150 and default is 100.

MAXCA : is the maximum number of proposals that can be sent to the master by a particular sub-problem during a major cycle (default is 10).

NCANFR : is the frequency of proposal generation (default is 15)

INVFRQ : is the frequency of basis re-inversion in the Simplex routine (default is 30).

INVFMA : is the frequency of basis re-inversion for the master problem (default is 20).

ITOL : switch to indicate whether tolerances are to be read.

IRST : is restart paramater indicating when non-zero that the decomposition algorithm is to be restarted at the cycle it was saved in a previous run (default is zero).

ISAVE : parameter which indicates when non-zero that restart information is to be saved when the algorithm is stopped (default is one)

ISOL : when set to one it indicates that a phase 3 is to be performed (default is zero).

ICLAST : maximum number of cycles to be performed (default is 999)

NPH : when one it indicates that each sub-problem is solved to optimality and sends one proposal to the master before the first major cycle.

RPERC : tolerance used in the proposal generation machinism (cf. variable dictionatry) (default is 40 %)

RSTOP : tolerance used in the primal-dual stopping criterion (default is .01 %)

The following tolerances can be specified if ITOL $\neq$ 0 via a NAMELIST called TO : ZTOLZE, ZTOLPV, ZTCOST, ZTOLRJ, ZTCAND. The meanong and default values of these tolerances is given in the variable dictionary.

The problem's data follows the specification of control parameters. The master data comes first and is followed by the sub-problem's data. Each of this is in standard MPS format. The data that is to be input depends on whether it is the master or the sub-problem. Thus there are two cases :

Case 1 : (Master problem)

1. The coupling rows are to be declared in the master problem's ROWS' section.
2. Only the elements corresponding to $\mathbf{A1_0}$ (coupling columns) are to be specified in the COLUMNS' section. If there are no coupling columns then the COLUMNS' section would just be the keyword "COLUMNS"
3. If several rhs columns are defined in the problem, they must be declared in the master and in each sub-problem in the same order.
4. If a rhs column is all zero in the master it must be declared in the rhs section by entering any zero element.

5. The rhs corresponding to coupling rows must be declared once and only once in the rhs section of the master problem.

Case 2 : (Sub-problem)

1. The coupling rows may or may not be declared in the sub-problem data.
2. The COLUMNS' section of the r^{th} (r = 1,...R) sub-problem consists of elements in $[\mathbf{A1}_r^t, \mathbf{A2}_r^t]^t$.
3. If several rhs columns are defined in the problem, they must be declared in the master and in each of the sub-problem in the same order.
4. If a rhs column is all zero in any sub-problem it must be declared in the rhs section by entering any zero element.

As an example consider the problem

$$
\begin{array}{lll}
\text{Min} & -xo - x1 - x2 - 2y1 - y2 & \\
\text{st} & & \\
 & xo + x1 + 2x2 + 2y1 + y2 & \leq 40 \\
 & x1 + 3x2 & \leq 30 \\
 & 2x1 + x2 & \leq 20 \\
 & y1 & \leq 10 \\
 & y1 + y2 & \leq 10 \\
 & y2 & \leq 15 \\
 & xo, x1, x2, y1, y2 & \geq 0
\end{array}
$$

Suppose that the seven rows (starting with the objective row) are named OBJ, RM1, RS11, RS12, RS21, RS22, RS23 respectively and the five columns are named Co, C1, C2, C3, C4 respectively then the input format (DECOMP's format)

for this problem is as shown below.

lpoinr = 5000

lmax = 2

iobj = 1

userhs = 1

kinp = 2

kstr = 1

maxnca = 100

maxca = 10

ncanfr = 15

invfrq = 30

invfma = 20

nph = 1

kmult = 3

itol = 0

icpi = 0

irst = 0

iclast = 999

isave = 1

isol = 1

rperc = 40.0

rstop = 0.01

NAME MASTER

ROWS

N OBJ

```
 L RM1
COLUMNS
 Co        OBJ       -1.0
 Co        RM1        1.0
RHS
  RHS1    RM1       +40.0
ENDATA
NAME        SUB1
ROWS
 N OBJ
 L RS11
 L RS12
COLUMNS
  C1        OBJ       -1.0
  C1        RM1       +1.0
  C1        RS11      +1.0
  C1        RS12      +2.0
  C2        OBJ       -1.0
  C2        RM1       +2.0
  C2        RS11      +3.0
  C2        RS12      +1.0
RHS
  RHS1    RS11      +30.0
  RHS1    RS12      +20.0
ENDATA
```

```
NAME        SUB2
ROWS
 N  OBJ
 L  RS21
 L  RS22
 L  RS23
COLUMNS
    C3        OBJ       -2.0
    C3        RM1       +2.0
    C3        RS21      +1.0
    C3        RS23      +1.0
    C4        OBJ       -1.0
    C4        RM1       +1.0
    C4        RS22      +1.0
    C4        RS23      +1.0
RHS
    RHS1      RS21      +10.0
    RHS1      RS22      +10.0
    RHS1      RS23      +15.0
ENDATA
```

