RR-74-22

BASIS FACTORIZATION FOR BLOCK-ANGULAR LINEAR PROGRAMS: UNIFIED THEORY OF PARTITIONING AND DECOMPOSITION USING THE SIMPLEX METHOD

Carlos Winkler

November 1974

BASIS FACTORIZATION FOR BLOCK-ANGULAR LINEAR PROGRAMS: UNIFIED THEORY OF PARTITIONING AND DECOMPOSITION USING THE SIMPLEX METHOD*

Carlos Winkler

Abstract

A General Block-Angular Basis Factorization is developed to represent the inverse of the basis of block-angular linear problems in factorized form. This factorization takes advantage of the structure of the matrix and can be efficiently updated when one column is replaced by another.

Partitioning and Decomposition methods (excluding Dantzig-Wolfe decomposition) for block-angular linear problems with coupling constraints, or coupling variables, or both, are shown to be variants of a Simplex Method using this General Block-Angular Basis Factorization form of the inverse, with various criteria as to the vector pair selected to enter and to leave the basis. By considering other criteria new algorithms are obtained. In particular, algorithms are presented for which at each iteration only a subset of the terms in the factorization needs to be used or to be updated. Preliminary experimental results with such an algorithm for block-angular linear problems with coupling constraints are included.

Results are extended to the case when imbedded in the block-angular structures there are blocks which themselves are of block-angular form. Applications to the solution of dynamic linear programs (staircase structure) are developed.

---

*Submitted to the Department of Operations Research and the committee on Graduate Studies of Stanford University as a dissertation in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

I wish to thank my thesis advisor, Professor George B. Dantzig who encouraged this research, which greatly benefited from his comments and suggestions. Thanks also to the typing and drafting staff of IIASA for the preparation of the final manuscript.

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

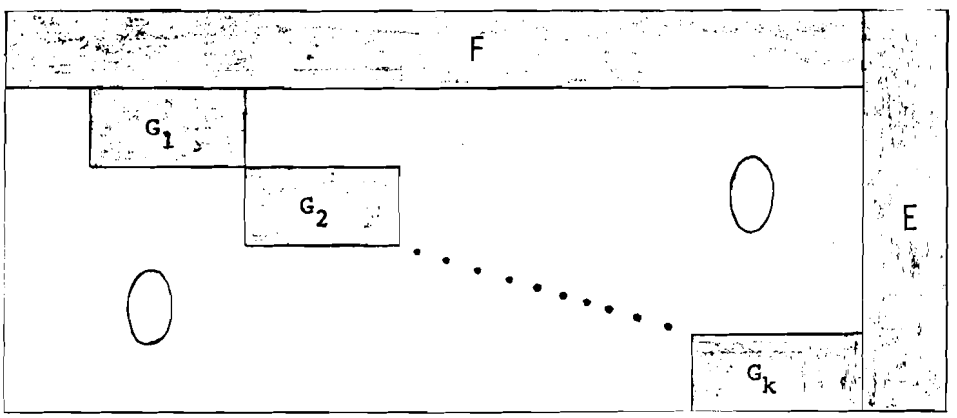
CHAPTER 1

GENERAL

1.1. Introduction and Summary

With the growing awareness of the potentialities of the linear programming approach to both dynamic and static problems of industry, of the economy, and of applied systems analysis, the size of the models has increased to the point where the main obstacles toward full application are the limitations of current computational computer codes to cope with the size of the matrix. However, especially in large-scale models, the matrix usually has special struture, because the system consists of independent subsystems coupled by only a few common constraints or linking variables.

As an example of such a special structure, consider the following matrix where non-zeros appear in a block-angular pattern (shaded areas):



Such a matrix arises in apllications where each small block of non-zeros (G_i) represents the technology matrix of an industry (or sector of the economy), the longer rectangular block F represents the common constraints imposed on them by sharing the same resources, and the thin rectangular block E represents coupling activities.

Many algorithms have been proposed over the years to take adavantage of the special structure of block-angular linear systems. Among those not based on the Dantzig-Wolfe decomposition principle |13|, we have Dantzig and Van Slyke's Generalized Upper Bounding |12|, Balas' Infeasibility Pricing Method |1|, Rosen's Primal Partitioning Method |32|, and the methods of Kaul |22|, Müller-Mehrbach |27|, Bennett |4|, Orchard-Hays |29|, Ohse |28|, Knowles |23|, Beale |3|, Gass |15|, Ritter |31|, Hartmann and Lasdon |20|, etc.

Grigoriadis and White |17|, |19|, shows that many of the methods for block-angular linear problems with coupling constraints can be viewed as having a common data handling structure and differing only in the strategy used as to the vector pair selected to enter and to leave the basis.

In the following we present a block-angular basis factorization theory that provides a unifying framework for partitioning and decomposition methods not based on the Dantzig-Wolfe decomposition principle, which allows us to view them as special instances of the Simplex Method using basis factorization.

In its generality it gives us an additional degree of freedom, since it can be specialized to any of the previous approaches or alternatively to obtain new variants. This, in addition to a more thorough theoretical understanding, allows us to design specialized algorithms to take full advantage of a particular block-angular structure. For block-angular linear problems with coupling constraints such an algorithm has been programmed with good experimental results (see Appendix A). In addition the theory gives us a good starting point for developing nested factorization methods.

In the remainder of this chapter we will clarify the sense in which we use certain concepts and terminologies and motivate the development in later chapters.

In Chapter 2 we develop and validate the General Block-Angular Basis Factorization (GBBF) and show how to update the factorized terms in the representation of the inverse as one column substitutes for another in the basis.

Chapter 3 is devoted to the use of the GBBF in the Simplex Method. First its use in performing the backward and forward transformations is analyzed and its implications on the choice of simplex strategy are discussed. Then some algorithms are developed that take full advantage of the structure, and some consideration is given to alternative ways of implementing them on computer codes.

In Chapter 4 GBBF is used to give a unified presentation of Partitioning and Decomposition methods not based on the Dantzig-Wolfe decomposition principle. Existing methods for

block-angular linear problems with coupling constraints, or coupling variables, or both, are shown to be variants of the Simplex Method using GBBF with various strategies as to the vector pair to enter and to leave the basis. Some new strategies that look promising in conjunction with GBBF are presented.

Chapter 5 is devoted to nested factorizations that arise in cases where some of the components of the original factorization have also a block-angular structure that can conveniently be factorized further. Nested factorization methods to solve staircase problems are analy ed.

Finally in Chapter 6 some comments and conclusions are presented.

Appendix A contains experimental results of tests with a Basis Factorization Algorithm for block-angular linear problems with coupling constraints.

1.2. Concepts, Terminologies and Motivations

It will be convenient to clarify the sense in which we use certain concepts and terminologies.

Simplex Method: Any LP algorithm that follows a path along adjacent basic solutions of the set of linear relations in such a way that no basis is repeated.

Accordingly we distinguish two aspects of the Simplex Method:

Strategy: Rules as to how to move iteratively from one basic solution to the next, i.e. criteria as to the vector pair

selected to enter and leave the basis.

Data-Handling Structure: Information as to what to carry forward, and in what form, from one iteration to the next.

Improvements in the Simplex Method usually involve changing one or both of the above. For example the data-handling structure started in 1947 with the simplex tableau [10]. This was followed by the revised simplex using the explicit inverse, and this was soon followed by the product form of the inverse [11].

Each of these data-handling structures can be combined with any of the selection strategies such as the usual primal, dual or primal-dual selection criteria [7].

A strategy may be efficient with a given data-handling structure and not so with a different data-handling structure. Moreover criteria such as the greatest change in the objective function [40] may be efficient compared to the others if a tableau simplex structure is used, but some other criteria may be better if the product form structure is used.

With the above concepts in mind, the advantages of a general theory become clearer. If we are able to identify or discover a common body of data-handling structures for general block-angular systems, it will be much easier to separate the strategy from the data-handling aspects in the existing algorithms. In an analogous way, in identifying the strategies, it will be much easier to get a feeling for the convergence characteristics (efficiency) of the method by first comparing it with alternative strategies for the general Simplex Method.

Also by studying the original matrix structure and how the data-handling aspects are treated, we may be able to identify a strategy that makes best use of both.

Other advantages are that convergence follows from that of the Simplex Method and this makes it possible to conveniently write one code to test many different methods or strategies.

In the remainder the terminology primal (dual, primal-dual) strategy will be used to refer to the rules used in the primal (Dual, Primal-Dual) Simplex Method as to how to move iteratively from one basic solution to the next.

Nice Properties under the Assumption that the Block-Angular Sub-Matrices are Square and Nonsingular*

To motivate the data-handling aspects, consider the "square" block-angular basis structure.

$$B_N = \begin{bmatrix} I_0 & \cdots & A_i & \cdots & A_k \\ & \ddots & & & \\ & & B_i & & \\ & & & \ddots & \\ & & & & B_k \end{bmatrix}$$

I_0 $m_0 \times m_0$ identity

B_i $m_i \times m_i$ nonsingular

$m_T = \sum_{i=0}^{k} m_i$

This basis has certain nice properties. To see this, consider first a special case, the matrix $\hat{B}_j$ associated with block j and its inverse:

*The actual basis structure of a block-angular linear program need not, of course, have square blocks along the diagonal but later we will associate with it a basis that does.

$$\hat{B}_j = \begin{bmatrix} I & & & & & & \\ & \ddots & & & & & \\ & & I & & & & \\ & & & B_j & & & \\ & & & & I & & \\ & & & & & \ddots & \\ & & & & & & I \end{bmatrix}, \quad \hat{B}_j^{-1} = \begin{bmatrix} I & & & & & & \\ & \ddots & & & & & \\ & & I & & & & \\ & & & B_j^{-1} & & & \\ & & & & I & & \\ & & & & & \ddots & \\ & & & & & & I \end{bmatrix} \tag{1.2}$$

where $\bar{A}_j = A_j B_j^{-1}$.

We can now express

$$B_N = \prod_{i=1}^{k} \hat{B}_i \quad \text{and} \quad B_N^{-1} = \prod_{i=1}^{k} \hat{B}_i^{-1} \tag{1.3}$$

where the terms forming the products can be commuted.

Some of these nice properties of the square block-angular basis are:

1) Instead of inverting one big matrix of dimension $m_T \times m_T$, one can invert k small matrices of dimension $m_i \times m_i$ $(i = 1,\ldots k)$.

2) To represent (in terms of basis) an incoming vector d "belonging" to block j, we have

$$\hat{d} = B_N^{-1} d = \hat{B}_j^{-1} d \quad , \tag{1.4}$$

i.e. we need only the inverse associated with the block. This implies savings in computations and data transfer. To show (1.4) we prove instead $d = B_N \hat{d} = \hat{B}_j \hat{d}$. Partitioning d and $\hat{d}$, we may

write this out more explicitly

$$\hat{d}_0 + \sum_{i=1}^{k} A_i \hat{d}_i = d_0$$

$$B_i \hat{d}_i = d_i \qquad i = 1,\ldots,k \tag{1.5}$$

But since d is in block j, for $i \neq j$ $d_i = 0$, which implies $\hat{d}_i = 0$ for $i \neq j$, since the B_i's are nonsingular. Hence equations (I.5) reduce to

$$\hat{d}_0 + A_j \hat{d}_j = d_0$$

$$B_j \hat{d}_j = d_j \quad . \tag{1.6}$$

But this corresponds to $\hat{B}_j \hat{d} = d$ and hence $\hat{d} = \hat{B}_j^{-1} d$.

3) The "price" vector Π is defined by $\Pi B_N = C$ (see Ch. 3). To calculate the partition Π_j of Π corresponding to block j, we need only compute

$$(C_0, 0,\ldots,0, \Pi_j, 0,\ldots,0) = (C_0, 0,\ldots,0, C_j, 0,\ldots,0)\hat{B}_j^{-1} \quad , \tag{1.7}$$

implying the same kind of savings as in 2). Relation (1.7) follows from the structure of B_N, which implies $\Pi_0 = C_0$ and $\Pi_0 A_i + \Pi_i B_i = C_i$ for $i = 1,\ldots,k$.

Our motivation then is to preserve as much of these nice properties (mentioned above) as we can for the more general case when the block-angular basis arises from problems having either coupling constraints or coupling variables or both.

CHAPTER 2

BLOCK-ANGULAR BASIS FACTORIZATION THEORY

2.1. The Problem

Consider the block-angular linear problem with coupling constraints and variables

$$\max z^{(+)}$$

$$\text{(P)} \qquad \begin{aligned} \text{s.t.} \quad & Uz + D_0x_0 + D_1x_1 + \cdots\cdots\cdots D_kx_k + H_0y = b_0 \\ & G_1x_1 \qquad\qquad\qquad\qquad + H_1y = b_1 \\ & \qquad\ddots \qquad\qquad\qquad\qquad\quad \vdots \\ & \qquad\qquad\ddots \qquad\qquad\quad \vdots \\ & \qquad\qquad\qquad G_kx_k + H_ky = b_k \end{aligned}$$

$$(x_0, x_1, \ldots, x_k, y) \geq 0$$

where U is $m_0 \times 1$, D_i is $m_0 \times n_i$ $i = 0,1,\ldots,k$, H_i is $m_i \times n_{k+1}$ $i = 0,1,\ldots,k$, G_i is $m_i \times n_i$ $i = 1,\ldots k$, b_i is $m_i \times 1$ $i = 0,1,\ldots k$, x_i is $n_i \times 1$ $i = 0,1,\ldots k$, y is $n_{k+1} \times 1$ and z scalar.

(+) We assume that for $\min \{cx: Ax = b, x \geq 0\}$ we let $z = -cx$ and solve $\max \{z: \begin{pmatrix} 1 & c \\ 0 & A \end{pmatrix}\begin{pmatrix} z \\ x \end{pmatrix} = \begin{pmatrix} 0 \\ b \end{pmatrix}, x \geq 0$ and for $\max \{cx: Ax = b, x \geq 0\}$ we let $z = cx$ and $\max \{z: \begin{pmatrix} -1 & c \\ 0 & A \end{pmatrix}\begin{pmatrix} z \\ x \end{pmatrix} = \begin{pmatrix} 0 \\ b \end{pmatrix}, x \geq 0\}$. Thus, both for minimizing and maximizing, we can use a negative reduced cost criterion to indicate that a non-basic column will improve the current solution if it replaces one in the basic set. This will be assumed throughout.

We assume that each of the matrices G_i and (UD_0) has rank equal to its row count. This can always be achieved (if necessary) by augmenting the system with artificial variables with appropriate coefficient structure.

The constraints $Uz + \sum_{i=0}^{k} D_i x_i + H_0 y = b_0$ will be called coupling constraints and rows corresponding to them will also be referred to as common rows. Similarly the y variables will be called coupling variables.

2.2. Constructive Development of the Block-Angular Basis Factorization

Let J_i = {set of indices (of columns) associated with activities in block i} $i = 1,\ldots,k$

J_0 = {indices of columns in D_0}

J_{k+1} = {indices associated with activities y}

$A|_J$ = restriction of matrix A to columns with indices in set J.

Let B_T be a basis for problem (P) and suppose that M is the set of indices of basic columns. Let

$$L_i = M \cap J_i \qquad \text{and consider}$$

$$G_i|_{L_i} \qquad i = 1,\ldots,k \ .$$

Let K_i be the indices of a maximum set of linearly

independent columns in $G_i|_{L_i}$ and*

$$K = \bigcup_{i=1}^{k} K_i \quad .$$

By assumption the rank of G_i is equal to its row count, so that we can augment the columns of $G_i|_{K_i}$ by including enough other columns of G_i to form a basis B_i of linearly independent columns in G_i. Let M_i be the indices of the set of columns of G_i forming B_i (i.e. $B_i = G_i|_{M_i}$).

The nonsingular matrix

$$B_N = \begin{bmatrix} I & A_1 & \cdots & A_k \\ & B_1 & & \\ & & \ddots & \\ & & & B_k \end{bmatrix} \qquad \text{where } A_i = D_i|_{M_i}$$

is square block-angular and has the "nice" properties discussed earlier. We now express the relationship between B_T and B_N in the form of a product:

$$B_T = B_N B_A \tag{2.1}$$

where

$$B_A = B_N^{-1} B_T \tag{2.2}$$

---

* For many practical applications it has been observed that the number of elements in K is close to $\sum_{i=1}^{k} m_i$. It is this that makes the factorization scheme which follows efficient in practice.

The columns of B_A corresponding to K are unit columns so that it is convenient for discussion purposes here to permute its rows and columns so that the units form a submatrix identity I in the lower right partition.

$$\text{permuted } B_A = {}_pB_A = \begin{pmatrix} B_w & \\ V & I \end{pmatrix} , \tag{2.3}$$

where as we have noted the number of columns in I is, for an important class of practical applications, close to that of $\sum_{i=1}^{k} m_i$.

Columns corresponding to K (or to I above) are called trivial, the remaining, $M \cap K^c$ (where K^c is the complement of K) are called non-trivial. We refer to the upper left matrix as the "Working Basis" or "WB" for short.

Without loss of generality we assume that

$${}_pB_A = P\, B_A P \tag{2.4}$$

where P is a permutation matrix satisfying PP = I.

We can further factorize ${}_pB_A$ into

$${}_pB_A = \begin{pmatrix} B_w & \\ V & I \end{pmatrix} = \begin{pmatrix} B_w & \\ & I \end{pmatrix} \begin{pmatrix} I_w & \\ V & I \end{pmatrix} = {}_p\hat{B}_w\, {}_p\hat{V} \tag{2.5}$$

and hence express the basis in factorized form as the product

$$B_T = B_N\, P\, {}_p\hat{B}_w\, {}_p\hat{V}\, P \quad , \tag{2.6}$$

or by permuting again the factors ${}_p\hat{B}_w$ and ${}_p\hat{V}$ (i.e. $\hat{B}_w = P_p\hat{B}_w P$ and $\hat{V} = P_p\hat{V}P$)

$$B_T = B_N \hat{B}_w \hat{V} \quad . \tag{2.7}$$

Lemma 1: B_w, the Working Basis, is nonsingular.

Proof: Obviously B_w is square. Hence

$$0 \neq \det B_T = \det B_N \det \hat{B}_w \det \hat{V} \quad .$$

Since permutations do not change the absolute value of the determinant

$$|\det \hat{V}| = |\det {}_p\hat{V}| = 1$$

and

$$\det \hat{B}_w \neq 0 \quad .$$

Moreover, by permuting $\hat{B}_w$ we get

$$0 \neq \det \begin{pmatrix} B_w \\ & I \end{pmatrix} = \det B_w \quad . \qquad ||^*$$

Hence we can work with the following factorized representation for the inverse

$$B_T^{-1} = \hat{V}^{-1} \hat{B}_w^{-1} B_N^{-1} \quad . \tag{2.8}$$

---

* Double slashes will be used for end of proof.

For applications it is not necessary to permute the matrices $\hat{B}_w$ and $\hat{V}$ to have rows and columns of B_w and V (see (2.5)) in the upper left and lower left corners respectively. However, for the development of the formulas for updating the factorized representation of the inverse when one column replaces another in the basis, it will be convenient, for notational purposes, to work with the permuted matrices. Therefore let

$$ {}_pB_T = P\, B_T\, P \quad \text{and} \quad {}_pB_N = P\, B_N\, P \ . \tag{2.9} $$

Then from (2.6)

$$ {}_pB_T = {}_pB_N\, {}_p\hat{B}_w\, {}_p\hat{V} \ . \tag{2.10} $$

Notice that expression (2.10) differs from (2.7) only in that all terms are permuted. Thus, for simplicity, in what follows the left subscript p will be dropped when working with the permuted matrices, since this will be clear from the context.

2.3. Some Properties of the Factorized Representation of the Inverse

Recalling the nice properties of square block-angular systems, we see for the general block-angular case that in addition to the block-inverses we have to carry the inverse of the Working Basis and the matrix V of $\hat{V}$. Hence under the assumption that the dimension m_w of B_w is "small" relative to m_T, or more precisely that the number of non-zeros in V and B_w^{-1} (or some representation of B_w^{-1}) is "small", the additional

amount of information stored and manipulated will be small. In particular, with regard to preserving as much as possible of the nice properties:

1) Instead of inverting one big $m_T \times m_T$ matrix we can still invert and maintain k small $m_i \times m_i$ matrices ($i = 1,\ldots,k$). However, in addition an $m_W \times m_W$ Working Basis will need to be inverted and maintained; also V will be needed.

2) The first step in updating a vector from block j proceeds the same as that described earlier - and hence the same computational advantages carry through. However, in addition we have to use B_W^{-1} and V. Hence if, as we have assumed, the non-zeros in B_W^{-1} and V are low relative to those of the block inverses $\hat{B}_i$, $i \neq j$, not required in the first step, we will get savings in the forward transformation over a direct representation of B_T^{-1} .

3) For calculating a Π_j the situation is similar to that of the updates in (2). As will be shown in Chapter 3 there is the additional advantage that when the basic variables which correspond to columns not in the Working Basis are feasible, then the $\hat{V}$ matrix is not needed in the backward transformation. This is always the case in Phase 2.

Because no simple statement can be made at this point on how much work is required to update the factorized representation of the inverse (after the replacement of one column in the basis

by another), we will defer discussion of this to later. In section 2.4 we show how to do this updating efficiently.