APPENDIX B

Variable Dictionary

The program has four different common storage areas : (a) COMMON BLOCK, (b) COMMON FT09, (c) COMMON INDISK and (d) COMMON blank. COMMON BLOCK contains problem parameters and other data that fix the dimension of the arrays of the program. COMMON FT09 contains information that specify what and where to write on the direct access file. COMMON INDISK contains the information of the problem that is to be swapped between the direct access file and core storage. COMMON blank contains other variables that will be used during the running of the program. The following is a list of variables defined under COMMON statements.

B.1. Common BLOCK :

KMULT : Number of candidates in multiple pricing. Maximum is five. (I*2)

LTRAC : Record Length in bytes of the direct access file used to store master and subproblem data. Set to 13028. (I*2)

MAST : Control variable. Set to zero in BLOCK DATA. Keep value at zero if no sub-problem. Set to one in phase 1

and 2 of D-W algorithm. (in INDATA). Set to 2 in phase 3 (in RESULT). With NPH = 1, MAST keeps value at zero when a sub-problem is solved to optimality right after its data has been read. (I*2).

NCAND : Number of proposals generated so far in current master major cycle and not yet packed. Set to zero in BLOCK DATA. Reset to zero after a pack operation. (I*2).

NEMAX : Maximum number of non-zero elements in matrix and eta files for a sub-problem. The matrix file of the master problem can have up to LPOINR-1 non-zero elements. Its eta file can have up to NEMAX-LPOINR non-zero elements. Set to 10,000. (I*2).

NCRMAX : Maximum number of coupling rows. Set to 99. (I*2).

NLAMAX : Maximum number of columns in master and sub-problem, including logicals. Set to 1000. (I*2).

NPROS : Maximum number of proposals in the proposal buffer. Set to 25. (I*2).

NRMAX : Maximum number of rows in a problem. Set to 400. (I*2).

NTMAX : Maximum number of transforms (eta vectors) in the eta file. Set to 800. (I*2).

NZMAX : Maximum number of entries in a proposal. Also equal to maximum number of coupling rows (including objective) plus one. Set to 100. (I*2).

QQ : Character string 'CROW' used to form convexity row

names. (I*4)

Qx : Character Q concatenated by x where x stands for A, B, C, E, F, H, I, L, N, O, R, U, M, and blank.

ZTCOST : Zero tolerance in reduced cost. Set to 1.0E-04. (R*8).

ZTOLPV : Pivot tolerance. Set to 1.0E-06. (R*8).

ZTOLRJ : Feasibility tolerance used in CHUZR, FORMC, INVERT. Set to 1.0E-07. (R*8).

ZTOLZE : Zero tolerance used for data and computations. Set to 1.0E-07. (R*8).

ZTCAND : Tolerance used for proposal generation. Tolerance also used to determine which coupling constraint must be enforced in phase 3 of D-W algorithm. Set to 1.0E-02. (R*8).

B.2. Common FT09 :

NOLA(*,*) : Control array. NOLA(2,L) points toward end of eta file of sub-problem L (master when L = LMAX+1), i.e. equals LE(NETA+1) - 1. NOLA(3.L) points toward secondary storage of data of sub-problem L (master when L = LMAX+1). NOLA(4,L) points toward secondary storage of rows and columns names of sub-problem L (master when L = LMAX+1). NOLA(1,LMAX+1) points toward secondary storage of restart information. Dimension is NOLA(4,11).

KD : Control variable associated with direct access file 9. (I*4).

LMAXP1 : Number of sub-problems plus one. (I*2).

B.3. Common INDISK :

A(*) : Contains the data matrix. For the sub-problem eta vectors are stored right after the data of the sub-problem For the master, A(1) to A(LPOINR-1) contain the data of the master and eta vectors are stored starting from A(LPOINR) onward. Dimension is A(10000). (R*8).

IA(*) : Row index of the *th element of the matrix or eta file. Dimension is IA(NEMAX). (I*2).

ISTYPE(*) : Indicates the type of the *th row. Zero for non-binding row, one for inequality row and minus one for equality row. Dimension is ISTYPE(NRMAX). (I*2).

ITCNT : Simplex iteration counter. (I*2).

ITSINV : Number of Simplex iterations since last invert. (I*2).

JH(*) : Indicates the column index that is basic in the *th row. Dimension is JH(NRMAX). (I*2).

KINBAS(*) : Indicates the status of the *th column in the current solution. Zero for non-basic and i for basic in row i. Dimension is KINBAS(NLAMAX). (I*2).

LA(*) : Pointer to the *th column in A(*). Dimension is LA(NLAMAX). (I*2).

LE(*)	:	Pointer to the *th eta vector in A(*). Dimension is LE(NTMAX+1). (I*2).
NCOL	:	Number of columns of the problem (logicals, structurals and rhs). (I*2).
NELEM	:	Number of non-zero elements in matrix file. (I*2).
NETA	:	Number of eta vectors currently in the eta file. When NETA is zero it triggers a call to INVERT. (I*2).
NROW	:	Number of rows in the problem currently under consideration. (I*2).
NT	:	Cumulated transformation time in BTRAN and FTRAN (I*4).
NUMCAN	:	Number of proposals generated during last cycle. (I*2)
NX	:	Cumulated I/O time. (I*2).
RHSCOL	:	Index of active rhs column among columns of the problem. (I*2).
ITIM	:	Time when entering NORMAL in a cycle. (I*4).
ITINV	:	Auxilliary variable. (I*4).
JTIM	:	Auxiliary variable. (I*4).
JTINV	:	Cumulated transformation time. (I*4).

B.4. Common Blank :

COND	:	Approximate condition number of the basis inverse. Computed in CHSOL. (R*4).
DCMIN(*)	:	Reduced cost of the *th column selected in multiple

pricing pass. Dimension is DCMIN(5). (R*8).

DE : Value of the column leaving the basis. Also after CHUZR, value at which a candidate column would enter the basis if pivoted in. Used also as auxiliary variable. (R*8).

DEOB(*) : Value of the objective value minus dual variable of the corresponding convexity constraint of the *th sub-problem after treatment in NORMAL. Dimension is DEOB(LMAXP1). (R*8).

DNEW : New value of dual objective function. It is computed in MASTER at the beginning of a cycle when all sub-problems have been solved to optimality. (R*8).

DOB : Objective value minus dual variable of convexity row for sub-problem currently treated by NORMAL. (R*8)

DOBMA : Auxiliary variable used in CHECK. (R*8).

DOLD : Old value of dual objective value when starting a new cycle in MASTER. (R*8).

DP : Value of variable entering the basis. Computed in UPBETA. Tested in CHECK. Used also as auxiliary variable. (R*8).

DPROD : Auxiliary variable. (R*8).

DSUM : Auxiliary variable. (R*8)

DY : Potential improvement in ibjective value after calling CHUZR with JENTRY = 2. Also an auxiliary variable. (R*8).

ERMAX : Maximum row error in absolute value. Computed in CHSOL. (R*8).

ICAND : Pointer toward NCAND for proposal label of last proposal of last sub-problem previously treated. (I*2)

ICASUB(*) : Number of proposals generated at completion of *th sub-problem. Dimension is ICASUB(10). (I*2).

ICLAST : Cycle at which the algorithm is to be stopped. (I*2).

ICOLS(*) : Array used when solving master problem only. Indicates the subproblem origin of the *th proposal. Dimension is ICOLS(NLAMAX). (I*2).

ICPI : Cycle at which a special set of prices is to be read and used instead of master dual vector. (I*2).

ICUMB(*) : Column index of the *th extreme ray generated when treating a sub-problem in a cycle. Dimension is ICUMB(8). (I*2).

IFFEZ : Zero if the problem currently treated is infeasible and one otherwise. (I*2).

INCO : Not used.

INVFMA : Master problem inversion frequency. (I*2).

INVRQ : Sub-problem inversion frequency. (I*2).

IOBJ : Index of the objective row. See user's guide. (I*2).

IPROP(*) : Used to store number of proposals under different criteria.

I	QPROP
1	R

	2	F
	3	A
	4	B
	5	O
	6	U

See QPROP. Dimension is IPROP(6). (I*2).

IPROS : Pointer toward first free position in proposal buffer. (I*2)

IROWP : Pivot row index. (I*2).

IRST : Restart parameter. See user's guide. (I*2).

ISAVE : When equals one it indicates that restart information is to be saved. See user's guide. (I*2).

ISOL : When set to one indicates that a phase 3 is to be initiated after ICLAST cycles or at optimum. (I*2).

ISUNB : When non-zero indicates that the problem is unbounded and how many extreme rays have been generated so far. (I*2).

ITMA : Set to ITCNT when entering NORMAL. (I*2).

ITSLCA : Iteration counter since last extreme point proposal generation. Set to minus 5 when entering NORMAL to treat a sub-problem. (I*2).

IVIN : Index of column entering the basis. (I*2).

IVOUT : Index of column leaving the basis. (I*2).

IYIV : Auxiliary index. (I*2).

JCOLP(*) : Index of the $*^{th}$ column selected as candidates in a

multiple pricing pass. Dimension is JCOLP(5). (I*2).