Thus with the additional effort to maintain and to make use of B_w^{-1} and V, we can carry over much of the desirable properties of independent square block-angular problems. If the dimension m_w of B_w is not too large (relatively) and the additional work in updating the factorized representation of the inverse is not too excessive, we can expect the block-angular factorization method to be more efficient than working directly on the basis B_T using general methods.

We now explore these points more deeply. First we introduce some notation. We classify columns as being either Type A or Type B.

Type A: Those that, except for the common rows, have non-zeroes in rows corresponding to at most one block i = 0,1,...,k, i.e. those with indices belonging to $J_A = \bigcup_{i=0}^{k} J_i$.

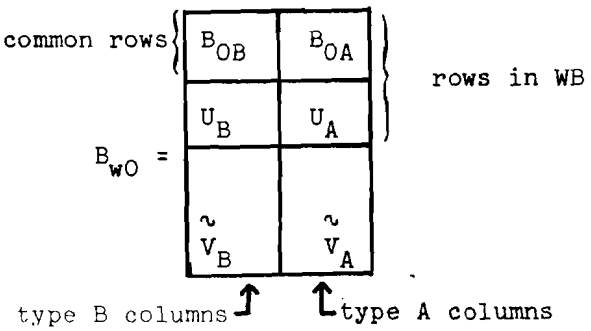
Type B: Otherwise, i.e. $J_B = J_{k+1}$.

Furthermore, the basic columns of Type A are further subclassified into

Type A1: Those basic columns associated with block i, for i = 1,...,k (i.e. Type A columns), that belong to their own block basis B_i.

Type A2: Otherwise, i.e. basic columns associated with block i, for i = 1,...,k, that belong to the Working Basis.

Let B_{wO} be the matrix of columns common to B_T and the Working Basis. Partition B_{wO} according to



Let

$$\tilde{B}_{wO} = B_N^{-1} B_{wO} = \begin{pmatrix} B_w \\ V \end{pmatrix} = \begin{bmatrix} \hat{B}_{OB} & \hat{B}_{OA} \\ \hat{U}_B & \hat{U}_A \\ V_B & V_A \end{bmatrix}, \tag{2.11}$$

i.e.

$$B_W = \begin{bmatrix} \hat{B}_{OB} & \hat{B}_{OA} \\ \hat{U}_B & \hat{U}_A \end{bmatrix} \tag{2.12}$$

partitioned as above.

We call a column that is in B_N but not in B_T a pseudobasic column; its corresponding variable will be referred to as pseudobasic also.

Recall from section 2.2 that K_i was chosen to have the indices of a maximum set of linearly independent columns in $G_i|_{L_i}$ and that $M \cap K^c$ contains the indices of columns in the Working Basis (where K^c is the complement of $K = \bigcup_{i=1}^{k} K_i$).

Observe that:

a) The maximum sets of linearly independent columns in $G_i|_{L_i}$ (i = 1,...,k) need not be unique. If this is the case we could choose the indices of columns in any such set to be in K_i and hence in K. Thus alternative factorizations are possible that lead to different Working Basis's of the same dimension.

b) If K_i is not required to contain the indices of a maximum set, but only of a subset of linearly independent columns in $G_i|_{L_i}$, then the resulting factorization would have a Working Basis of higher dimension (less indices in K, more in K^c, i.e. of columns in WB).

Following the constructive procedure in section 2.2 we always obtain a Working Basis with the smallest possible dimension. For the case when one column replaces another in the basis, we want to obtain the new factorization from the old one. Therefore it is convenient to establish some easy way to check conditions for a Working Basis being minimal (i.e. there being no alternative factorization giving rise to a WB of smaller dimension).

Theorem 1: The Working Basis is minimal if and only if $\hat{U}_A = 0$.

Proof: Let B_W be minimal. Assume (on the contrary) that $\hat{U}_A \neq 0$. Pick a non-zero element of $\hat{U}_A$ and suppose it is on row j in the partition corresponding to some block i. This non-zero element is used as a pivot to replace the pseudobasic

columns of block basis i associated with row j. The new B_N now includes one more vector previously in the Working Basis. Thus, the new WB will have one less non trivial vector--a contradiction !

$$\text{WB minimal} \longrightarrow \hat{U}_A = 0$$

Now suppose $\hat{U}_A = 0$ and the Working Basis is not minimal. Then for at least one block i (for i = 1,...,k) there is a set of linearly independent columns among those with indices in $M_i \cup L_i$ that constitutes a basis and that does not include at least one of the pseudobasic variables (i.e. those with indices in $M_i \cap L_i^c$ where L_i^c is the complement of L_i). Suppose this new block basis B_i^{new} is partitioned as

$$B_i^{new} = \begin{pmatrix} {}_1B^1 & {}_1B^2 & {}_1B^3 & {}_1B^4 \\ {}_2B^1 & {}_2B^2 & {}_2B^3 & {}_2B^4 \\ {}_3B^1 & {}_3B^2 & {}_3B^3 & {}_3B^4 \\ {}_4B^1 & {}_4B^2 & {}_4B^3 & {}_4B^4 \end{pmatrix}$$

with superscripts

1 : basic columns that remained

2 : pseudobasic columns that remained

3 : new columns (previously in WB partition) that have replaced basic columns (possible none)

4 : new columns (previously in WB) that have replaced pseudobasic columns (at least one)

and left subscripts

1 : rows in which basic columns that remain were basic

2 : rows in which pseudobasic columns that remain were basic

3 : rows in which basic columns replaced were basic

4 : rows in which pseudobasic columns replaced were basic.

Then pre-multiplying B_i^{new} by B_i^{-1}

$$B_i^{-1}\, B_i^{new} = \begin{pmatrix} P_1 & 0 & {}_1\hat{B}^3 & {}_1\hat{B}^4 \\ 0 & P_2 & {}_2\hat{B}^3 & {}_2\hat{B}^4 \\ 0 & 0 & {}_3\hat{B}^3 & {}_3\hat{B}^4 \\ 0 & 0 & {}_4\hat{B}^3 & {}_4\hat{B}^4 \end{pmatrix}$$

where P_1 and P_2 are permutations of identities.

But $\begin{pmatrix} {}_2\hat{B}^3 & {}_2\hat{B}^4 \\ {}_4\hat{B}^3 & {}_4\hat{B}^4 \end{pmatrix}$ has as coefficients those of columns that were in the WB in rows corresponding to the U partition. Thus they constitute a subset of coefficients of $\hat{U}_A$ and hence they are all 0. But this implies that rows corresponding to left subscript 4 (at least one) are 0 and hence that B_i^{new} is singular, which is a contradiction.

$$\therefore \quad \hat{U}_A = 0 \longrightarrow \text{WB minimal}$$

$$\text{and} \quad \text{WB minimal} \longleftrightarrow \hat{U}_A = 0 \quad . \qquad \|$$

Lemma 2: The dimension m_w of a minimal Working Basis satisfies

$$m_w \leq m_O + m_B \leq m_O + n_{k+1} \qquad (2.13)$$

where m_B is the number of coupling variables in the basis.

Proof: For B_w, a minimal Working Basis $\hat{U}_A = 0$; thus

$$B_w = \begin{pmatrix} \hat{B}_{OB} & \hat{B}_{OA} \\ \hat{U}_B & 0 \end{pmatrix} \quad .$$

Suppose U_B is $m_R \times m_B$ where m_B is the number of type B variables (coupling variables) in the basis. Then

$$m_w = m_O + m_R \quad .$$

Now for B_w to be nonsingular $\hat{U}_B$ has to have full row rank. This requires

$$m_R \leq m_B$$

and hence

$$m_w \leq m_O + m_B \leq m_O + n_{k+1} \quad . \qquad \|$$

2.4. Updating the Factorized Representation of the Inverse

Before presenting a procedure for updating the representation of the factorized inverse after the replacement of one column in the basis by another, some results that are needed later will be developed. It will be convenient to use * as a superscript to denote a matrix in the updated representation to distinguish it from the corresponding matrix before the updating. Also, unless stated otherwise, partitions of $m_T \times m_T$ matrices will be assumed to have been permuted to correspond with those of the factorization, i.e. so as to have rows and columns in the Working Basis in the upper left corner.

2.4-1. Increase or Reduction in the Dimension of the Working Basis

Some of the update situations will involve an increase or a reduction in the dimension of the Working Basis. In developing the updating formulas for these cases we assume that the inverse is given in product form.

We want to decrease the dimension of a Working Basis when it has a structure such as

$$B_W = \begin{pmatrix} B_W^* & \\ v & 1 \end{pmatrix} \text{ to another with structure } \begin{pmatrix} B_W^* & \\ & 1 \end{pmatrix}$$

which can substitute for it in applications of a product form representation.

Let

$$E_v = \begin{pmatrix} I & \\ -v & 1 \end{pmatrix},$$

then

$$B_w E_v = \begin{pmatrix} B_w^* & \\ v & 1 \end{pmatrix} \begin{pmatrix} I & \\ -v & 1 \end{pmatrix} = \begin{pmatrix} B_w^* & \\ & 1 \end{pmatrix}$$

and

$$\begin{pmatrix} B_w^{*-1} & \\ & 1 \end{pmatrix} = E_v^{-1} B_w^{-1} , \tag{2.14}$$

and hence it is accomplished by adding an elementary row eta to the representation of the inverse.

Similarly, to add a row, i.e. to get from

$$B_w \text{ or } \begin{pmatrix} B_w & \\ & 1 \end{pmatrix} \text{ to } \begin{pmatrix} B_w & \\ v & 1 \end{pmatrix},$$

$$\begin{pmatrix} B_w & \\ v & 1 \end{pmatrix}^{-1} = E_v \begin{pmatrix} B_w^{-1} & \\ & 1 \end{pmatrix} \tag{2.15}$$

which again is accoplished through an elementary row transformation.

2.4-2 General Updating Formulas

Theorem 2: Let E, E_N be the elementary matrices that update B_T^{-1} and B_N^{-1}, i.e. $B_T^{*-1} = EB_T^{-1}$ and $B_N^{*-1} = E_N B_N^{-1}$, and let

$$E = \begin{pmatrix} E_1 & E_2 \\ E_3 & E_4 \end{pmatrix} \text{ and } E_N = \begin{pmatrix} E_N^1 & E_N^2 \\ E_N^3 & E_N^4 \end{pmatrix}$$

correspond to the partitioning used in the factorization (which is assumed not to change). Further suppose $E_2 = 0$ or $E_N^3 = 0$, then

$$B_w^{*-1} = (E_1 - E_2 V)B_w^{-1}(E_N^1)^{-1} \quad . \tag{2.16}$$

Proof: We have

$$B_T^{*-1} = \hat{V}^{*-1}\hat{B}_w^{*-1}B_N^{*-1} = EB_T^{-1} = E\hat{V}^{-1}\hat{B}_w^{-1}B_N^{-1}$$

or

$$\hat{B}_w^{*-1} = \hat{V}^* E\hat{V}^{-1}\hat{B}_w^{-1}B_N^{-1}B_N^* \quad .$$

But

$$B_N^{-1}B_N^* = E_N^{-1} \quad .$$

Let

$$\tilde{E}_N = \begin{pmatrix} \tilde{E}_N^1 & \tilde{E}_N^2 \\ \tilde{E}_N^3 & \tilde{E}_N^4 \end{pmatrix} = E_N^{-1} \quad ,$$

then

$$\hat{B}_w^{*-1} = \hat{V}^* E\hat{V}^{-1}\hat{B}_w^{-1}\tilde{E}_N \quad ,$$

and writing this product in partitioned form

$$\begin{pmatrix} B_w^{*-1} & \\ & I \end{pmatrix} = \begin{pmatrix} I_w & \\ V^* & I \end{pmatrix}\begin{pmatrix} E_1 & E_2 \\ E_3 & E_4 \end{pmatrix}\begin{pmatrix} I_w & \\ -V & I \end{pmatrix}\begin{pmatrix} B_w^{-1} & \\ & I \end{pmatrix}\begin{pmatrix} \tilde{E}_N^1 & \tilde{E}_N^2 \\ \tilde{E}_N^3 & \tilde{E}_N^4 \end{pmatrix}$$

and now restricting ourselves to the rows and columns in the Working Basis,

$$B_w^{*-1} = (E_1\ E_2)\begin{pmatrix} I_w & \\ -V & I \end{pmatrix}\begin{pmatrix} B_w^{-1} & \tilde{E}_N^1 \\ & \tilde{E}_N^3 \end{pmatrix}$$

$$B_w^{*-1} = \left\{(E_1 - E_2V), E_2\right\}\begin{pmatrix} B_w^{-1} & \tilde{E}_N^1 \\ & \tilde{E}_N^3 \end{pmatrix}$$

$$B_w^{*-1} = (E_1 - E_2V)B_w^{-1}\,\tilde{E}_N^1 + E_2\tilde{E}_N^3$$

Since E_N is an elementary column matrix we have

$$E_N = \begin{cases} \begin{pmatrix} I & E_N^2 \\ & E_N^4 \end{pmatrix} \rightarrow E_N^{-1} = \begin{pmatrix} I - E_N^2(E_N^4)^{-1} & \\ & (E_N^4)^{-1} \end{pmatrix} \rightarrow \begin{cases} \tilde{E}_N^1 = (E_N^1)^{-1} \\ \tilde{E}_N^3 = 0 = -E_N^3(E_N^1)^{-1} \end{cases} \\ \text{or} \\ \begin{pmatrix} E_N^1 & \\ E_N^3 & I \end{pmatrix} \rightarrow E_N^{-1} = \begin{pmatrix} (E_N^1)^{-1} & \\ -E_N^3(E_N^1)^{-1} & I \end{pmatrix} \rightarrow \begin{cases} \tilde{E}_N^1 = (E_N^1)^{-1} \\ \tilde{E}_N^3 = -E_N^3(E_N^1)^{-1} \end{cases} \end{cases}$$

Hence in either case we get the same expressions for $\tilde{E}_N^1$ and $\tilde{E}_N^3$. Subsituting above, we obtain

$$B_w^{*-1} = (E_1 - E_2V)B_w^{-1}(E_N^1)^{-1} - E_2E_N^3(E_N^1)^{-1} \quad .$$

But under the conditions in the hypothesis $E_2E_N^3 = 0$, so that

$$B_w^{*-1} = (E_1 - E_2V)B_w^{-1}(E_N^1)^{-1} \quad . \qquad ||$$

As will be seen in section 2.4-3, most of the update situations can be arranged to satisfy the conditions of the above theorem and usually $(E_N^1)^{-1} = I$ and $E_2 = 0$, so that $B_w^{*-1} = E_1B_w^{-1}$, or under conditions such that it simplifies to $B_w^{*-1} = (I_w - \eta v)B_w^{-1}$. The following results will always allow us to express these updating relationships as product of elementary transformation matrices.

Theorem 3: Let $\eta \epsilon R^m$ be a column vector and $v \epsilon R^m$ a row vector. Suppose $v\eta - 1 \neq 0$, then $I_m - \eta v$ is nonsingular. Furthermore if $v_p \neq 0$ is a component of v then

$$(I_m - \eta v) = E_{R_1}E_{C_1}E_{R_2} \qquad (2.17)$$

where E_{R_1} and E_{R_2} are the elementary row matrices given by

$$E_{R_1} = \begin{pmatrix} I_1 & & \\ -v_1/v_p & -a/v_p & -v_2/v_p \\ & & I_2 \end{pmatrix} \qquad (2.18)$$

$$E_{R_2} = \begin{pmatrix} I_1 & & \\ bv_1 & bv_P & bv_2 \\ & & I_2 \end{pmatrix} \tag{2.19}$$

$a \neq 0$, $b \neq 0$ arbitrary constants, $v = (v_1, v_P, v_2)$, and E_C is an elementary column matrix given by

$$E_{C_1} = \begin{pmatrix} I_1 & -(1/b)\eta_1 & \\ & \bar{\eta}_P & \\ & -(1/b)\eta_2 & I_2 \end{pmatrix} \tag{2.20}$$

with $$\bar{\eta}_p = -\frac{1-v\eta}{ab} \quad \text{and} \quad \eta = \begin{pmatrix} \eta_1 \\ \eta_P \\ \eta_2 \end{pmatrix}. \tag{2.21}$$

Proof: Note that if $v = 0$, the theorem is trivially true. If not then there exists some $v_P \neq 0$. It is easy to verify by direct multiplication that

$$(I_m - \eta v) = E_{R_1} E_{C_1} E_{R_2}$$

and therefore

$$\det(I_m - \eta v) = \det E_{R_1} \cdot \det E_{C_1} \cdot \det E_{R_2}$$

$$= \left(-\frac{a}{v_P}\right)\left(\frac{1-v\eta}{ab}\right)\left(bv_P\right) \neq 0$$

if $(1 - v\eta) \neq 0$. ||

Corollary:

$$(I - \eta v)^{-1} = \begin{pmatrix} I_1 & & -v_2 \\ -v_1 & & v_P \\ & & l_2 \end{pmatrix} \begin{pmatrix} I_1 & & \frac{\eta_1}{b\eta} \\ & & 1/-\eta \\ & & \frac{\eta_2}{b\eta} \end{pmatrix} \begin{pmatrix} I_1 & & \\ -v_1 & -v_P & -v_2 \\ & a & a \\ & & l_2 \end{pmatrix}$$

Another important point to consider is the possibility of an alternative minimal basis. Especially of interest is the case where we can replace a vector in a block basis by a vector in the Working Basis belonging to the same block.

Theorem 4: Let column b with index in J_i be the basic activity associated with row r in forming the inverse of block i. Let $v_r = (v_B, v_A)$ be the corresponding row of V. Then if $v_A \neq 0$, any one of the columns corresponding to a non-zero component of v_A, say v_{A_j} (i.e. the j-th column in B_w), can be exchanged with b to give a new basis for block i and a new Working Basis. Moreover

$$B_w^{*-1} = E_R B_w^{-1} \tag{2.23}$$

where E_R is an elementary row matrix given by

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

and if B_w is minimal, so is B_w^*.

Proof: Recall that by (1.4) for any column from some block P,

$$\hat{d} = B_N^{-1}d = \hat{B}_P^{-1}d$$

and hence by (1.6) $\hat{d}$ can have non-zeros only in the common rows and in the rows of its own block P. It follows, since all columns in the partition corresponding to v_A are of Type A, that under the conditions of the hypothesis any column corresponding to a non-zero component v_{A_j} of v_A must belong to block i and can replace the activity basic in row r of the block basis since its pivot element is different from 0. Recall that B_w minimal implies $\hat{U}_A = 0$ so that

$$\begin{pmatrix} B_w \\ \\ V \end{pmatrix} = \begin{pmatrix} \hat{B}_{OB} & \hat{B}_{OA} \\ \hat{U}_B & \hat{U}_A \\ V_B & V_A \end{pmatrix} = \begin{pmatrix} \hat{B}_{OB} & \hat{B}_{OA} \\ \hat{U}_B & 0 \\ V_B & V_A \end{pmatrix},$$

and hence the updated vector j that will be exchanged with b has zeros in the rows corresponding to the partition $(\hat{U}_B\hat{U}_A)$. Thus the eta vector will have zeros there and all the remaining columns will be unchanged in these rows. Also the representation of the exchanged vector b in terms of the new block basis corresponds to the eta vector so that $U_A^* = 0$ and B_w^* minimal.

The exchange corresponds to a simple permutation of columns, so that $B_T^* = B_T E$, E a simple permutation matrix, for

which $E^{-1} = E$ and $B_T^{*-1} = EB_T^{-1}$. Also $B_N^{*-1} = E_N B_N^{-1}$ with

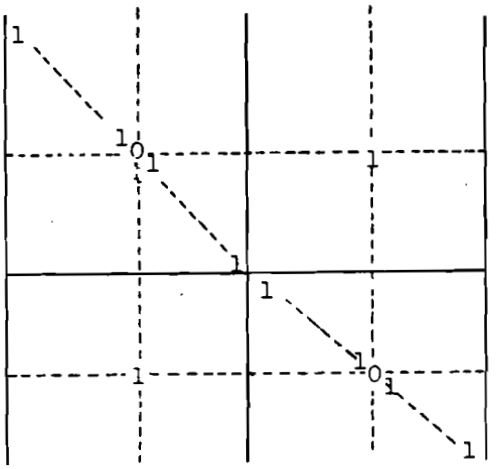
$$E_N = \begin{pmatrix} I & E_N^3 \\ & E_N^4 \end{pmatrix} ,$$

since the pivoting occurs in a row not in the Working Basis. Thus the updating formula (2.16) becomes

$$B_w^{*-1} = (E_1 - E_2 V) B_w^{-1} .$$

Also

$$E = \begin{pmatrix} E_1 & E_2 \\ E_3 & E_4 \end{pmatrix} =$$



so that $E_2 V$ has zeros in all rows except row j, and E_1 is an identity except for row j which is zero. Hence

$$(E_1 - E_2 V) = \begin{pmatrix} 1 & & & \\ & \ddots & & \\ & & 1 & \\ & & -v_r & \\ & & & \ddots \\ & & & & 1 \end{pmatrix} = E_R$$

$$(E_1 - E_2V) = \begin{pmatrix} 1 & & & \\ & \ddots & & \\ & & 1 & \\ & & -v_{r_1} & \\ & & & \ddots \\ & & & & 1 \end{pmatrix} = E_R$$

and

$$B_w^{*-1} = E_R B_w^{-1} \quad . \qquad \| \; .$$

2.4-3 An Updating Procedure

The replacement of one outgoing column (OC) from the basis by another, the incoming column (IC), gives rise to four somewhat different updating cases:

1) IC of Type A and OC in Working Basis
2) IC of Type A and OC in B_N
3) IC of Type B and OC in Working Basis
4) IC of Type B and OC in B_N.