KCYC : Cycle counter of the D-W algorithm. Equivalenced with MAJOR in MASTER. (I*2).

KFASE : Indicates where in the buffer YA(*,*) the cost vector is to be formed. When one it indicates that the sub-problem currently treated is either not feasible or has lost feasibility. In this case the cost vector is formed in YA(*,5). When two it is formed in YA(*,6) (I*2).

KINP : Unit number of device on which master and sub-problem data is read. (I*4)

KRIT(*) : Indicates stopping criterion when solving sub-problem LS(*). If KRIT(*) is one then no proposal is found to be optimal or inbounded. If KRIT(*) equals two then up to NC(*) proposals can be generated. Dimension is KRIT(10). (I*2).

KSTR : Computational strategy. See user's guide. (I*2).

LCORE : Indicates which sub-problem is currently in core. It is zero when the master is in core. (I*4).

LMAX : Number of sub-problems. (I*2).

LS(*) : Index of the $*^{th}$ sub-problem to be treated in the current cycle. Master problem is treated when LS(*) is zero. Dimension is LS(11). (I*2).

LSUB : Index of sub-problem currently treated and when zero it indicates master problem. (I*2).

MAXCA : Maximum number of proposals generated (kept when KSTR is five) when treating a sub-problem. (I*2).

MAXNCA : Maximum number of proposals generated (kept when KSTR is five) in a master cycle. (I*2).

MSTAT : Current status of the problem currently treated. (I*2).

MSTATU(*) : Status of the *th sub-problem after its treatment. MSTATU(LMAXP1) contains the status of the master. Status is as follows : I for infeasible, N for infeasible and infeasibility proved, N for feasible, blank for optimal and U for unbounded. (I*2).

MULT : Number of selected candidates in multiple pricing pass (I*2).

NAME(*) : Array of proposal labels. Used to form proposal labels stored in array ICOLS. NAME(*) contains the sub-problem number if *th proposal is an extreme point generated when treating the sub-problem. It contains minus the sub-problem index if the proposal is an extreme ray. (I*4).

NC(*) : Number of proposals to be generated (kept when KSTR is five) when treating sub-problem LS(*).

NCANFR : Proposal generation frequency. See user's guide. (I*2)

NCASUB : Number of proposals generated in sub-problem being solved. (I*2).

NCOLO : Number of columns in the master excluding columns generated as proposals (coupling columns). (I*2).

NROWO : Number of coupling rows. (I*2).

NCUMB(*) : Column index of the *th extreme ray generated when treating a sub-problem. The corresponding column is temporarily marked as being fixed to zero. Dimension is NCUMB(5). (I*2).

NLETA : Number of transforms generated when forming the basis L-U factors. (I*2).

NTEMP(*) : Auxiliary array used mainly in input. Dimension is NTEMP(10). (I*2).

QPROP : Variable which contains a character indicating the proposal selection criterion :

blank	no proposal,
R	percentage of improvement,
F	frequency,
A	test A,
B	teat B,
O	optimality,
U	extreme ray.

(I*2).

RCTEST : Testing value used for proposal generation based on reduced cost. Set to (1.0E+4)ZTCOST. (R*4).

RPERC : Minimum percentage of relative improvement in sub-problem objective function to generate a proposal according to profitability criterion. Set to 40 %. (R*4).

RSTOP : Minimum percentage of relative gap between current

primal and best dual solution to stop the algorithm. Set to .01 %. (R*4).

SUMINF : Sum of infeasibilities in a problem. (R*4).

TIMER : Time information for printing purposes. (R*4).

USERHS : Index of active rhs among rhs columns. Default is one. (I*2).

X(*) : Contains the current basic solution. Dimension is X(NRMAX). (R*8).

YA(*,*) : Working storage. Vectors YA(*,1) to YA(*,5) are used to expand the columns selected for multiple pricing and there can be utmost KMULT of these. YA(1,6) to YA(NROWO,6) contains minus the simplex multipliers of the master problem. YA(*,6) is then used to compute minus the simplex multipliers of the sub-problem. When KFASE is one YA(*,5)is used to compute minus the simplex multipliers corresponding to the infeasibility form for a sub-problem. Dimension is YA(NRMAX,6). (R*8).

YTEMP(*) : Minus simplex multipliers of the $*^{th}$ convexity row. Dimension is YTEMP(10). (R*8).

Z(*,*) : Proposal buffer. Dimension is Z(NZMAX,NPROS) (R*8).

VITA

The author R. P. Sundarraj was born in Madras, India, on September 15 1960.

He earned a Bachelor of Engineering degree in Electronics and Communications Engineering from Madras University, India, in May 1982. He came to the United States in the Fall of 1984. He expects to earn a Master of Science degree with a major in Management Science in March 1988. He is now pursuing a doctoral degree in Management Science.