In Fig. 1 we give a flow-sheet of an efficient updating procedure covering all four cases for the factorized representation of the inverse after the replacement of one column in the basis by another.

We can compactly state some of the important features of the updating procedure in the form of a theorem, and then develop it in greater detail in a constructive way in the proof.

Theorem 5 (Updating Procedure): The flow-sheet in Fig. 1 gives a valid procedure for updating the factorized representation of

the inverse after the replacement of one column in the basis by another. In particular, if the old Working Basis was minimal so will be the new one, and (except when a pseudobasic variable is driven out of some block basis to keep the Working Basis minimal (see ** in Fig. 1)) at most one block inverse needs to be updated due to the replacement of only one column in it by another (in the exception at most two columns are replaced in the block basis's).

Proof (Validation of the Updating Procedure): Referring to Fig. 1 we point out that since all tests are of the yes-no type it suffices to show that each path gives a correct updating procedure for the case it involves.

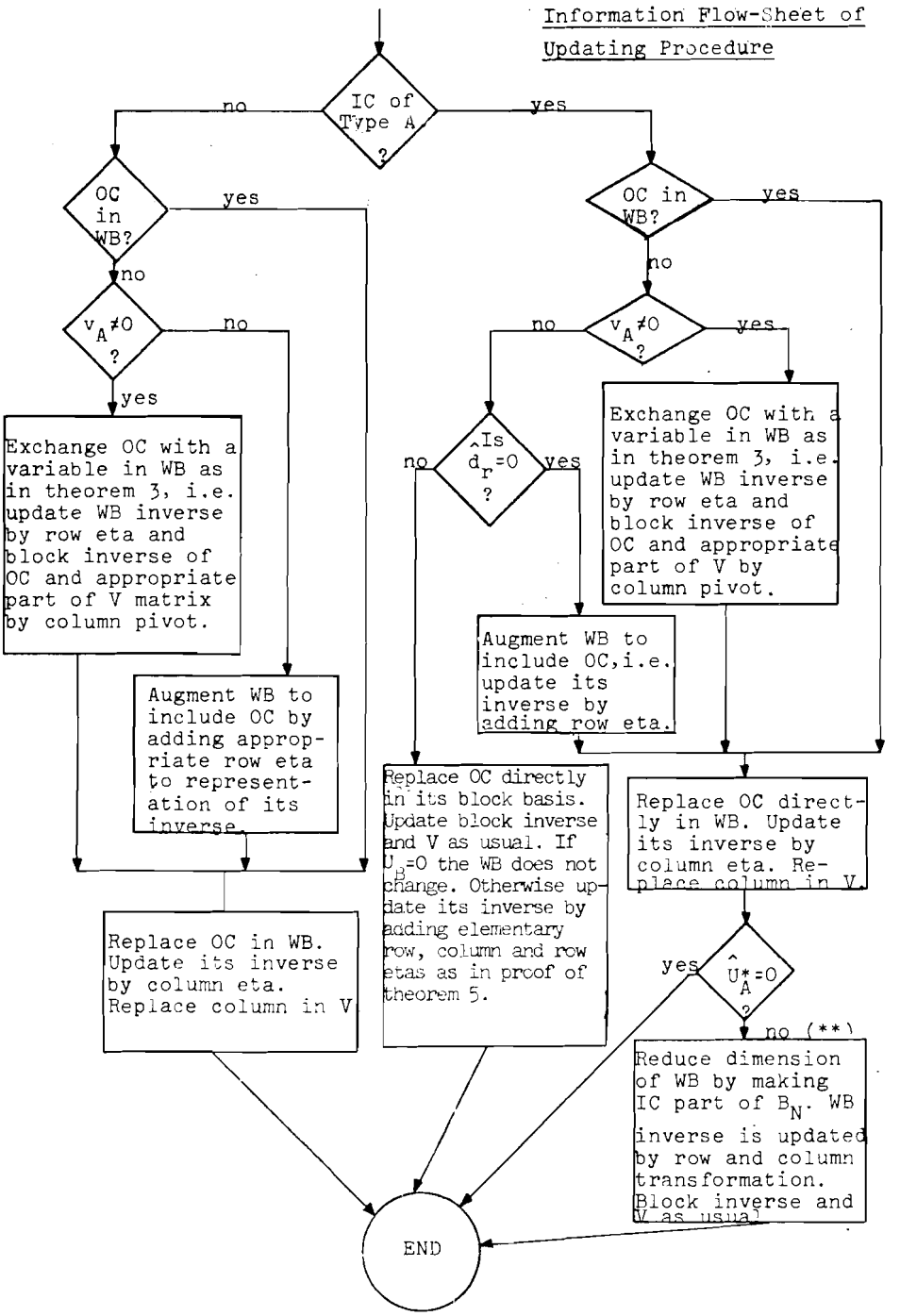
Case I. Incoming Column of Type B

Case I-a. Outgoing Column in Working Basis

Since we start with a minimal Working Basis, $\hat{U}_A = 0$. Letting $B_N^* = B_N$, the updating corresponds to changing one column in $\tilde{B}_{wO}$ (see (2.11)). If the outgoing column is of Type B, then $\hat{U}_A^* = \hat{U}_A = 0$ and the new Working Basis is minimal. If the outgoing column is of Type A2, then after the exchange, $\hat{U}_A^*$ is equal to $\hat{U}_A$ without the column corresponding to the outgoing column (which is now in $\hat{U}_B^*$) and hence $\hat{U}_A^* = 0$ and the new Working Basis is minimal.

The elementary column matrix E that updates $B_T^{*-1} = EB_T^{-1}$ has its pivot element in some row in the Working Basis and hence

FIGURE 1


Information Flow-Sheet of Updating Procedure
IC of Type A ?
no
yes
OC in WB?
yes
no
$v_A \neq 0$?
no
yes
Exchange OC with a variable in WB as in theorem 3, i.e. update WB inverse by row eta and block inverse of OC and appropriate part of V matrix by column pivot.
Augment WB to include OC by adding appropriate row eta to representation of its inverse.
Replace OC in WB. Update its inverse by column eta. Replace column in V
OC in WB?
yes
no
$v_A \neq 0$?
no
yes
Is $\hat{d}_r = 0$?
no
yes
Exchange OC with a variable in WB as in theorem 3, i.e. update WB inverse by row eta and block inverse of OC and appropriate part of V by column pivot.
Augment WB to include OC, i.e. update its inverse by adding row eta.
Replace OC directly in its block basis. Update block inverse and V as usual. If $U_B = 0$ the WB does not change. Otherwise update its inverse by adding elementary row, column and row etas as in proof of theorem 5.
Replace OC directly in WB. Update its inverse by column eta. Replace column in V.
yes
$\hat{U}^*_A = 0$?
no (**)
Reduce dimension of WB by making IC part of B_N. WB inverse is updated by row and column transformation. Block inverse and V as usual
END

$$E = \begin{pmatrix} E_1 & 0 \\ E_3 & I \end{pmatrix} .$$

Also, since $B_N^* = B_N$ we have $E_N = I$ and the updating formula (2.16) in theorem 2 reduces to

$$B_w^{*-1} = E_1 B_w^{-1} .$$

Also V changes only in the column of the outgoing variable, which is replaced by the partially updated incoming column IC, i.e. B_N^{-1} (IC), restricted to rows not in WB. Notice that all the necessary information is generated during the first step in the forward transformation.

Case I-b. Outgoing Column in Some Block Basis

Suppose the outgoing variable belongs to block j and corresponds to row r of the inverse. Let $v_2 = (v_B, v_A)$ be the corresponding row of $V = (V_B, V_A)$. If $v_A \neq 0$ pick a component, say $v_{A_i} \neq 0$. By theorem 4 we can assign the outgoing variable to WB and replace it in the block basis by the column corresponding to v_{A_i}, obtaining a new expression for the WB, $B_w^{*-1} = E_R B_w^{-1}$. Besides the block inverse j and V have to be updated by a simple column pivot. After this exchange the outgoing variable is in the Working Basis and we are back to case I-a.

If $v_A = 0$ the dimension of the Working Basis is increased by one to include the pivot row and the outgoing variable. This corresponds to going from

$$B_w = \begin{pmatrix} B_{OB} & B_{OA} \\ U_B & U_A \end{pmatrix} \xrightarrow{\text{to}} \begin{pmatrix} B_{OB} & B_{OA} \\ U_B & U_A \\ v_B & v_A \quad 1 \end{pmatrix} = B_w^*$$

As shown in section 2.4-1, the inverse of B_w^* is obtained from B_w^{-1} (see (2.10)) by pre-multiplying by the elementary row matrix

$$E_R = \begin{pmatrix} I & \\ -v_r & 1 \end{pmatrix} .$$

Now the outgoing column is in the WB and we proceed as in case I-a. Since $\hat{U}_A^* = \begin{pmatrix} \hat{U}_A \\ v_A \end{pmatrix} = 0$ the resulting WB is minimal.

Case II. Incoming Column of Type A

Let d be the incoming column and

$$\hat{d} = B_N^{-1} d \quad , \quad \bar{d} = B_T^{-1} d .$$

Let $\hat{d} = \begin{pmatrix} \hat{d}_O \\ \hat{d}_A \\ \bar{d}_v \end{pmatrix}$, $\bar{d} = \begin{pmatrix} \bar{d}_O \\ \bar{d}_A \\ \bar{d}_v \end{pmatrix}$

be partitioned as B_{wO} (see also (2.11)).

Also let $\hat{d}_r$ and $\bar{d}_r$ be the elements of $\hat{d}$ and $\bar{d}$ on the pivot row.

Case II-a. Outgoing Column in Working Basis

Replace the outgoing column directly in the Working Basis. This corresponds to updating as in case I-a. Let $\bar{B}_w^{-1} = B_w^{*-1} = E_1 B_w^{-1}$. Then for $\bar{B}_w$ we have $\hat{U}_A^* = (\hat{U}_A \hat{d}_A)$, i.e. it consists of zeros except

possibly for the column corresponding to the incoming variable (i.e. $\hat{d}_A$). If $\hat{d}_A = 0$ the Working Basis is minimal and we finish by updating V as in case I-a.

Otherwise pick an element $\hat{d}_{A_r} \neq 0$ and use it as pivot to introduce the incoming column into its block basis, displacing a pseudobasic variable. This corresponds to $B_T^* = B_T$ and $B_N^{*-1} = E_N B_N^{-1}$, i.e.

$$E = I \text{ and } E_N = \begin{pmatrix} E_N^1 & \\ E_N^2 & I \end{pmatrix} \text{(see theorem 2)}$$

where

$$E_N^1 = \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & \eta & 1 & \\ & & & & \ddots \\ & & & & & 1 \end{pmatrix} \text{ with } \eta_i = \begin{cases} 1/\hat{d}_{A_r} & i = r \\ -\hat{d}_i/\hat{d}_{A_r} & \text{otherwise, for } i \text{ a row index in WB.} \end{cases}$$

Accordingly the update formula (2.16) reduces to

$$B_w^{*-1} = \bar{B}_w^{-1}(E_N^1)^{-1} = \bar{B}_w^{-1}\tilde{E}_N$$

where

$$\tilde{E}_N = \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & \hat{d}_w & 1 & \\ & & & & \ddots \\ & & & & & 1 \end{pmatrix}$$

and $\hat{d}_w$ is the restriction of $\hat{d}$ to rows in the Working Basis.

These changes correspond to the following two steps:

1) From B_w to $\bar{B}_w$ replace one column in B_w (without loss of generality the last one),

$$\text{i.e.} \qquad \bar{B}_w = \begin{pmatrix} \tilde{B}_{OB} & \tilde{B}_{OA} & \hat{d}_O \\ \tilde{U}_B & 0 & \hat{d}_A \end{pmatrix} \tag{2.25}$$

2) From $\bar{B}_w$ to $E_N^{-1}\bar{B}_w$: pivot on row r (without loss of generality the last),

$$\text{i.e.} \qquad E_N^{-1}\bar{B}_w = \begin{pmatrix} \tilde{\tilde{B}}_{OB} & \tilde{\tilde{B}}_{OA} & 0 \\ \tilde{\tilde{U}}_B & 0 & U_r \end{pmatrix} \tag{2.26}$$

where U_r is a unit vector with a unit component on row r. Now we are in the situation of reducing the size of the WB by pre-multiplying it by an elementary row matrix as discussed in section 2.4-1 to obtain a new minimal WB. Letting B_w^* denote the resulting WB we have

$$B_w^{*-1} = E_R(\bar{B}_w^{-1}\tilde{E}_N) = E_R E_1 B_w^{-1}\tilde{E}_N \tag{2.27}$$

with

$$E_r = \begin{pmatrix} I_1 & & \\ & v_r/\hat{d}_{A_r} & \\ & & I_2 \end{pmatrix} \text{ (see (2.14)).} \tag{2.28}$$

Formula (2.27) gives the expression to update the WB in the case when $\hat{d}_A \neq 0$. It is also necessary to update V. This is done by

deleting the column corresponding to the outgoing variable and updating the columns corresponding to type B by applying the same elementary transformation matrix used to update the block inverse when it was modified.

In some computer systems it is inefficient to add new information (in our case etas) to the beginning and end of a file (in our case the eta file). To get around this difficulty we can make use of the following equivalent expression for (2.27).

Proposition 1: Expression (2.27) can also be represented in product form as

$$B_w^{*-1} = E_C E_R E_1 B_w^{-1} \tag{2.29}$$

where E_R and E_1 are as in (2.23) and

$$E_c = \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & \eta_c & & \\ & & & 1 & \\ & & & & \ddots \\ & & & & & 1 \end{pmatrix}, \text{ with } \eta_{c_i} = \begin{cases} -\bar{U}_i / \bar{U}_r & i \neq r \\ 1 / \bar{U}_r & i \neq r \end{cases}$$

and, letting U_r be the r-th unit vector:

$$\bar{U} = (E_R E_1 B_w^{-1}) U_r \quad . \tag{2.30}$$

The proof of proposition 1 will be deferred to the end of the section in order not to disrupt the presentation of the

updating procedure.

Case II-b. Outgoing Column not in Working Basis

Sub-case II-b-1. $v_A \neq 0$

By theorem 4 it is possible to assign the outgoing column to WB, obtaining a new minimal WB whose inverse differs from the old one by an elementary row transformation. The exchange also implies updating the appropriate block inverse because of the replacement of the outgoing column by its exchange vector from the WB, and modifying V due to the changes in the block inverse. Now we are back to case II-a.

Sub-case II-b-2. $v_A = 0$

Sub-sub-case II-b-2-a. $\hat{d}_r = 0$

Augment the Working Basis to include the OC. As seen before in section 2.4-1, this corresponds to adding an elementary row transformation to the old Working Basis inverse according to (2.15). Since $v_A = 0$, after replacing the OC by the incoming column in the WB, the form of the WB is given by (2.25) and thus augmenting the Working Basis we fall back to case II-a.

Sub-sub-case II-b-2-b. $\hat{d}_r \neq 0$

In this case the outgoing column can be replaced in its block basis directly by the incoming column. This implies updating the block inverse as usual by adding a column eta to its representation. Columns of V corresponding to Type B variables

must also be updated by the same elementary column matrix.

As for the WB, since we are pivoting on a row not in the WB, we have for the elementary matrices in theorem 2

$$E_N = \begin{pmatrix} I_w & E_N^2 \\ & E_N^4 \end{pmatrix} \quad ; \quad E = \begin{pmatrix} I_w & E_2 \\ & E_4 \end{pmatrix}$$

where the columns of E_2 are 0 except for the pivot column which is $-\left(\frac{\bar{d}_w}{\bar{d}_r}\right)$

with $\bar{d}_w$ the restriction of $\bar{d}$ to rows in WB (recall $\bar{d}_r$ is the pivot element). Thus (2.16) reduces to

$$B_w^{*-1} = (I_w + \left(\frac{\bar{d}_w}{\bar{d}_r}\right) v_r) B_w^{-1} \tag{2.31}$$

since

$$E_2 V = -\frac{\bar{d}_w}{\bar{d}_r} v_r = -\frac{\bar{d}_w}{\bar{d}_r} (v_B, 0) \quad . \tag{2.32}$$

Now there are two possibilities:

A) $v_B = 0$, i.e. $v_r = (v_B, 0) = 0$, and so

$B_w^{*-1} = B_w$, i.e. the WB does not change.

B) $v_B \neq 0$. Consider

$$\bar{d} = \begin{pmatrix} \bar{d}_w \\ \bar{d}_s \end{pmatrix} = B_T^{-1} d = \hat{V}^{-1} \hat{B}_w^{-1} B_N^{-1} d = \hat{V}^{-1} \hat{B}_w^{-1} \hat{d}$$

$$\begin{pmatrix} \bar{d}_w \\ \bar{d}_s \end{pmatrix} = \begin{pmatrix} I_w & \\ -V & I \end{pmatrix} \begin{pmatrix} B_w^{-1} & \\ & I \end{pmatrix} \begin{pmatrix} \hat{d}_w \\ \hat{d}_s \end{pmatrix} = \begin{bmatrix} B_w^{-1}\hat{d}_w \\ \hat{d}_s - VB_w^{-1}\hat{d}_w \end{bmatrix}$$

and

$$\bar{d}_s = \hat{d}_s - V\bar{d}_w \quad ,$$

in particular for the r-th component

$$\bar{d}_r = \hat{d}_r - v_r\bar{d}_w \quad ;$$

and since $\bar{d}_r$ is the pivot element it is non-zero and we can divide by it

$$1 = \frac{\hat{d}_r}{\bar{d}_r} - v_r \frac{\bar{d}_w}{\bar{d}_r} \quad ,$$

i.e.
$$-1 - v_r \frac{\bar{d}_w}{\bar{d}_r} = -\frac{\hat{d}_r}{\bar{d}_r} \neq 0 \quad . \tag{2.33}$$

Hence, since (2.31), (2.33) and $v_B \neq 0$ satisfy its hypothesis we can use theorem 3. Choose some column P with $v_{B_P} \neq 0$, and $b = -1$, $a = \hat{d}_r \neq 0$. Then for this case, according to relations (2.17) through (2.21)

$$B_w^{*-1} = E_{R_2} E_C E_{R_1} B_w^{-1} \tag{2.34}$$

with

$$E_{R_1} = \begin{pmatrix} I_1 & & \\ -v_1 & -v_{B_P} & -v_2 \\ & & I_2 \end{pmatrix} \quad ; \quad E_C = \begin{pmatrix} I_1 & -\bar{d}_1/\bar{d}_r & \\ & 1/\bar{d}_r & \\ & -\bar{d}_2/\bar{d}_r & I_2 \end{pmatrix} \quad ;$$

$$E_{R_2} = \begin{pmatrix} I_1 & & \\ -v_1/v_{B_P} & -\hat{d}_r/v_{B_P} & -v_2/v_{B_P} \\ & & I_2 \end{pmatrix}$$

where we have partitioned

$$v_r = (v_1, v_{B_P}, v_2) \quad \text{and}$$

$$\bar{d}_w = \begin{pmatrix} \bar{d}_1 \\ \bar{d}_P \\ \bar{d}_2 \end{pmatrix}.$$

To show that the Working Basis in (2.34) is minimal, recall that the same columns remain in the Working Basis, and only their representation in terms of B_N (see (2.2) and (2.3)) may have changed due to pivoting on the r-th row of B_N. But because $v_A = 0$ all type A2 columns have zeros in the pivot row of B_N and remain unchanged. Thus $\hat{U}_A^* = \hat{U}_A = 0$ and the Working Basis in (2.34) is minimal.

This finishes case II-b, and now all four possible update

cases have been convered. By following all paths in the Updating Procedure we see that at most one vector is replaced among those in the block basis's, except when reducing the dimensionality of the Working Basis (see ** box in lower right corner of Fig. 1), in which case it could be two. ||

Proof of Proposition 1: Recall from (2.27) that

$$B_w^{*-1} = E_R E_1 B_w^{-1} \tilde{E}_N \quad .$$

Let

$$A = B_w E_1^{-1} E_R^{-1} \quad . \tag{2.35}$$

Then

$$B_w^* = \tilde{E}_N^{-1} A \quad . \tag{2.36}$$

Without loss of generality we take the pivot column to be the last. Recall that $\tilde{E}_N$ and E_R pivot on the same row, which we again can take to be the last. Then

$$B_w E_1^{-1} = \begin{pmatrix} \hat{B}_{OB} & \hat{B}_{OA} & \hat{d}_O \\ \hat{U}_B & 0 & \hat{d}_A \\ v_B & 0 & \hat{d}_r \end{pmatrix}$$

$$A = B_w E_1^{-1} E_R^{-1} = \begin{pmatrix} \hat{B}_{OB} & \hat{B}_{OA} & \hat{d}_O \\ \hat{U}_B & 0 & \hat{d}_A \\ v_B & 0 & \hat{d}_r \end{pmatrix} \begin{pmatrix} I_O & & \\ & I_B & \\ -v_B/d_r & & 1 \end{pmatrix}$$

$$A = \begin{pmatrix} \tilde{\tilde{B}}_{OB} & \hat{B}_{OA} & \hat{d}_O \\ \tilde{\tilde{U}}_B & 0 & \hat{d}_A \\ 0 & 0 & \hat{d}_r \end{pmatrix} \tag{2.37}$$

and so the r-th row of A is $a_r = (0,\ldots,0,\hat{d}_r)$. Now $\tilde{\tilde{E}}_N^{-1}$ pivots on the last row and reduces the last column of A to the r-th unit vector. Hence

$$B_w^* = \begin{pmatrix} \tilde{\tilde{B}}_{OB} & \hat{B}_{OA} & 0 \\ \tilde{\tilde{U}}_B & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} ;$$

thus B_w^* is obtained from A by replacing the last column in A by the r-th unit vector. Hence letting

$$\bar{U} = A^{-1}U_r \quad , \quad \eta_{c_i} = \begin{cases} -U_i/\bar{U}_r & i \neq r \\ 1/\bar{U}_r & i \neq r \end{cases}$$

and

$$E_c = \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & \eta_c & 1 & \\ & & & & \ddots \\ & & & & & 1 \end{pmatrix}$$

we have

$$B_w^{*-1} = E_c A^{-1} \quad ,$$

and from (2.35)

$$B_w^{*-1} = E_c E_R E_l B_w^{-1} \quad . \qquad \|$$

CHAPTER 3

USING THE GENERAL BLOCK-ANGULAR BASIS FACTORIZATION (GBBF) IN THE SIMPLEX METHOD

In a revised Simplex Method (see [2]), a representation of the inverse is needed for performing two types of calculations:

1) Solving the system

$$\Pi B_T = C \text{ for } \Pi \quad ,$$

i.e.
$$\Pi = CB_T^{-1} \quad , \tag{3.1}$$

which is computed using the backward transformation (BTRAN) in product form algorithms (see [37]). Here C is the vector of coefficients in the objective function of the basic variables for primal strategies, or the r-th unit vector in dual strategies.

2) Solving the system

$$B_T\bar{d} = d \text{ for } \bar{d} \quad ,$$

i.e.
$$\bar{d} = B_T^{-1}d \quad , \tag{3.2}$$

which is computed using the forward transformation (FTRAN) in product form algorithms. In the section we show how these calculations can be carried out efficiently using GBBF.

3.1. The Backward Transformation (BTRAN)

Using the GBBF representation of the inverse, we can write for (3.1)

$$\Pi = CB_T^{-1} = C\hat{V}^{-1}\hat{B}_W^{-1}B_N^{-1} \quad . \tag{3.3}$$

Define $\hat{C} = C\hat{V}^{-1}$ and $\hat{\Pi} = \hat{C}\hat{B}_W^{-1}$. We can consider the backward transformation as consisting of three steps:

Step 1: Calculate $\hat{C} = C\hat{V}^{-1}$

Step 2: Calculate $\hat{\Pi} = \hat{C}\hat{B}_W^{-1}$

Step 3: Calculate $\Pi = \hat{\Pi}B_N^{-1}$

which we will now analyze separately.

Let the row vectors $= (C_0, C_1, \ldots, C_k)$, $\Pi = (\Pi_0, \Pi_1, \ldots, \Pi_k)$, $\hat{C} = (\hat{C}_0, \hat{C}_1, \ldots, \hat{C}_k)$ and $\hat{\Pi} = (\hat{\Pi}_0, \hat{\Pi}_1, \ldots, \hat{\Pi}_k)$ be partitioned according to rows in block $0,1,\ldots,k$. Similarly let V_i be the restriction of V to rows in block i. Furthermore, for $i = 1,\ldots,k$ we partition $C_i = (C_i^0, C_i^1)$, $\hat{C}_i = (\hat{C}_i^0, \hat{C}_i^1)$, $\hat{\Pi}_i = (\hat{\Pi}_i^0, \hat{\Pi}_i^1)$, where the subscript 0 corresponds to rows of block i for which the basic variable is in the WB, and the superscript 1 to those basic in their own block. Similarly we partition $V_i = (V_i^0, V_i^1)$ where V_i^0 corresponds to columns basic in the common rows and V_i^1 to the other columns in the Working Basis.

For simplicity let also

$$C_0^1 = (C_1^0,\ldots,C_k^0) \quad \text{and} \quad \hat{C}_0^1 = (\hat{C}_1^0,\ldots,\hat{C}_k)$$

and let I_0^1 be an $\hat{m}$ x $\hat{m}$ identity where $\hat{m}$ is the dimension of C_0^1.

3.1-1 Step 1

By rearranging rows and columns (if necessary) we can express $\hat{C} = C\ \hat{V}^{-1}$ as

$$(\hat{C}_0,\hat{C}_0^1,\hat{C}_1^1,\ldots,\hat{C}_k^1) = (C_0,C_0^1,C_1^1,\ldots,C_k^1) \begin{pmatrix} I_0 & & & \\ & I_0^1 & & \\ -V_1^0 & -V_1^1 & I_1^1 & \\ & & & \ddots \\ -V_k^0 & -V_k^1 & & I_k^1 \end{pmatrix}$$

from which we obtain the following relationships:

$$\hat{C}_0 = C_0 - \sum_{i=1}^{k} C_i^1\ V_i^0$$

$$\hat{C}_0^1 = C_0^1 - \sum_{i=1}^{k} C_i^1\ V_i^1 \qquad (3.4)$$

$$\hat{C}_i^1 = C_i^1 \qquad i = 1,\ldots,k \ .$$

w we have to consider three cases.

Case 1 : During Phase 2

Here $C = (1,0,0,\ldots,0)$ (assuming the objective function is the first row in the common rows), and hence $C_i = 0$ $i = 1,\ldots,k$. Thus from relationships (3.4) we obtain $\hat{C} = C$ and hence Step 1 is not required during Phase 2.

Case 2 : Phase 1, variables not in WB feasible

Again $C_i^1 = 0$ for $i = 1,\ldots,k$ and hence $\hat{C} = C$ and Step 1 is not required.

Case 3 : Phase 1, some variables not in WB infeasible

In this case in general $\hat{C} \neq C$ and we have to go through Step 1. However, if we are minimizing an unweighted sum of infeasibilities, then the components of C take on values 0, 1 or -1*, and hence, as can be observed from relations (3.4), no multiplications or divisions will be necessary, but only additions or subtractions.

3.1-2 Step 2

Recall that $\hat{\Pi} = \hat{C}\ \hat{B}_w^{-1}$. This is an ordinary backward transformation, and hence the number of operations required for its calculation will be proportional to the number of non-zero in the representation of the inverse of the Working Basis.

---

* The infeasibility form can be expressed as $\min \left(\sum_{i \in S_1} X_i - \sum_{i \in S_2} X_i \right)$ where $S_1 = \{i : X_i > 0$ basic and artificial$\}$ and $S_2 = \{i : X_i < 0$ basic$\}$. Thus the components C_i take on values 1, -1 or 0 according to i belonging to S_1, S_2 or none of them (see also [2]).

3.1-3 Step 3

Rewriting $\Pi = \hat{\Pi} B_N^{-1}$ as $\Pi B_N = \hat{\Pi}$, from the structure of B_N (see (1.1)) we see that

$$\Pi_0 = \hat{\Pi}_0$$

and

$$\Pi_0 A_i + \Pi_i B_i = \hat{\Pi}_i \qquad \text{for } i = 1,\ldots,k$$

or

$$\Pi_i = (\hat{\Pi}_i - \Pi_0 A_i) B_i^{-1} \tag{3.5}$$

or

$$(\Pi_0,0,\ldots,0,\Pi_i,0,\ldots,0) = (\hat{\Pi}_0,0,\ldots,0,\hat{\Pi}_i,0,\ldots,0)\hat{B}_i^{-1}. \tag{3.6}$$

That is, for any $i = 1,\ldots,k$, calculating Π_i is equivalent to an ordinary backward transformation using the representation for the inverse of its block basis, and hence the number of operations is also proportional to the number of non-zeros in this representation.

3.2 The Forward Transformation (FTRAN)

Using the GBBF we can write for (3.2)

$$\bar{d} = \hat{B}_T^{-1} d = \hat{V}^{-1}\, \hat{B}_w^{-1}\, B_N^{-1} d \quad . \tag{3.7}$$

f the incoming column belongs to block i, then by (1.4)

$$B_N^{-1} d = \hat{B}_i^{-1} d \quad \text{and} \quad \bar{d} = B_T^{-1} d = \hat{V}^{-1}\, \hat{B}_w^{-1}\, \hat{B}_i^{-1} d \quad . \tag{3.8}$$

That is, we need only its own block inverse, the Working Basis inverse and the V matrix to perform the calculations of the forward transformation.

3.3. Implications for the Choice of Simplex Strategy*

By using the factorized representation of the inverse and by taking full advantage of the structure, appreciable savings can be obtained in the number of operations that have to be performed and in the amount of data required in both the backward and forward transformations. In particular:

1) Whenever all blocks are feasible the V matrix is not used in the backward transformation. Hence a good strategy would be to make all blocks feasible in the beginning.

2) When using a primal simplex strategy with partial pricing (see [30]) to coincide with columns in a block (or in some blocks), only the Π_i's corresponding to that block (or blocks) have to be calculated, with considerable savings in the backward transformation with respect to the case when a general representation is used for B_T^{-1}. When using a dual simplex strategy the whole pivot row has to be updated ("priced out") and hence partial pricing is not possible. However, there is one special case when using GBBF which is formalized in the following lemma.

---

* Strategy is used here as defined in section 1.2, i.e. rules as to how to move iteratively from one basic solution to the next.

Lemma 3: If the outgoing variable belongs to block i and corresponds to row r of its block basis inverse, and if the corresponding row of V, $v_r = 0$, then the solution to the system $\Pi^r B_T = U_r$, where U_r is the r-th unit vector, is given by

$$\Pi^r_j = 0 \qquad \forall j \neq i$$

and

$$\Pi^r_i = U_r \hat{B}_i^{-1} \quad .$$

Proof: $\Pi^r = U_R B_T^{-1} = U_r \hat{V}^{-1} \hat{B}_w^{-1} B_N^{-1}$. But $U_r \hat{V}^{-1} = U_r$ since $v_r = 0$, and $U_r \hat{B}_w^{-1} = U_r$ since pivot row is not in WB. Thus $\Pi^r = U_r B_N^{-1}$ or equivalently $\Pi^r B_N = U_r$, from which the lemma follows because of the special structure of B_N (see (1.1)). ||

Hence, for updating the whole pivot row we need in this case to perform only one backward transformation using only one block inverse, and to price out only the columns on that block and the columns of the coupling variables, since all the others will price out to 0. The above is a generalization of a result of Ohse [28].

3) Except for the coupling columns the forward transformation also requires only the use of one of the block inverses. Also the updating of the factorized representation of the inverse simplifies considerably in the absence of coupling columns (see Fig. 2 in section 3.4). This suggests a special treatment for coupling columns.

4) Savings occur also in total time spent in inversions,

since on the average smaller matrices will be inverted and the number of inversions, not counting those of the Working Basis, will remain roughly the same.

With these points in mind we first turn to develop an efficient strategy for the block-angular linear problem with coupling constraints only.

3.4. A Strategy for Block-Angular Linear Problems with Coupling Constraints

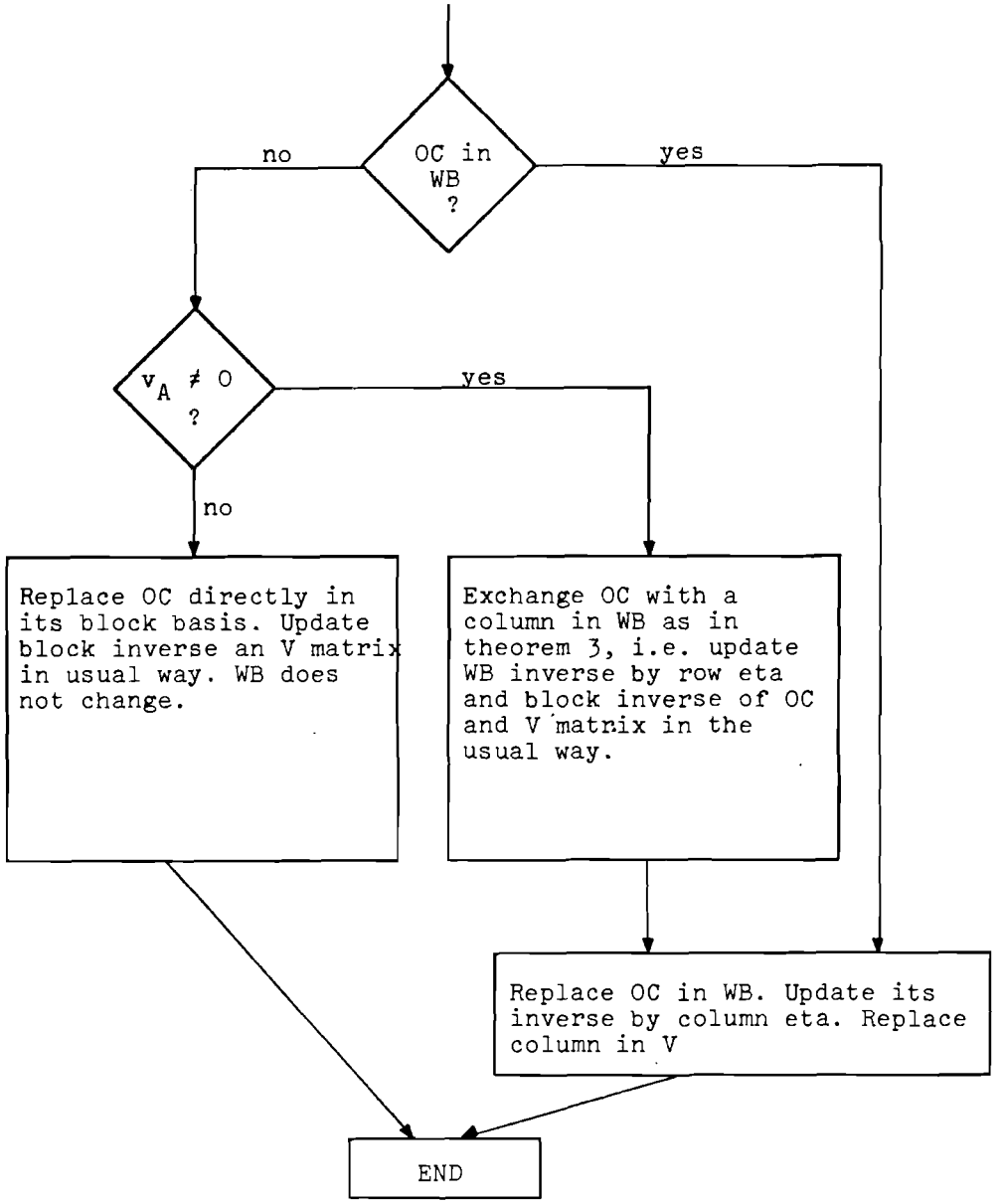
In contrast to block columns, coupling columns require the use of all block inverses both for the backward and for the forward transformation. They also tend to increase the size of the Working Basis and to complicate the updating procedure. For all these reasons they should be treated differently: for instance, to consider them as candidates to enter the basis and to price them out only when no improvements can be made with block variables, or to treat them as fixed parameters whenever possible. Hence the strategy for the special case of a block-angular problem with coupling constraints will play an important role in that of the more general problem P.

3.4-1 Simplifications in the Updating Procedure

Since there are no coupling columns in this case all variables are of Type A and the updating procedure given in Fig. 1 reduces to that in Fig. 2. There are now only three updating cases which depend exclusively on the position of the outgoing variable in the basis. Also since the Working Basis includes

FIGURE 2

Information Flow-Sheet of Updating Procedure for Block-Angular Linear Problems with Coupling Constraints



all common rows, and as a consequence of Lemma 2, for this case we have that the Working Basis will have constant size $m_0 \times m_0$.

3.4-2 Strategy Considerations

Dual and parametric strategies for block-angular linear problems with coupling constraints (see section 4.1) are based on the observation that if all blocks are independently sub-optimized at the beginning then the resulting solution is dual feasible for the overall problem. Hence the first step in these methods is to sub-optimize all blocks.

On the other hand, in order to make full use of the reduction in computations and data transfer in the backward transformation for primal strategies when using GBBF, it is necessary to have all blocks feasible.

Alternatively we could sub-optimize them on the heuristic that later we could approach feasibility in the common rows "from above" (in the maximizing case) and could expect to arrive at the feasible region with a higher value of the objective function, and hence would have fewer iterations in Phase 2. Primal strategies offer the following advantages during this first step:

1) They do not require a dual feasible solution after solving all blocks, allowing us to stop before reaching optimality if this is considered convenient to save iterations.
2) For the same reason they do not require any special treatment if some block has an unbounded solution or if the matrix $D_0 \neq 0$ (see problem P in 2.1).

After this first step primal strategies have the advantage of allowing savings in BTRAN if we make use of partial pricing. Since for large problems partial pricing is desirable to reduce overall computation time, the added benefits of reducing the amounts of data and of computations required in the backward transformation make it even more attractive here. We will refer to the use of the partial pricing technique applied to the columns of a block as the Partial Block Pricing strategy, or PBP for short.

The above considerations lead to the following two-step primal strategy:

Step 1: Optimize all block problems (alternatively stop once feasibility is reached). If some block has no feasible solution STOP, the whole problem is infeasible. Otherwise proceed to Step 2.

Step 2: Use the partial block-pricing strategy to take advantage of the savings in BTRAN that are made possible by the factorized representation of the inverse. This step terminates in one of the usual primal termination states, i.e. no feasible solution, unbounded solution or optimal solution.

3.4-3 Experimental Results

The above two-step strategy for block-angular linear problems with coupling constraints has been coded in FORTRAN IV in a program called G-GUB. Preliminary experimental results look

promising and are reported in [38]. In Appendix A we give a more detailed flow-sheet of the algorithm resulting from the application of the above two-step strategy and reproduce some of the results in [38].

In the following we will refer to this algorithm as the Coupling Constraints Algorithm (or CCA for short). For other strategies for block-angular linear problems with coupling constraints see section 4.1.

3.5. A Strategy for the General Problem

Pricing out and updating coupling columns requires in general the use of k block inverses instead of one. They also tend to increase the size of the Working Basis and to complicate the updating of the factorized representation of the inverse. To amplify these points further::

a) From Lemma 2, if there are m_B coupling variables in the basis then $m_0 + m_B$ is an upper bound on the size of the Working Basis. Thus, the fewer coupling variables in the basis, the smaller is this upper bound estimate.

b) When the size of the Working Basis increases, so does the number of columns in the V matrix. Moreover, each Type B column (coupling column) can have non-zeros in all rows of V as compared to Type A2 columns which can have non-zeros only in rows of V belonging to their own block. Thus, the higher the number of coupling variables in the basis,

the higher the number of operations required to update a vector.

c) A comparison of Figures 1 and 2 shows that the work to update the factorized representation of the inverse can be expected to be more substantial when there are coupling variables in the basis.

Thus we conclude:

1) Pricing out and updating a coupling column to introduce it to the basis requires more work than for a block column, since the full Π vector is needed for pricing and hence all block inverses have to be used during the backward and forward transformations.

2) Having coupling columns in the basis increases the work per iteration even when pricing out only block variables by a), b) and c) above.

From these considerations evolves the general philosophy "do not touch the coupling columns until it becomes necessary". That is, reduce the general problem P to one with only coupling constraints by considering the coupling variables y fixed at some value, and use CCA to solve it. Only then relax the restrictions on y.

The hope is that most of the work can be done without using the coupling variables and that they will enter the game only at the end for relatively few iterations. This is probably the case in many large applications, where from knowledge of the problem it is possible to specify a value of y for which the

whole problem has a feasible solution, and then all of Phase 1 and most of Phase 2 can be done using CCA.

3.5-1 General Strategy

Repeatedly, in the General Algorithm that will be presented in 3.5-2, the values of y will be fixed as a parameter to reduce problem P to one with coupling constraints only. Whenever it is not possible to keep the values of y fixed any more we will use the following General Strategy, or GS for short:

Step 1: Relax restrictions on y. If some component of y was fixed at some feasible value different from its bounds, introduce it to the basis by increasing its value if this improves the value of the objective function, or by decreasing it otherwise, using the usual primal simplex criteria to determine the outgoing variable.

Step 2: Optimize the objective function using the PBP strategy to select the incoming variable (for this purpose consider coupling columns as a block k+1).

3.5-2 A General Algorithm

All the previous considerations lead to the following General Algorithm, whose flow-sheet is given in Figure 3:

Step 0: Fix the coupling variables at a value $y = y_0$. If a value of y is known for which the whole problem is feasible it can be used as y_0. Otherwise an arbitary value between its bounds can be taken.

Step 1: Minimize on each block the sum of infeasibilities. If

all problems get feasible go to Step 3. Else continue to Step 2.

Step 2: Use the General Strategy to minimize the sum of infeasibilities for blocks (common row constraints still relaxed). If no feasible solution is attained STOP, the whole problem is infeasible. Else fix y at its current value and continue to Step 3.

Step 3: Use the Coupling Constraints Algorithm to minimize the sum of infeasibilities in the common rows. If a feasible solution is attained go to Step 5. Else continue to Step 4.

Step 4: Use the General Strategy to minimize the sum of infeasibilities in the common rows. If no feasible solution is attained STOP, the whole problem is infeasible. Else fix y at its current value and continue to Step 5.

Step 5: Use the Coupling Constraints Algorithm to minimize the objective function. If an unbounded solution is encountered STOP, problem unbounded. Else continue to Step 6.

Step 6: Use the General Strategy to minimize the objective function. If an unbounded solution is encountered STOP, problem is unbounded. Otherwise the optimal solution is obtained.

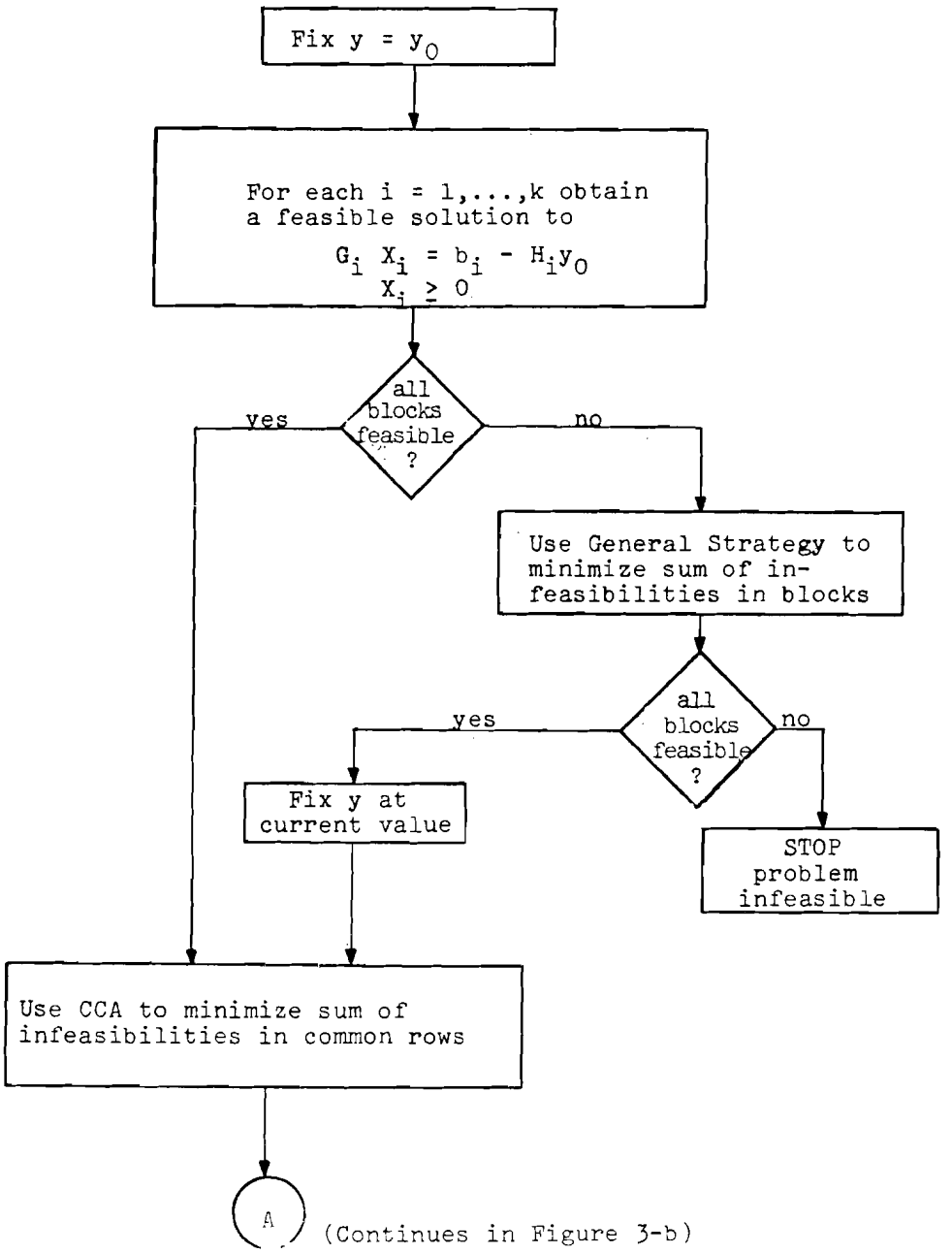
3.5-3 Observations

The General Algorithm has been stated in terms of a General Strategy and the Coupling Constraints Algorithm with the following ideas in mind:

1) The General Strategy as stated in 3.5-1 is one sensible strategy that takes advantage of the structure of the problem by using the partial block

FIGURE 3-a

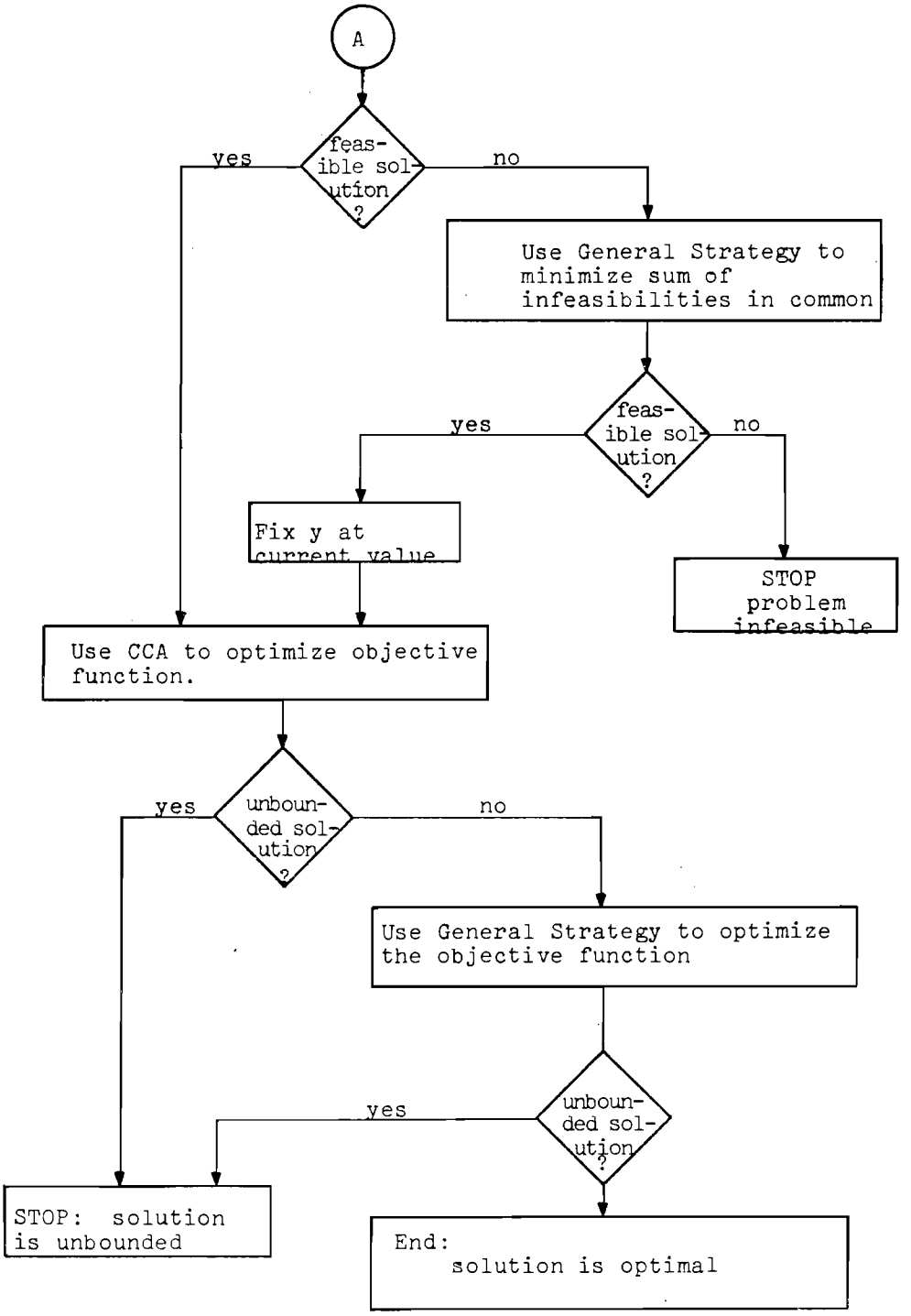
Information Flow-Sheet for General Algorithm



(Continues in Figure 3-b)

FIGURE 3-b

(Continuation of Flow-Sheet in Fig. 3-a)



pricing strategy. If it turns out that better results can be obtained by using a different strategy, for example pricing out coupling variables only when block variables do not supply a candidate, or every t cycles for t > 1, then it can be used as General Strategy to the benefit of the General Algorithm without any other modification.

2) Similarly, if an improved strategy is found for the block-angular linear case with coupling constraints, it can be adopted for the CCA to the advantage of the General Algorithm.

3) Notice that if a y is known for which the whole problem P is feasible, then the method reduces to the Coupling Constraints Algorithm followed by Step 6. Thus, when there are no coupling variables it reduces to CCA.

At the end of Chapter 4, after analyzing other partitioning and decomposition methods, further strategies are compared.

3.6. Representation for Inverse and V Matrix

So far only some product form representation of the inverse was assumed in describing the updating of the factorized representation of the inverse. This assumption was not necessary; it was used only because product form representations have been found to be the most efficient ones for general large scale linear problems, and this allowed us to speak of the updating of an inverse in terms of adding a column eta or a row eta to its representation, from which it is easy to obtain an intuitive

feeling of the relative effort required to maintain and to carry an inverse.

Thus for each basis we can use the representation that gives the best results for its inverse. For general sparse basis's and L-U factorization inversion with the use of the Forrest-Tomlin updating method for the triangular factors (see [14]) seem to give the most economic representation, and hence it seems to be the one to use for block inverses.

In certain cases though, some or all of the blocks may have special structure which we can take advantage of. For instance, if some block corresponds to a network, each basis will be a tree and we do not have to keep an inverse; it suffices to keep a set of pointers that allow us to reconstruct the tree [7]. That is, the GBBF approach not only gives us the advantages of a reduced BTRAN and FTRAN, but also allows us to take advantage of the special structure of each block.

The Working Basis can be expected in general to be more dense than the original matrix. Also its size may vary from one iteration to the next and both row and column elementary matrices may be required to update its inverse. Hence the Forrest-Tomlin method cannot be used. If the Working Basis is very dense, an explicit inverse (see [7]) will be best. Otherwise an L-U factorization followed by product form updates can be used. For the latter the following conditions will keep the density from getting too high:

1) A fair proportion of vectors in the Working Basis can be expected to consist of common row slacks and of

other columns in D_0 (see problem P).

2) Common rows will consist of rows with non-zeros in more than one block, but not necessarily in all. Columns belonging to a block having only zeros in a common row do not contribute in that row a non-zero to the Working Basis. The same is true if all columns of a block having non-zero in a given common row are non-basic, and even if some are basic, there is a certain probability that it may still be so.

3) From Theorem 1 for a minimal Working Basis $\hat{U}_A = 0$.

Besides these three points there are also some programming considerations that make it advantageous to use a product form representation for the Working Basis inverse: it is possible to use essentially the same INVERT, BTRAN and FTRAN subroutines for Working Basis and block inverses.

As for the V matrix, from (2.11)

$$\begin{pmatrix} B_w \\ \\ V \end{pmatrix} = B_N^{-1} B_{wO} \quad .$$

Hence it is not mandatory to store V, since in B_N^{-1} and B_{wO} we have all the information needed to carry out the computations involving V. In BTRAN V is not needed anyhow when all the variables not in the Working Basis are feasible, and so in this case it would make no difference whether we have stored V or not. In FTRAN on the contrary all the block inverses would be required

in the last part of it which involves V. This is only justified in case the number of non-zeros in all block inverses is less than that in V. This would probably mean that the size of the Working Basis is large, since the number of columns in V equals that in the Working Basis, and the advantages of using the GBBF approach are reduced with respect to a general method. Hence for problems where it is advantageous to use the GBBF approach, i.e. those leading to a Working Basis with a relatively small size, it is better to store V (i.e. store the non-zeros of V).

Recall that in updating an incoming column d we calculate first

$$\hat{d} = \begin{pmatrix} \hat{d}_w \\ \\ \hat{d}_s \end{pmatrix} = B_N^{-1} d$$

where $\hat{d}_w$ is the restriction of $\hat{d}$ to rows in the Working Basis. Since this information is already available it seems convenient to store the whole updated vector $\hat{d}$. This way we also have available the vectors forming the Working Basis and it is not necessary to recompute them every time we want to invert it. All that is required is a flag that tells us which rows are in the WB and which ones are not.

In cases where storage restrictions do not allow storing V, in addition to a higher computational effort in FTRAN it becomes necessary to generate the row of V which correspond to the

pivot row and which is required in the updating procedure. As can be readily seen from (2.11) this is equivalent to BTRAN using the block inverse of the outgoing variable plus pricing out the columns in the Working Basis with the π^r thus calculated.

3.7. Other Considerations

Up to now we have implicitly assumed the use of the most negative reduced cost as the criterion for selecting the column to enter the basis among those that were priced out. Of course other criteria and techniques widely used in the primal Simplex Method are also possible here; so for instance multiple pricing [30], i.e. where at each pricing operation the k columns having the most negative reduced cost among those priced out are selected as candidates and updated, and are then used in a sub-optimization where the greatest change rule is applied to determine the column to enter the basis.

There are also other criteria which do not look good in general, but look promising when part of a GBBF approach. These will be examined in more detail in Chapter 4, especially in Section 4.1-5.

CHAPTER 4

A UNIFYING APPROACH TO PARTITIONING AND DECOMPOSITION METHODS

General Block-Angular Basis Factorization allows us to unify existing Partitioning and Decomposition methods (not based on the Dantzig-Wolfe decomposition principle) for solving block-angular linear problems. In essence all of them can be viewed as the Simplex Method using the GBBF representation for the inverse, and differing on the strategy as to the vector pair selected to enter and to leave the basis.

For each method we will refer to the appropriate place in the literature for its detailed description, and state it here only in terms of the pivot strategy it uses. When necessary we will expand somewhat on alternative ways of implementing them. We will first consider block-angular linear problems with coupling constraints, then those with coupling variables and finally those with both coupling constraints and variables.

4.1. Block-Angular Linear Problems with Coupling Constraints

As was discussed in 3.4-1, in this case the Working Basis is always of constant size $m_0 \times m_0$, and the updating procedure for the factorized representation of the inverse

simplifies to that in Figure 2 because of the absence of Type B variables. Many "Partitioning" or "Decomposition" methods have been proposed over the years for this class of problem. We will now analyze them to identify their strategy in the Simplex Method using GBBF (but not necessarily in the order in which they were first presented). At the end of the section we look at some new strategies which can be implemented efficiently when using the GBBF approach in the Simplex Method.

4.1-1 Primal Simplex Strategy: Methods of Kaul [22], Bennett [4] and Müller-Mehrbach [27].

All three authors proposed their methods independently about the same time. Kaul's method is better known as Generalized GUB (short for Generalized Generalized Upper Bounding [22]), and Müller-Mehrbach's as the Method of Direct Decomposition [27]. There are slight variations in their methods, but all correspond to the GBBF Simplex Method using the usual primal strategy of introducing into the basis the column with the most negative reduced cost, and hence leading "to the same solution path as the Simplex Method" [22], [27].

4.1-2 Rosen's Primal Partitioning Method [32]

We can distinguish three steps in the overall strategy of Rosen's method:

Step 1: Solve all block problems to optimality. If some block is infeasible STOP, the whole problem is infeasible. Else go to Step 2.

Step 2: Relax the non-negativity constraints on variables in the block basis's. Solve the relaxed problem. If the solution is infeasible STOP, the problem is infeasible. Else go to Step 3.

Step 3: Check whether the relaxed non-negativity constraints are satisfied. If so STOP, the solution is optimal. Else rearrange variables between blocks and Working Basis as shown in Rosen [32] for at least one block having a variable not satisfying the non-negativity constraints. This way at least one infeasible variable is exchanged to the Working Basis. Return to Step 2.

Observations:

1) The validity of the above strategy was proven by Rosen in [32]. It follows also from the relaxation strategy in Geoffrion [16] (see also Lasdon [24]), of which this is a special case.

2) Notice that the relaxed problem in Step 2 could be unbounded even though the whole problem is not. To avoid this a bounding row making the sum of all variables less or equal to a very large number is added to the common rows. If this constraint is binding in an optimal solution the whole problem is declared to be unbounded.

3) Notice that at the end of Step 2 the solution is dual feasible for the whole problem, and after returning from Step 3 at least one previously violated non-negativity constraint is enforced. Thus the solutions in Step 2 form a non-decreasing sequence of lower bounds to the problem (assuming we are minimizing).

4) Rosen [32] suggests using a primal strategy in Step 2. In this case we can take advantage of the savings in BTRAN with the PBP strategy. A primal approach also allows us to handle the situation when some block problem is unbounded without adding a bounding constraint to it. Other authors have suggested using a dual approach (see Grigoriadis [17]), in which case PBP could not be used, and a bounding row would have to be added to any unbounded problem.

5) Notice that all the pivoting in Step 2 occurs in the common rows, i.e. the outgoing column always belongs to the WB, leading to the easiest update situation (see Fig. 2). This is a consequence of relaxing the non-negativity constraints on variables in the block basis's. It also implies that it is not necessary to completely update the incoming column on these rows. That is, the V matrix is not required in the forward transformation.

6) The dual form of Rosen's method is known as Gass' Dualplex Method (see 4.2.2).

4.1-3 Primal-Dual Strategy: Balas [1], Knowles [23]

In this approach all block problems are first optimized. This solution is then dual feasible for the whole problem, and primal feasible except possibly in the common rows. The primal-dual strategy is then employed to reduce the sum of

infeasibilities in the common rows, maintaining a dual feasible solution for the whole problem.

This strategy is intuitively appealing. Its computational advantages will depend on how many non-basic columns there are on each restricted primal, since if there is only one, the effort per iteration is essentially the same as in a dual method. On the other hand, if there are many, most of the work can be expected to be on the primal iterations of the restricted problem, and in this case we can take advantage of the savings in BTRAN with PBP.

Balas first presented this algorithm as "An Infeasibility Pricing Decomposition Method for Linear Programs (Version A)" [1]. Knowles [23] later wrote a FORTRAN code for a version of the Algorithm. He obtained encouraging preliminary experimental results.

4.1-4 Dual and Parametric Strategies: Ohse [28], Orchard-Hays [29]

Again all blocks are optimized first in order to obtain a dual feasible solution to the whole problem which is primal infeasible only in the common rows. A constraint bounding the sum of the variables to be less or equal to a very large number has to be added in case of unbounded subproblems.
Ohse's Dual Method [28] then uses the usual dual strategy (see [7]), taking advantage of the reduction in computations when using the factorized representation of the inverse whenever the outgoing variable is basic in some block and $v_r = 0$ (see Lemma 3).

Any dual partial pricing scheme to select the outgoing variable can be used. This allows some leverage as to what variable to select to leave the basic set. Recall that the computational effort to update the factorized representation of the inverse depends exclusively on the position of the outgoing variable (see 3.4-1), and hence in this method we can exert some influence to fall into the easier cases.

Orchard-Hays Block-Product Algorithm [29] is a parametric method. It modifies the right hand side of the common rows to make the dual feasible solution obtained after solving all block problems also primal feasible. Then the right hand side of the common rows is varied parametrically to its original value.

Besides adding bounding constraints to handle unbounded blocks, Orchard-Hays also shows how to get a dual feasible solution when having non-unit vectors in D_0 (see [29]). This same approach can be applied to the Primal-Dual and Dual methods when necessary.

4.1-5 Other Strategies

Among many other possible strategies we would like to mention two variations on primal strategies.

The Partial Block Pricing (PBP) Primal Strategy: referred to already in section 3.4-2 and used in the Coupling Constraints Algorithm which has given encouraging results on some test problems (see Appendix A). All block problems are first optimized (or made feasible). Then the PBP primal Strategy is used to take advantage of the savings in the backward

transformation that it makes possible when using the GBBF method.

PBP with a Ratio Pricing Criteria: Ratio pricing was proposed originally by Markowitz (see Dantzig [7]). It requires taking for each non-basic column a ratio of its updated coefficients in the objective row and in the infeasibility row. Thus in the usual product form methods two backward transformations, one to obtain the prices on the objective row, the other those on the infeasibility row, and two pricing operations, one to update the coefficients in the objective row, the other those in the infeasibility row, are necessary. This makes the computational requirements excessive.

In this context a somewhat modified ratio pricing rule is proposed, which can be implemented efficiently on block-angular problems when the GBBF approach is used.

Suppose that in Phase 1, for column j, d_j is the reduced cost for the minimization of the sum of infeasibilities, and c_j for the objective function. Then define

$$SD = \{j : d_j < -\delta\} , \tag{4.1}$$

i.e. the set of columns that would decrease the sum of infeasibilities if introduced to the basis (δ is the 0 tolerance for the reduced cost in the computer).

The ratio pricing criterion suggested by Markowitz is:

$$\text{choose} \quad \frac{c_q}{d_q} = \min \{\frac{c_j}{d_j} : j \in SD\} , \tag{4.2}$$

i.e. choose that column to enter the basis which gives the maximum improvement in the objective function per unit decrease in the sum of infeasibilities.

When solving block-angular linear problems with coupling constraints, after optimizing all block problems, we then have a solution which is dual feasible (unless some block gives an unbounded solution) for the overall problem and primal infeasible only in the common rows. Thus the objective function will have a value below the optimum (minimizing case). Hence we modify the ratio pricing criterion. Herewith define

$$SM = \{j : c_j < -\delta \ , \ j \in SD\} \tag{4.3}$$

$$SO = \{j : |c_j| \leq \delta \ , \ j \in SD\} \tag{4.4}$$

$$SP = \{j : c_j > \delta \ , \ j \in SD\} \ . \tag{4.5}$$

Then in Phase 1, if SD is empty the problem is infeasible. Otherwise use the :

Ratio Pricing Criterion

Rule 1: If SM is empty use rule 2. Otherwise select the incoming variable so that

$$\frac{d_q}{c_q} = \max \{\frac{d_j}{c_j} : j \in SM\} \ , \tag{4.6}$$

i.e. maximize the reduction in the sum of infeasibilities per unit improvement in the objective function.

Rule 2: If SO is empty use rule 3. Otherwise select the incoming variable so that

$$d_q = \min \{d_j : j \in SO\} \quad . \tag{4.7}$$

Rule 3: Select the incoming variable so that

$$\frac{c_q}{d_q} = \max \{\frac{c_j}{d_j} : j \in SP\} \quad . \tag{4.8}$$

That is, if possible we select, using rule 1, the column that gives the largest decrease in the sum of infeasibilities per unit improvement in the objective function. If there is no column which improves both objectives then by rule 2 we select the one that gives the greatest decrease in the sum of infeasibilities without affecting the objective function value. If this set is empty we fall back on rule 3 and select to enter the basis the column that gives the minimum increase in the objective function (minimizing case) per unit decrease in the sum of infeasibilities.

Observe that these pricing criteria have many points in common with the primal-dual strategy, but they do not require keeping a dual feasible solution and hence allow the use of partial pricing.

Herewith we change SD to

$$SD_i = \{j : d_j < -\delta \quad , \quad \text{column } j \text{ in block } i\} \tag{4.9}$$

in (4.3) through (4.5). Then if we are in Phase 1 and SD_i is not empty we use the ratio pricing criterion. If SD_i is empty

we proceed to another block. If it is empty for all blocks i then the problem is infeasible. Now we can state the :

PBP Algorithm using the Ratio Pricing Criterion

Step 1: Optimize each block problem (or make feasible). If some problem is infeasible STOP, the overall problem is infeasible.

Step 2: Minimize the sum of infeasibilities in the common rows using the partial block pricing strategy with the above ratio pricing criterion to select the incoming column among those priced. If the minimum is not 0 STOP, the problem is infeasible.

Step 3: Use the Coupling Constraints Algorithm to obtain the optimum.

The difference from the Coupling Constraints Algorithm lies in Step 2, which can be implemented efficiently using the GBBF method. Hopefully when achieving feasibility in the common rows at the end of Step 2 the problem will be optimal or near optimal, requiring only a few Phase 2 iterations in Step 3.

Implementation

Let $\Pi = (\Pi_0, \Pi_1, \ldots, \Pi_k)$ be the dual prices associated with the objective function for the current basis, and let $\sigma = (\sigma_0, \sigma_1, \ldots, \sigma_k)$ be the dual prices associated with the sum of infeasibilities for the current basis. Then when pricing out a non-basic column in block i, we need to compute

$$d_j = \sigma_0 D_j^i + \sigma_i G_j^i \tag{4.10}$$

and

$$c_j = \Pi_0 D_j^i + \Pi_i G_j^i \tag{4.11}$$

where D_j^i is the restriction of column j to common rows and G_j^i its restriction to rows in its own block i. Thus we only have to calculate σ_0, σ_i, Π_0 and Π_i and hence the number of operations in the two backward transformations will be proportional to two times the number of non-zeros in the representation of the Working Basis and in that of the inverse of block i. If the number of non-zeros in the representation of the Working Basis is less than in any one of the block inverses, then probably for problems with 3 or more blocks this would still be less than the number of operations in one backward transformation using a general representation for the problem inverse.

The pricing can also be done at only a small additional cost. For each non-basic column in block i, first compute d_j by (4.10). If $d_j > -\delta$ then $j \notin SD_i$ and it is not necessary to compute c_j. Otherwise c_j is computed and we see which of the three rules applies.

Thus the pricing effort increases by the percentage of columns that have a negative reduced cost for the sum of infeasibilities. Especially in the later stages these can be expected to constitute a small fraction of the total number of non-basic columns.

4.1-6 Some Comments

Extensive experimental tests and comparisons in a systematic way are necessary to determine which of the above strategies will perform better for block-angular linear problems

with coupling constraints when using the GBBF Simplex Method. Up to now there are only a few experimental results (see [23], [38]) and no comparisons available, which stresses the need for systems optimization laboratories (see Dantzig [6]) to remedy this situation .

From practical considerations primal strategies have the advantage (besides the savings in BTRAN when using GBBF) that they can be used in conjunction with almost all combinations of pricing and pivoting criteria used in current production codes; so for instance besides pricing out at each pricing operation only a subset of the non-basic variables (partial pricing), several candidates may be selected and updated (multiple pricing) using greatest change criteria in a sub-optimization involving only these candidates to determine which ones are to be introduced to the basis and in what order.

4.2. Block-Angular Linear Problems with Coupling Variables

This is the special case of the general block-angular problem P when there are no coupling constraints. It is the dual problem to the block-angular problem with coupling constraints analysed in 4.1. Of course one solution strategy could be to dualize and then use any one of the methods in 4.1. It is however of interest to have methods that solve it directly, since this structure arises often in applications involving uncertainty (see [25]).

Also for block-angular linear problems with coupling variables there are some methods that have been around for a

long time and for which no experimental results are available.

4.2-1 Beale's Pseudobasic Variables Method [3]

Beale's method uses a primal strategy. The central idea is to treat the coupling variables as parameters in order to preserve a square block-angular structure for the basis. After a closer look it becomes apparent that they are only treated as parameters in the first part of the algorithm, where they are fixed at a value for which the system is assumed to have a solution. Later on they are really treated as basic variables; however, the algebra of the Simplex Method has been modified so that the values of the incoming variable and of the other basic variables are expressed as a function of one of the coupling variables which is then allowed to change value until one of the basic variables leaves the basic set. The values of the "linking parameters" are modified in this process exactly as all the other basic variables. Using GBBF Beale's method is equivalent to the specialization of the General Algorithm to the case when there are no coupling constraints.

4.2-2 Gass' Dualplex Method [15]

As has been pointed out before, this method can be viewed as the dual of Rosen's algorithm (see 4.1-2). It assumes that for a given $Y = Y_0$ fixed all block problems have a feasible solution. Its strategy expressed in terms of the GBBF Simplex

Method is:

Step 1: Set k = 0. Optimize each block problem for $Y = Y_0$. If any problem is unbounded STOP, the whole problem is unbounded. Otherwise go to Step 2.

Step 2: Set k = k + 1. Solve a restricted problem where only coupling variables are allowed to enter the basic set. If an unbounded solution is encountered STOP, the problem is unbounded. Otherwise continue to Step 3.

Step 3: Let Y_k and Π_k be the values for the Y variables and for the dual prices in the optimal solution at the end of Step 2. Fix Y_k as a parameter. For each block use Π^k to price out non-basic block variables. If all price out optimal STOP, the current solution is optimal. Otherwise exchange as many pseudo-basic variables as possible with non-basic variables that price out non-optimal. (Observe that when fixing Y^k as a parameter B_N becomes the true basis, with all previously pseudobasic variables at value 0. Since we pivot only in rows of pseudo-basic variables the value of the solution does not change, and hence at the end the solution from Step 2 is still feasible, but not basic any more.) Return to Step 2.

4.2-3 Other Strategies

Both Beale and Gass assume the knowledge of an initial $Y = Y_0$ for which each block problem has a feasible solution. If this is not the case the problem can be set up in the usual way, a Phase 1 procedure to minimize the sum of infeasibilities. If this minimum is greater than zero the problem is infeasible.

Otherwise the value of Y in the first feasible solution is used as Y_0 to start either method in Phase 2. With this added feature Beale's method is equivalent to the specialization of the General Algorithm in section 3.5-2 for the case when there are no coupling constraints.

Alternatively the PBP strategy with the ratio pricing criterion to select the incoming variable could be used in Phase 1.

4.3. Block-Angular Linear Problems with Coupling Constraints and Variables

Only a few algorithms have been proposed in the literature for the general problem P. They usually were worked out as extensions of algorithms for the block-angular linear case with coupling constraints, as for instance the Generalized GUB method (see 4.1-1) and Rosen's algorithm (see 4.1-2). Extensions of the dual and primal dual strategies have not been presented, probably because of the requirement of having a dual feasible solution at hand to start the procedures.

4.3-1 Primal Strategy: Hartman and Lasdon's Method [20]

Hartman and Lasdon use the usual primal simplex strategy. They develop a basis factorization scheme in which a column corresponding to a block variable that becomes pseudobasic is dropped from the block basis together with the row in which it was basic, thus reducing the size of the block basis. Hence all basis's may vary in size. Besides they do not require the Working

Basis to be minimal. To avoid it increasing too much, checks are made which require doing some computation.

They present some computational results of their method for a special class of production and inventory problems. For a given problem, the smaller the size of the Working Basis, the faster the iterations. In particular, overall solution time is smallest if coupling columns are not introduced into the basis in Phase 1 (unless necessary), to keep the Working Basis small. These observations agree with our analysis in 3.5.

4.3-2 Ritter's Method [31]

Ritter's method amounts to a generalization of Rosen's method to problems having also coupling variables. It uses the same relaxation strategy as Rosen's (see also Geoffrion [16] and 4.1-2) with a slightly different criterion as to which variables to relax and which violated relaxed variables to enforce to account for the presence of the coupling variables. All comments on Rosen's method (see 4.1-2) apply also here (with some slight modifications in some cases). We state his method under the assumption that a Y_0 is known for which all blocks have a feasible solution.

Step 1: For $Y = Y_0$ fixed, optimize all blocks.

Step 2: Relax non-negativity constraints on variables corresponding to the block basis's. Solve the relaxed problem (no restrictions on coupling variables). If the relaxed problem has no solution STOP, the problem is infeasible.

Step 3: Check whether the relaxed non-negativity constraints are

satisfied. If so STOP, the solution is optimal. Otherwise rearrange variables between blocks and Working Basis as in Rosen's algorithm for at least one block having a variable not satisfying the non-negativity constraints. Whenever according to this rule a coupling variable associated with the Working Basis has to be switched with a block basic variable in its own block, the size of the Working Basis is instead increased by adding the block basic-column and its pivot row to it. This way at least one column associated with an infeasible variable is introduced to the Working Basis. Return to Step 2.

Observe that when using the GBBF approach, besides Step 1, also Step 2 is the same as Rosen's and Ritter's algorithms, i.e. relax non-negativity constraints on basic variables not in the Working Basis.

Only the rearrangement procedure in Step 3 is more general in Ritter's method to account for the coupling variables. Thus the five observations which we presented earlier in 4.1-2 after Rosen's method apply here.

4.3-3 Other Strategies

The General Algorithm presented in 3.5 is another example of a primal strategy, more refined than the usual simplex strategy to get the most out of the structure of the problem. The ratio-pricing technique described in 4.1-5 is another possibility that looks promising and can be extended directly to problems with coupling variables and constraints because it does not require dual feasibility. It was not incorporated into

the General Algorithm because it is still untested, as compared to the partial block pricing strategy with the most negative reduced gradient criterion used in CCA. If tests later show that partial block pricing with the ratio pricing criterion is more efficient, it should be incorporated into the General Algorithm.

Of the strategies for block-angular linear problems with coupling constraints that require dual feasibility, the easiest to extend is Orchard-Hays' parametric strategy (see 4.1-4). Recall that in this approach, after solving all block problems, the right hand side in the common rows is changed to force the current solution to be both dual and primal feasible. For problems with coupling constraints and variables the same thing can be done also to the cost coefficients in the coupling constraints to force them dual feasible after all block problems have been optimized. After this, both the modified cost coefficients of the coupling variables and the modified right hand side of the common rows are forced back to their old values using a parametric technique.

4.4. Specializations of the General Algorithm

To end this chapter we want to mention the specializations of the General Algorithm for some special cases.

For block-angular linear problems with coupling constraints it reduces to the CCA method described in 3.4. Furthermore, if each block corresponds to a GUB set, then the CCA method in turn reduces to the GUB algorithm (see [12]), in which for each GUB set we initially select as key variable the one making the GUB

set feasible and giving the best value for the objective function.

For block-angular problems with coupling variables the General Algorithm specializes to the same strategy as Beale's Pseudobasic Variables Method (see 4.2-1).

CHAPTER 5

NESTED FACTORIZATION

5.1. General

Imbedded in block-angular structures are blocks which themselves are of block-angular form etc. recursively. Thus the application of GBBF in the Simplex Method could lead to many levels of representation of inverses in factorized form. In the following the special case of block-angular structures with coupling constraints will be considered to show how the basis factorization approach developed so far lends itself naturally to nested applications. Later these results will be applied to the solution of staircase problems.

5.2. Notation and Concepts

Recall

$$\text{from (2.7)} \quad B_T = B_N \hat{B}_W \hat{V} \quad ,$$

$$\text{from (1.3)} \quad B_N = \prod_{i=1}^{k} \hat{B}_i \quad ,$$

and from (1.2)

$$\hat{B}_j = \begin{bmatrix} I_0 & 0 & \cdots & 0 & A_j & 0 & \cdots & 0 \\ & I_1 & & & & & & \\ & & \ddots & & & & & \\ & & & B_j & & & & \\ & & & & \ddots & & & \\ & & & & & I_k & & \end{bmatrix}$$

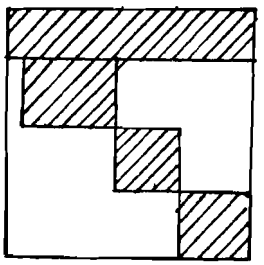
We can write for (1.2)

$$\hat{B}_j = \begin{bmatrix} I_0 & & & \\ & I_1 & & \\ & & \ddots & \\ & & & B_j \\ & & & & \ddots \\ & & & & & I_k \end{bmatrix} x \begin{bmatrix} I_0 & & & & A_j \\ & I_1 & & & \\ & & \ddots & & \\ & & & I_j & \\ & & & & \ddots \\ & & & & & I_k \end{bmatrix} \qquad (5.1)$$

and hence

$$\hat{B}_j^{-1} = \begin{bmatrix} I_0 & & & & -A_j \\ & I_1 & & & \\ & & \ddots & & \\ & & & I_j & \\ & & & & \ddots \\ & & & & & I_k \end{bmatrix} x \begin{bmatrix} I_0 & & & \\ & I_1 & & \\ & & \ddots & \\ & & & B_j^{-1} \\ & & & & \ddots \\ & & & & & I_k \end{bmatrix} \qquad (5.2)$$

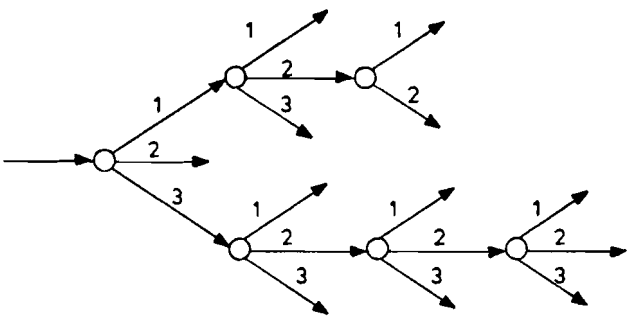
In the nested factorizations that will be considered here, at least for one j B_j has the structure of a block-angular problem with coupling constraints, i.e. of the type:



(5.3)

and we can represent its inverse in factorized form. Again some of the blocks in B_j could have the block-angular structure (5.3) and so on, and hence we could have many levels of factorization.

The basis factorization developed in Chapter 2 will be referred to as a 1 level factorization (or the level 1 factoization), and accordingly its Working Basis will be called the level 1 Working Basis and its block basis's the level 1 block basis's. If one or more of the block basis's are factorized again, then each one of them gives rise to a level 2 Working Basis and to level 2 block basis's. To simplify this process we would like to represent it in the following by a tree-like diagram (which will be called the "associated tree"):

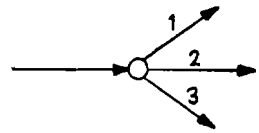


(5.4)

Here a directed arc represents a basis. If an arc does not end in a node it means that a general representation is used for the inverse of the basis it represents. Otherwise a factorized representation is used and we associate the Working Basis and V matrix of the factorization with the node, and each one of the block basis's with an outgoing arc.

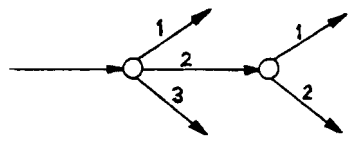
For example, for a block-angular basis B_T with three blocks

would be the associated tree if factorization is not used;



if we factorize.

Further, if block 2 has also a block-angular structure with two blocks, then



would be the associated tree of this 2 level factorization.

Notice that the level of a basis corresponds to the number of nodes in the path starting from the origin that leads to it.

As with the basis, we can use the associated tree to represent the classification of the columns of the problem. That is, all columns in a matrix correspond to an arc. If it does not lead to a node the columns are not subdivided further. Otherwise we associate with each outgoing arc the sub-matrix

of columns belonging to the block whose basis it represents. For simplicity we assume that at any level of the factorization all columns in a block-angular matrix belong to some block (i.e. $D_0 = 0$ (see 2.1)). This way each path corresponds to a subset of the problem columns having a particular structure.

5.3. A Nested Updating Procedure

Let IC be the incoming column,

OC the outgoing column, and

EC the column in the Working Basis that is exchanged with the OC in some block (whenever the case arises).

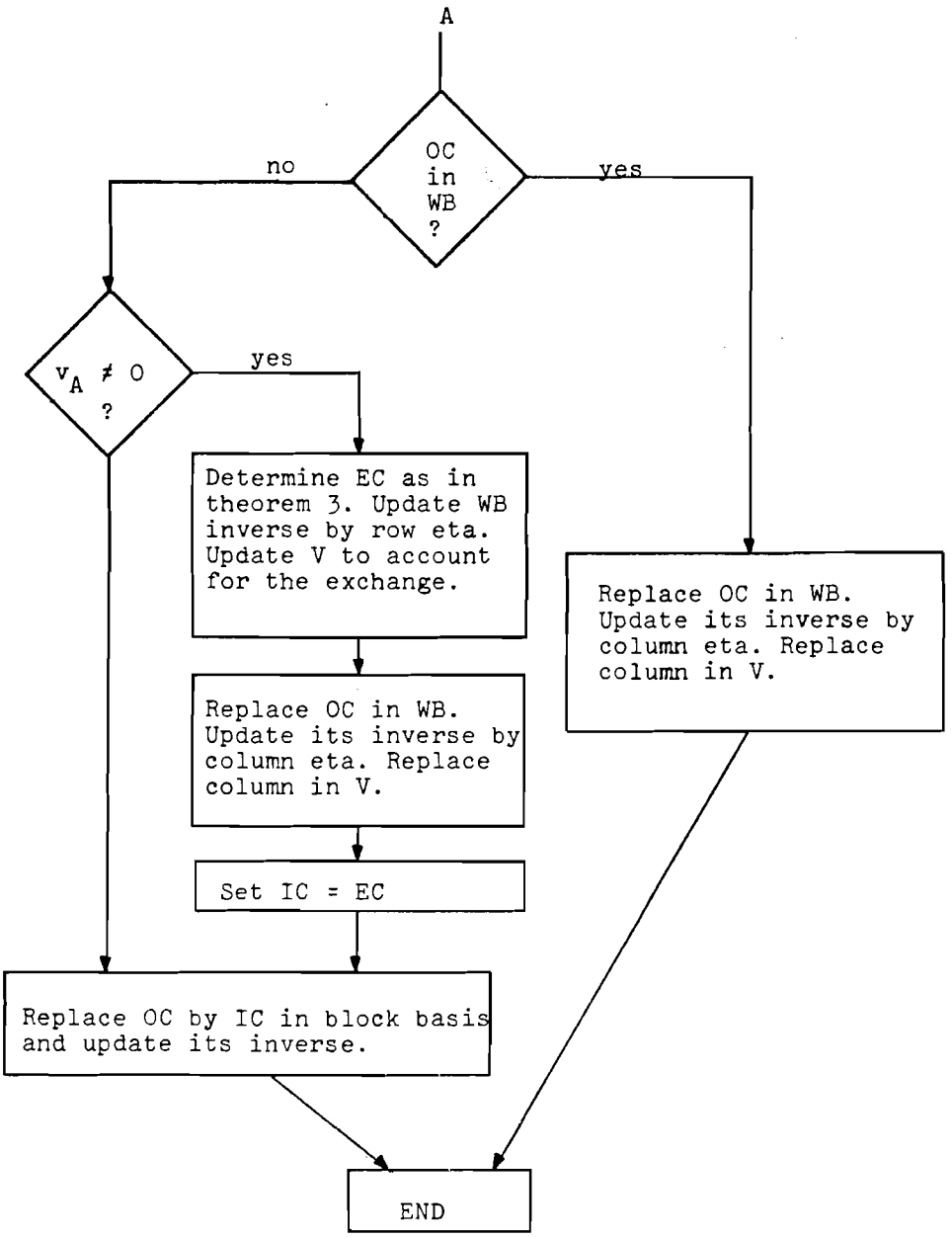
It will be convenient to modify the Information Flow-Sheet of the Updating Procedure for Block-Angular Linear Problems with Coupling Constraints (see Figure 2), so that the updating of the inverse of a block basis is done as a last step (when the case arises). This modified Information Flow-Sheet is shown in Figure 4.

Observe that when the OC is in the Working Basis only its inverse is updated and a column is replaced in the V matrix, independently of the representation used for the block inverses. In the other cases, due to the replacement of one column by another in some block basis, the representation of its inverse has to be updated as a last step using the appropriate updating procedure. If a factorized representation is used for it, then we can use again the scheme in Figure 4.

In general then, whenever a block basis has to be up-

FIGURE 4

Modified Information Flow-Sheet of Updating Procedure for Block-Angular Linear Problems with Coupling Constraints



dated, a check is made to determine whether a factorized representation is used for its inverse. If not we proceed as before. Otherwise first the OC is replaced by the IC in the A_j matrix (see (5.2)), and then the procedure in Figure 4 is used to update its factorized representation. The resulting nested updating procedure is shown in Figure 5. For simplicity indices have been omitted except to indicate the level of factorization, since the position of the OC uniquely determines the path that is taken.

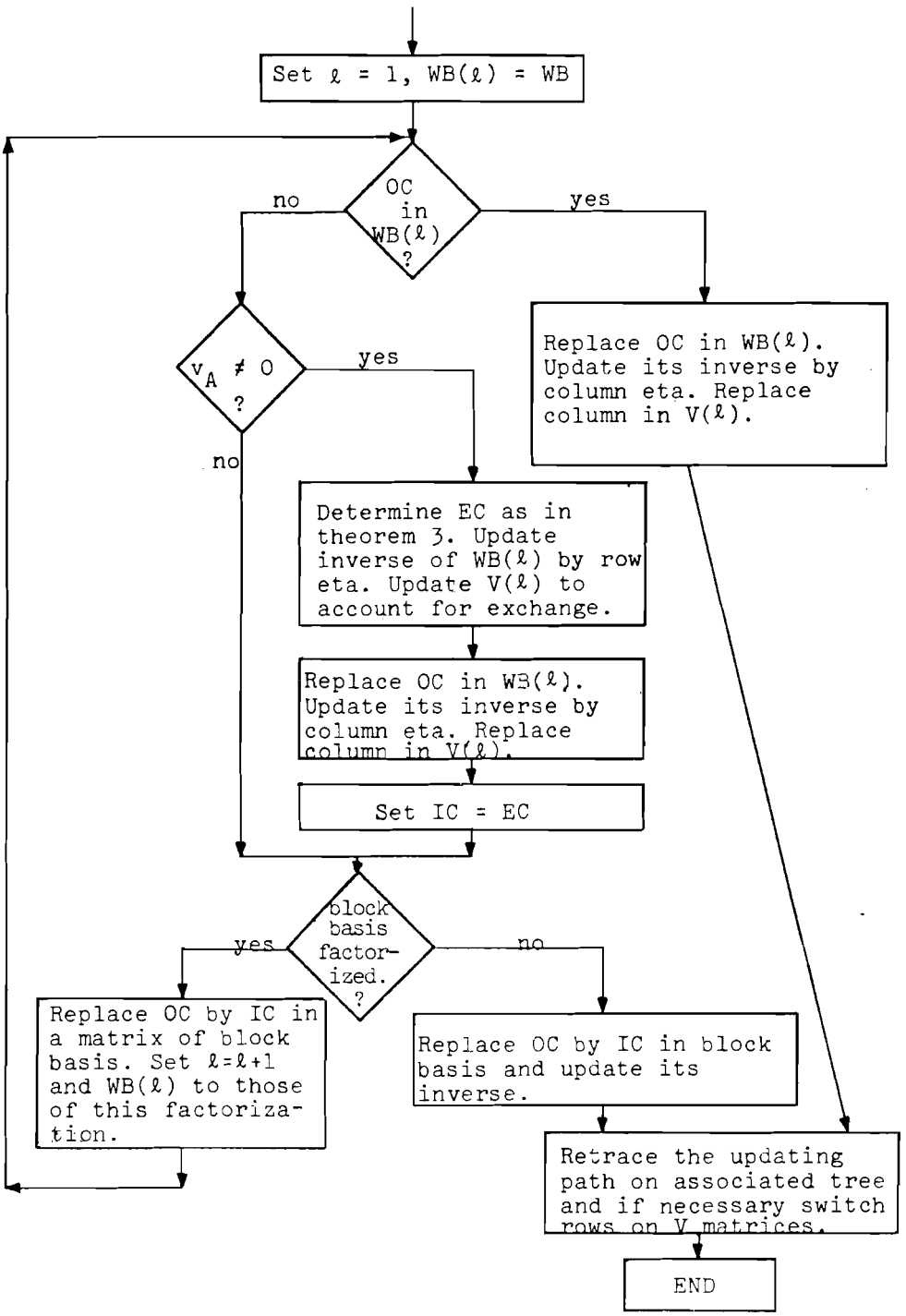
Recall from section 2.3 that

$$\begin{pmatrix} B_W \\ V \end{pmatrix} = B_N^{-1} B_{WO} \quad .$$

Thus, knowing the pivot row and having the representation of the IC in terms of its block basis we can update the V matrix before updating the block inverse. In the nested factorization case, however, it may turn out that in order to update the latter a pair of columns has to be exchanged between its Working Basis and one of its block basis's. This permutation requires the switching of the rows of V corresponding to the pivot rows of the exchanged columns. This is included as the last step in Figure 5. All it requires is storing the information about the pivot rows of columns that have been exchanged (at most a pair for each level of factorization), and the actual switching of the rows (or the switching of the row indices of its non-zero coefficients) in the affected V matrices can be postponed until they are needed for the first time for a backward

FIGURE 5

Nested Updating Procedure



or a forward transformation. Some properties of the nested updating procedure are summarized in the following.

Proposition 2: If L_{max} is the maximum level of factorization, then to update the factorized representation of the inverse after the replacement of one column by another in the basis, at most $L_{max} + 1$ of the WB's and block inverses and L_{max} of the V matrices have to be updated. Moreover,

a) if the outgoing variable is in a level k WB, at most k of the WB inverses and k - 1 of the V matrices have to be updated, and

b) if the outgoing variable is in a level k block basis that is not factorized further, then at most k + 1 of the WB and block inverses and k of the V matrices have to be updated.

Proof: Suppose the OC is on a level k basis. Then for levels $\ell = 1,2,\ldots,k-1$ we cycle on the loop in Figure 5 and each time we have to update at most one WB inverse and one V matrix (if $v_A \neq 0$), i.e. k - 1 in all. For $\ell = k$ if the OC is in the WB we update its inverse and finish (except possible for switching rows on V matrices already modified) which shows a). If the OC is in a block basis that is not factorized further, then it may be necessary to update the k-th level WB and V matrix beside the block inverse of the OC, and hence b). The first part follows from b) with $k = L_{max}$. ||

5.4. Nested Factorization in the Simplex Method

As was discussed earlier (see section 3) a representa-

tion of the inverse is needed in a revised Simplex Method to perform two kinds of calculations: the backward and the forward transformations.

5.4-1 Backward Transformation

In section 3.1 the backward transformation for the 1 level factorization

$$(3.3) \qquad \Pi = C\,\hat{V}^{-1}\,\hat{B}_W^{-1}\,B_N^{-1}$$

is considered as consisting of three steps:

Step 1: Calculate $\hat{C} = C\hat{V}^{-1}$

Step 2: Calculate $\hat{\Pi} = \hat{C}\hat{B}_W^{-1}$

Step 3: Calculate $\Pi = \hat{\Pi}B_N^{-1}$.

In particular, when all level 1 blocks are feasible, Step 1 is not required. For Step 3 we had

$$\Pi_0 = \hat{\Pi}_0$$

and

$$\Pi_i = (\hat{\Pi}_i - \Pi_0 A_i)B_i^{-1} \quad , i = 1,\ldots,k \quad \text{(relation (3.5))}$$

If a factorized representation is used for B_i, i.e. $B_i^{-1} = \hat{V}_i^{-1}\,\hat{B}_{W_i}^{-1}\,B_{N_i}^{-1}$, then by letting

$$C_i = (\hat{\Pi}_i - \Pi_0 A_i) \qquad (5.5)$$

we obtain

$$\Pi_i = C_i B_i^{-1} = C_i \hat{V}_i^{-1} \hat{B}_{W_i}^{-1} B_{N_i}^{-1} \qquad (5.6)$$

i.e. the same relationship as in (3.3). Thus it is possible to use the above three steps for the calculation of Π_i. Notice though by (5.5) that now all components of C_i may be non-zero and Step 1 has to be performed no matter whether in Phase 1 or in Phase 2. Otherwise everything is as before and the same approach can be extended to higher levels of factorization.

Proposition 3: If k is the number of arcs in a path of the associated tree, then to calculate the components of the price vector Π needed to price out the columns corresponding to that path, only the k inverses and k - 1 V matrices associated to it are required in the backward transformation.

Proof: By induction. It is true for k = 1 and k = 2, i.e. no factorization and level 1 factorization. Assume it is true for $k = 1,\ldots,\ell$, $\ell \geq 2$. Let a subindex 0 denote the common rows and a subindex i the rows in block i for the level 1 factorization. Partition a column d according to this into

$$d = \begin{pmatrix} d_0 \\ d_1 \\ \vdots \\ d_i \\ \vdots \\ d_{\bar{k}} \end{pmatrix} \quad . \quad \text{Then if d belongs to block i } d_i = 0 \text{ for } j = 1,\ldots,\bar{k} \text{ , } j \neq i$$

and hence $\Pi^t d = \Pi_0 d_0 + \Pi_i d_i$

Thus to compute $\Pi^t d$ for columns associated with an arc of length $k = \ell + 1$ we have to compute $\Pi_0 d_0$ and $\Pi_i d_i$. To compute $\Pi_0 d_0$, Π_0 is required, for whose calculation only the level 1 WB inverse is needed. To compute $\Pi_i d_i$ is equivalent to pricing out the columns associated with a path of length $k - 1 = \ell$ in the associated tree of the block basis B_i, and hence by the inductive assumption only ℓ inverses and $\ell - 1$ V matrices are needed to calculate the components of the Π_i vector required for it. Thus it is also true for $k = \ell + 1$. ||

5.4-2 Forward Transformation

As discussed in section 3.2 the forward transformation for a column d from block i can be expressed as

$$(3.8) \quad \bar{d} = \hat{V}^{-1}\hat{B}_w^{-1}\hat{B}_i^{-1}d \quad .$$

Let $\hat{d} = \hat{B}_i^{-1}d$. Then by (1.6) and (5.2)

$$\begin{aligned} \hat{d}_i &= B_i^{-1}d_i \\ \hat{d}_0 &= d_0 - A_i\hat{d}_i \\ \hat{d}_j &= 0 \quad j = 1,\ldots,k \quad , j \neq i \quad . \end{aligned} \tag{5.7}$$

Thus as a first step

$$\hat{d}_i = B_i^{-1}d_i \tag{5.8}$$

has to be calculated. If B_i^{-1} is represented in a factorized form

$$\hat{d}_i = B_i^{-1} d_i = \hat{V}_i^{-1} \hat{B}_{W_i}^{-1} B_{N_i}^{-1} d_i$$

$$= \hat{V}_i^{-1} \hat{B}_{W_i}^{-1} \hat{B}_{ij}^{-1} d_i \qquad (5.9)$$

where $\hat{B}_{ij}$ is the basis of the j-th block in the factorization of B_i, to which d_i is assumed to belong. But this is the same relation as for the level 1 forward transformation and hence it is possible to continue on the same lines for higher levels of factorization.

Proposition 4: To update a column corresponding to a path of length k, only the k inverses and k - 1 V matrices associated with that path are required in the forward transformation.

Proof: By induction. From the no factorization and the 1 level factorization it is true for k = 1 and k = 2. Assume it is true for k = 1,...,ℓ , $\ell \geq 2$. Then for k = ℓ + 1, by relations (3.8) and (5.7) a column d can be updated from knowledge of the level 1 WB inverse, V matrix and one block inverse. The latter is used to calculate $\hat{d}_i = B_i^{-1} d_i$ (see (5.8)). But this corresponds to updating a column associated with a path of length k - 1 = ℓ in the associated tree of the block basis B_i, and hence by the inductive assumption only ℓ inverses and ℓ - 1 of the V matrices are required. Thus together with the level 1 WB and V matrix a total of ℓ + 1 inverses and ℓ V matrices are used. $\therefore$ True for $\ell \longrightarrow$ true for ℓ + 1. ||

5.4-3 Observations

Consider a multilevel basis factorization where for L

levels each level ℓ block basis can in turn be factorized giving rise to two or more level $\ell + 1$ block basis's. Then the number of arcs in the associated tree, and hence the number of inverses and matrices in the total factorized representation, increases exponentially with L. But according to propositions 2 and 4 the effort to update the total factorized representation of the inverse, as well as that to update an incoming column in terms of a basis, increases only linearly, since it involves only the terms associated with one path. By proposition 3 the same is also true for the work on the backward transformation when pricing out only the columns associated with one path.

Observe that the advantages of factorization do not stem from its giving a more economic representation for the inverse (which is certainly not the case) but from the fact that only a fraction of the total information needs to be used on any iteration. Thus, if this fraction involves a smaller number of nonzeros than a general representation it will be of advantage to use the factorized representation.

5.5. The General Algorithm Using Nested Factorization

Recall the Coupling Constraints Algorithm developed in 3.4-2. In Step 1 all block problems are optimized. Then in Step 2 the PBP strategy is used to take advantage of the reductions in BTRAN that are made possible by the use of a factorized representation of the inverse (i.e. to price out columns associated with one path only). In a generalization to nested factorization both steps require or allow modifications.

For Step 1 notice that in the nested case each block problem is a block-angular linear problem with coupling constraints and can be solved by using the Coupling Constraints Algorithm.

For Step 2 it is possible to specialize the PBP strategy to pricing out columns in only one path, or in some subset of the paths. Although pricing out columns in only one path at a time would minimize time per iteration, it could tend to increase the number of iterations if there are too many paths, because the candidate is selected from a small subset of the non-basic columns, and hence is likely to be dropped later on.

Probably the best would be to select a set of complete paths whose set of arcs and nodes form a subtree, and to price out non-basic columns associated with it. As a straight extension of Proposition 3 it is possible to show that only the inverses and V matrices associated with that subtree will be required in the backward transformation. Moreover, if suitable criteria exist for what constitutes a "good" candidate (not only an improving one) it is possible to start pricing out columns on one path and calculate only the components of the Π's needed for it, and continue with columns on other paths, one path at a time, calculating Π components when needed, and stopping whenever a "good" candidate is found. Based on limited experience with GUB there is evidence that the standard simplex criteria when applied to the above restricted set of columns will select good candidates for the full problem.

In the following the use of such a PBP strategy is as-

sumed. Then the General Algorithm for the nested case can be stated in terms of the next two steps.

Step 1: On each path solve the block problems associated with its last arc (i.e. those that are not factorized further). If one such problem is infeasible STOP, the whole problem is infeasible. Otherwise set $\ell = L_{max}$ and go to Step 2-a.

Step 2: 2-a) Solve each level ℓ blockangular block problem using the PBP strategy. If some problem is infeasible STOP, the whole problem is infeasible. Otherwise go to 2-b.

2-b) If $\ell > 1$ set $\ell = \ell - 1$ and return to 2-a. Otherwise the solution is optimal (or unbounded, if the problem is unbounded).

5.6. Application to Staircase Problems

5.6-1 The Staircase Problem

Consider the problem

$$\max Z$$

$$\text{s.t.} \quad Z + C_1X_1 + C_2X_2 + C_3X_3 + \ldots\ldots + C_nX_n = 0$$

$$A_1X_1 = b_1$$

$$\text{(SP)} \quad D_1X_1 + A_2X_2 = b_2$$

$$D_2X_2 + A_3X_3 = b_3$$

$$\ddots$$

$$D_{n-1}X_{n-1} + A_nX_n = b_n$$

$$X_i \geq 0 \quad i = 1,\ldots,n$$

where for $i = 1,\ldots,n$

A_i is a $m_i \times n_i$ matrix

X_i is a $n_i \times 1$ matrix

b_i is a $m_i \times 1$ matrix

C_i is a $1 \times n_i$ matrix

and for $i = 1,\ldots,n-1$

D_i is a $m_{i+1} \times n_i$ matrix .

The above problem is called a Staircase Problem (SP), because of the staircase pattern of the non-zeros in its matrix. The SP can be expressed more compactly as

$$\min \sum_{i=1}^{n} C_i X_i$$

s.t. $$A_1 X_1 = b_1$$

$$D_{i-1} X_{i-1} + A_i X_i = b_i \qquad i = 2,\ldots,n$$

$$X_i \geq 0 \qquad i = 1,\ldots,n \ .$$

Relations $D_{i-1}X_{i-1} + A_iX_i = b_i$ will be referred to as the i-th "stage" or "time period"; $A_1X_1 = b$, as the first stage (or time period).

5.6-2 Nested Factorization for the Staircase Problem

Observe that for any stage (except the first and the last),

the variables with non-zero coefficients can only have non-zero coefficients in the preceding and the following stage. Hence if one stage is removed, say the k-th, then problem SP reduces to two smaller independent staircase problems, i.e.

$$\min \sum_{i=1}^{k-1} C_i X_i$$

$$\begin{aligned} \text{s.t.} \quad & A_1 X_1 = b_1 \\ & D_{i-1} X_{i-1} + A_i X_i = b_i \qquad i = 2,\ldots,k-1 \\ (SP_1) \quad & X_i \geq 0 \qquad i = 1,\ldots,k-1 \end{aligned}$$

and

$$\min \sum_{i=k}^{n} C_i X_i$$

$$\begin{aligned} \text{s.t.} \quad & D_{i-1} X_{i-1} + A_i X_i = b_i \qquad i = k+1,\ldots,n \\ (SP_2) \quad & X_i > 0 \qquad i = k,\ldots,n \end{aligned} \quad .$$

Instead stage k together with the objective row can be considered as the common rows (coupling constraints) of a block-angular linear problem with coupling constraints which has two blocks; the first block is formed using stages 1 through k - 1 and the second using stages k + 1 through n. Each of these, in turn, has a staircase structure of lower dimensionality, which can also be treated in the same way, leading to an application of nested factorization.

In particular, by choosing at each level of factorization the stage in the "middle" of the staircase as the coupling stage, it can be ensured that each resulting block problem will have

a staircase structure with at most half the number of stages.

Proposition 5: Let i for $i_I \leq i \leq i_F$ be the indices of the stages of a staircase problem, where i_I is the index of the first stage and i_F the index of the last stage. Then if the stage with index $i_k = i_I + [(i_F - i_I + 1)/2]$ is removed (where $|X|$ stands for integer value of X), each one of the two resulting staircase problems with indices $i_I \leq i \leq i_k - 1$ and $i_k + 1 \leq i \leq i_F$ has at most half the number of stages in the original one.

Proof: If $(i_F - i_I + 1)$, the number of stages in the staircase, is even then $[(i_F - i_I + 1)/2] = (i_F - i_I + 1)/2$
and

$$i_k = \frac{i_F + i_I + 1}{2} \quad .$$

The number of stages in the first resulting staircase is

$$(i_k - 1 - i_I + 1) = \frac{i_F + i_1 + 1}{2} - i_I = \frac{i_F - i_I + 1}{2} \quad .$$

i.e. half the number of stages of the old one. The second must have one stage less and hence also less than half those of the old one. IF $(i_F - i_I + 1)$ is odd, then the number of stages in the first resulting staircase is

$$(i_k - 1 - i_I + 1) = [(i_F - i_I + 1)/2] < (i_F - i_I + 1)/2$$

and in the second

$$(i_F - (i_k + 1) + 1) = (i_F - i_k) = (i_F - (i_I + \lceil (i_F - i_I + 1)/2 \rceil))$$

but $i_F - i_I$ is even, so

$$= (i_F - i_I - (i_F - i_I)/2) = \frac{i_F - i_I}{2}$$

$$< \frac{i_F - i_I + 1}{2} \quad . \qquad ||$$

Proposition 6: If N is the total number of stages of a staircase problem which is factorized, using the rule in Proposition 5 to choose the coupling stage, until in all branches there are single stage block problems, then the maximum level of factorization is given by

$$L_{max} = \left\lceil \lg N \,/\, \lg 2 \right\rceil \quad .$$

Comment: A staircase problem with N = 31 time periods would have four levels of factorization.

Proof: Consider the values of N for which $2^k \leq N < 2^{k+1}$ for $k = 1,2\ldots$. For different values of k this covers all $N \geq 2$, which are of interest here. For $k = 1$, $2 \leq N \leq 3$, $L_{max} = 1$ and it is true. Assume it is true for $k = 1, 2, \ldots, \ell$, $\ell \geq 1$. Then for $\ell + 1$ consider the values of N such that $2^{\ell+1} \leq N < 2^{\ell+2}$. Selecting a coupling stage as in Proposition 5, each resulting

block has $N_i < 2^{\ell+1}$, i = 1,2, and by inductive hypothesis these can be factorized giving a maximum level of factorization of ℓ, and hence a total of $\ell + 1$ is obtained for $k = \ell + 1$. $\therefore$ true for $k \leq \ell \longrightarrow$ true for $k \leq \ell + 1$. ||

5.6-3 Observations

From the development in section 5.6-2 it is apparent that the algorithm in 5.5 can be applied to solve staircase problems. Observe that:

1) The Working Basis has, for each factorization, the same number of rows as that associated with the time period. Thus in the overall factorization of the inverse there will be one inverse associated with each time period. About half of this inverse will correspond to block basis's in the final level of factorization and the rest to Working Basis's at all levels.

2) The maximum level of factorization is given by $L_{max} = [\lg N/\lg 2]$ (by Proposition 6). Thus to price out the columns of any time period, only $[\lg N/\lg 2]$ inverses will be required for the backward transformation (by Proposition 3). Similarly for the forward transformation by Proposition 4.

3) To update the factorized representation of the problem inverse after the replacement of one basic column by another, at most $[\lg N/\lg 2] + 1$ inverses (each of the dimension of a stage) and $[\lg N/\lg 2]$ V matrices have to be updated (by Proposition 2).

4) By the mechanics of the general algorithm, the solution of an N-stage staircase problem reduces to the solution of

about N/2 independent one-stage problems as a first step. As a second step the PBP strategy is used on the block-angular problems resulting from linking at each level 2 problems from the previous level through a coupling stage.

5.6-4 Other Nested Factorization Approaches for Staircase Problems

Dantzig [5] suggests that every other stage be considered as blocks of a block-angular problem and the remaining ones as the common rows; then the Working Basis when formed will have a staircase structure with only half the number of stages. He suggests factorizing the WB further along the same lines in a nested way until only one stage is left. Since in the GBBF method a WB inverse may be updated by both elementary column and elementary row matrices, it becomes necessary to develop formulas to update the factorized representation of the inverse when a row matrix updates the unfactorized representation. Especially when there are many levels of factorization, it is necessary to follow the implications of one such update for all terms and quantities related to all higher levels of the factorization. Thus the nested updating procedure in this case can be expected to be more complex than in the approach taken in 5.6-2.

Observe also that if the first stage in an N-stage staircase problem is taken as the common rows, then the resulting block-angular problem with coupling constraints has only one block, whose structure is staircase with N - 1 stages. This

approach leads to a nested factorization with an N - 1 level of factorization. By the results in proposition 2, 3 and 4 the effort for updating the representation of the inverse, or for a backward transformation or a forward transformation, depends on the maximum level of factorization, which in this case grows linearly with N instead of logarithmically as in the approach taken in 5.6-2. Hence it appears to be less promising and will not be pursued further.

On the other hand, by considering the last stage of a staircase basis as a block problem in a block-angular basis factorization, a WB with an N - 1 stage staircase is obtained. By treating each such resulting staircase WB in the same way this leads to an N - 1 level factorization. A nested factorization along these lines was proposed by Saigal [34] for staircase problems.

5.6-5 Efficiency Considerations

Observe that, as was pointed out in 5.4-3, the advantages of factorization do not derive from a more economic representation for the inverse (for, in general, this will not be the case) but from the fact that only a fraction of the total information needs to be used on any given iteration, and if this fraction happens to involve a smaller number of non-zeros (than would a general representation) it will be of advantage to use the factorized representation of the inverse.

These advantages can be expected to be independent of the level of factorization whenever only the information correspond-

ing to one path in the associated tree is required, as for instance for the forward transformation and for updating the inverse; and also for the backward transformation when only columns associated with one path are priced out. As the level of factorization increases the number of paths in the associated tree can be expected to increase exponentially with it, and the columns associated with any one path are likely to constitute a small fraction of the total. In this case, although pricing out the columns corresponding to only one path would be best as regards time per iteration, it may increase too much the total number of iterations to solve the problem, because the candidate is selected from a small subset of the non-basic columns, and there is high probability of the selected column being dropped later from the basis.

But if columns associated with more than one path are priced out, for example those corresponding to some subtree, then from some level on, say level k, all the terms in the factorized representation of a level k block inverse are required for the backward transformation. In this case a general representation may be more economic for the block inverses at level k, limiting the maximum level of factorization to k.

Only extensive experimental tests on a variety of real problems can show whether or not this limitation is outweighed by the possible advantages in the forward transformation and in the updating of the representation of the inverse.

CHAPTER 6

CONCLUSIONS

A General Block-Angular Basis Factorization (GBBF) method has been presented together with an efficient procedure to update the factorized representation of the inverse after one column replaces another in the basis.

The use of this representation of the inverse in a revised Simplex Method for block-angular linear problems has the advantage that, though the total number of non-zeroes in the factorized representation may be higher than for a general representation of the inverse, only a fraction of these terms need to be used and updated on any given iteration.

It also allows unifying existing Partitioning and Decomposition methods (not based on the Dantzig-Wolfe decomposition principle) as variants of a revised Simplex Method using the GBBF form of the inverse, differing from each other with respect to the criteria used to select the vector pair to enter and to leave the basis. This opens the way to extensive testing to compare these methods on practical problems since only one computer program has to be written, having the different strategy options to select the vector pair to enter and leave the basis.

The approach is easily extended to nested applications

and can be applied in particular to the solution of staircase problems.

APPENDIX

EXPERIMENTAL RESULTS WITH THE COUPLING CONSTRAINTS ALGORITHM

In Figures A1 and A2 a more detailed information flow-sheet for the two step Coupling Constraints Algorithm is given (see section 3.4-2). This algorithm was implemented in a FORTRAN computer code under the name G-GUB. G-GUB in turn was developed as an extension of LPM1, an all in-core FORTRAN linear programming code written at Stanford by J.A. Tomlin. LPM1 stores the problem matrix by columns packed in a vector of non-zeroes, a vector of the same dimensions giving for each non-zero coefficient its row index, and a vector giving for each column the position of its first non-zero element in the two above vectors. The eta file is stored according to the same principle. It uses an L-U factorization for inverting the basis, followed by product-form updates.

The G-GUB FORTRAN Code

G-GUB was conceived as an out-of-core code, where at each time data for one block, matrix and eta file, is held in core in the form required by LPM1, while in the meantime the data for all other blocks is kept on a disk. The working basis inverse, D_0 and V are stored in the same way as a block 0. Whenever we

need the data for another block, the one in core is written out to disk (unless it has not changed) and the new one is read in. Due to the packing scheme used by LPM1, which was designed as an in-core code, the I-O operations for G-GUB are somewhat inefficient. From the viewpoint of anlyzing experimental results, however, this is not serious. For an experimental code such as G-GUB the time spent on I-O can be measured, which allows us to make comparisons on computation times alone, or to have an estimate of the effect of inefficiencies due to I-O. Because of this, it was felt that the time involved in coding could be considerably shortened by adapting an existing LP code instead of writing a new one with superior I-O facilities.

Other computational characteristics of G-GUB are:

1) Block inverses were inverted whenever N1 new etas had been added to it since its last inversion, and the same for the Working Basis. (For tests N1 = 30.)

2) Every N2 iterations the solution was recomputed by solving $X_B = \hat{V}^{-1}\hat{B}_W^{-1}B_N^{-1}b$. A first step for this is to calculate for each $i = 1,\ldots,k$, $\beta_i = B_i^{-1}b_i$. When doing this the accuracy of the computed β_1 was checked. If the maximum row error exceeded a tolerance, the corresponding block basis was re-inverted even though it was not necessary by the criterion in (1). At the same time the corresponding $V_i = B_i^{-1}C_i$ was re-computed using the new representation for the inverse. After this step the Working Basis was reinverted with its recomputed columns. The accuracy of the X_B thus recomputed was always found to be good. (For tests N2 = 60.)

3) All computations were performed on an IBM 360/91 at the Stanford Linear Acceleration Center (SLAC). The computing times reported are CPU seconds.

Description of Problems

Three problems were available. They are described in Table 1.

Table 1. Description of Problems

Problem	FIXMAR	FORESTRY	DINAMICO
Total number of rows	325	404	417
Total number of columns	452	603	527
Total density	1.8%	1.6%	1.8%
Number of blocks	4	6	3
Common rows	18	11	56
Block 1 rows/column/density	92/114/6.0%	73/103/6.1%	117/177/3.9%
Block 2	73/94/ 5.4%	47/71/ 12.3%	108/164/4.3%
Block 3	57/125/3.5%	69/109/8.9%	136/192/4.5%
Block 4	85/118/8.7%	72/134/5.7%	
Block 5		63/89/ 12.1%	
Block 6		69/97/ 7.4%	

First Runs

The first experiments, with an early version of the code, were done to compare solution times of G-GUB and MPS-360. The results with the two problems available at that time are given in Table 2. Note that the solution times of the experimental first version G-GUB is comparable with that of the commercial MPS-360 system.

Table 2. Solution Times Using G-GUB and MPS-360

Problem \ Code	G-GUB making first blocks feasible	G-GUB making first blocks optimal	MPS-360
FIXMAR	22	21	36
DINAMICO	126	113	112

Second Runs

The above times for G-GUB were considered encouraging. It was felt that for later tests LPM1 should be used as the standard LP since then the times would not be affected by differences in the codes and would be directly comparable. Besides, if G-GUB performed better, it was important to determine to what degree this was due to the Step 1 procedure, to the GBBF representation, or to the partial pricing strategy used in

conjunction with this latter one.

Therefore some slight modifications were introduced to the code, which allowed us to test different options. These options were

A) Basis representation using GBBF or the standard LPM1 LU factorization with product form updates for the basis of the whole problem.

B) Step 1: Here we considered three options: (1) solving blocks to optimality, (2) making blocks feasible, or (3) using standard Phase 1 without treating blocks first.

C) Pricing: (1) Partial Block Pricing (PBP) or (2) total pricing at each iteration (total).

By a combination of these options the following strategies could be tested:

Strategy	Basis Representation	Step 1	Pricing
-1	GBBF	feasibility	PBP
0	GBBF	optimality	PBP
1	GBBF	optimality	total
2	GBBF	no	PBP
3	GBBF	no	total
5	LPM1	no	total
5a	LPM1	no	PBP

Using problem FIXMAR these strategies were compared. The results are given in Table 3.

Table 3. Comparison of Strategies on Problem FIXMAR

Strategy	-1	0	1	2	3	5	5a
Total CPU sec	22.12	21.04	25.45	38.52	38.70	35.02	47.65
Core used	156k	156k	156k	156k	156k	200k	200k
I/O CPU sec	4.30	2.98	4.35	10.99	10.53	--	--
Total-I/O CPU sec	17.82	18.06	21.10	27.53	28.17	35.02	47.65
Comp. time/ Step 2 iter.	3.35	3.46	4.42	3.40	4.41	--	--
Comp. time/ LMP1 iter.	--	--	--	--	--	5.15	5.30

Third Run

By this time strategies 0 and 5 were compared on problem FORESTRY. The results are given in Table 4.

Table 4. Comparison of Strategies 0 and 5 on Problem FORESTRY

Strategy	0	5
Total CPU sec	60.52	91.22
Core used	158k	270k
I/O	11.82	--
Total - I/O CPU sec	48.70	91.22
Comp. time/ Step 2 iteration	9.30	--
Comp. time/ LPM1 iteration	--	10.1

Analysis of Results

1) With respect to comparable general LP's, the CCA algorithm can produce substantial reductions in overall computation time for block-angular linear problems with coupling constraints, as can be seen by comparing the total solution times for problems FIXMAR and FORESTRY using strategies 0 (CCA algorithm) and 5 (general LP).

2) If FIXMAR is any indication then each one of the three options in the CCA algorithm helps in reducing the overall solution time. The best results are obtained when all three are in effect; in this case we get a reduction by approximately a factor of 2.

3) Note that strategies 1 and 3 differ only in that 1 makes use of the Step 1 option and this gives about a 25% reduction in computation times. This would mean, if it were true in general for block-angular problems with coupling constraints, that general LP's could be made more efficient for this type of problem by using this strategy.

4) The mean time per iteration was seen to increase with the number of block vectors in the Working Basis. This relationship is plotted on Fig. A3 for DINAMICO, for which the effect is more pronounced due to the large number of common rows. Notice that the mean time per iteration with 45 block vectors in the Working Basis is about three times that with 0. This effect is due mainly to longer transformation times, especially in FTRAN, as the number of non-zeroes in the WB and in the V matrices increases linearly with the number of

block vectors in the Working Basis, and to an increase in the frequency of the more expensive type 3 updates. (The more block vectors in WB the smaller the probability of $v_r = 0$.)

This suggests a strategy modification to reduce the mean time per iteration. At the end of Phase 1, all block variables in the Working Basis are treated as parameters fixed at their current value. Thus there are no block vectors in the Working Basis and V = 0 and we get faster iterations because of the reduced transformation time. When the number of block vectors in the Working Basis has again increased to a level similar to that at the end of Phase 1, the variables treated as parameters are considered as candidates and their values modified in the direction to improve the objective function until they reach their bounds or displace a basic variable.

FIGURE A1

Coupling Constraints Algorithm:

Step 1

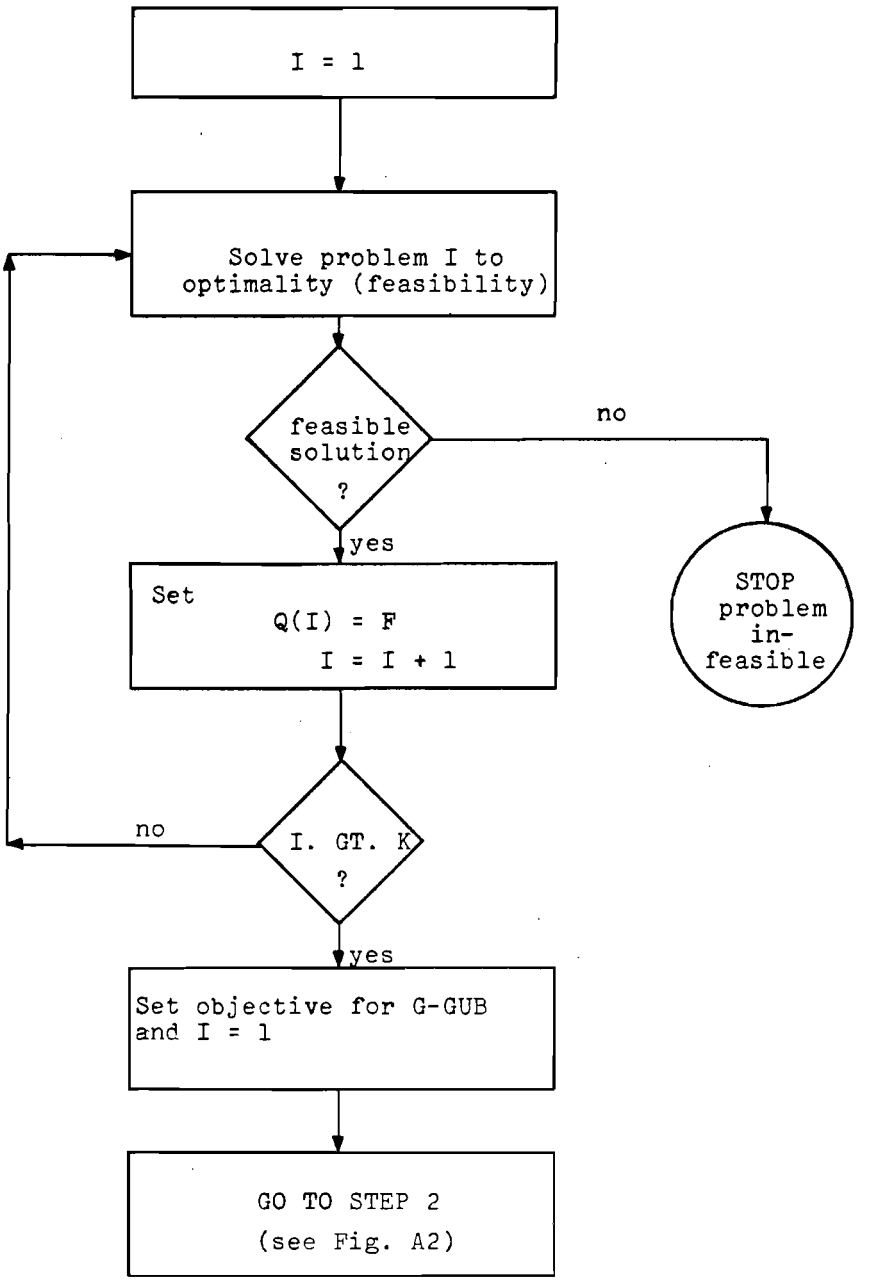


FIGURE A2

Coupling Constraints Algorithm: Step 2

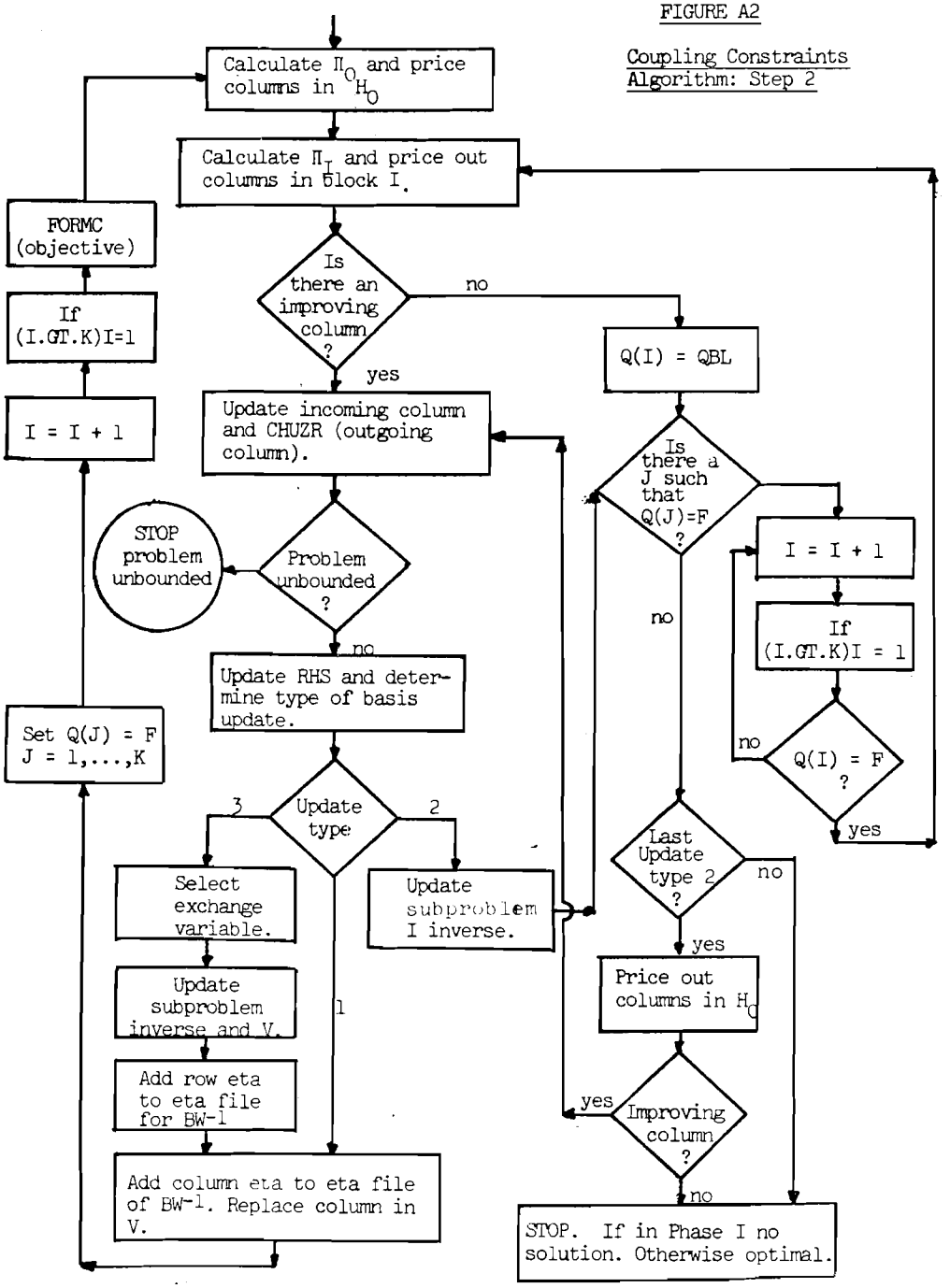
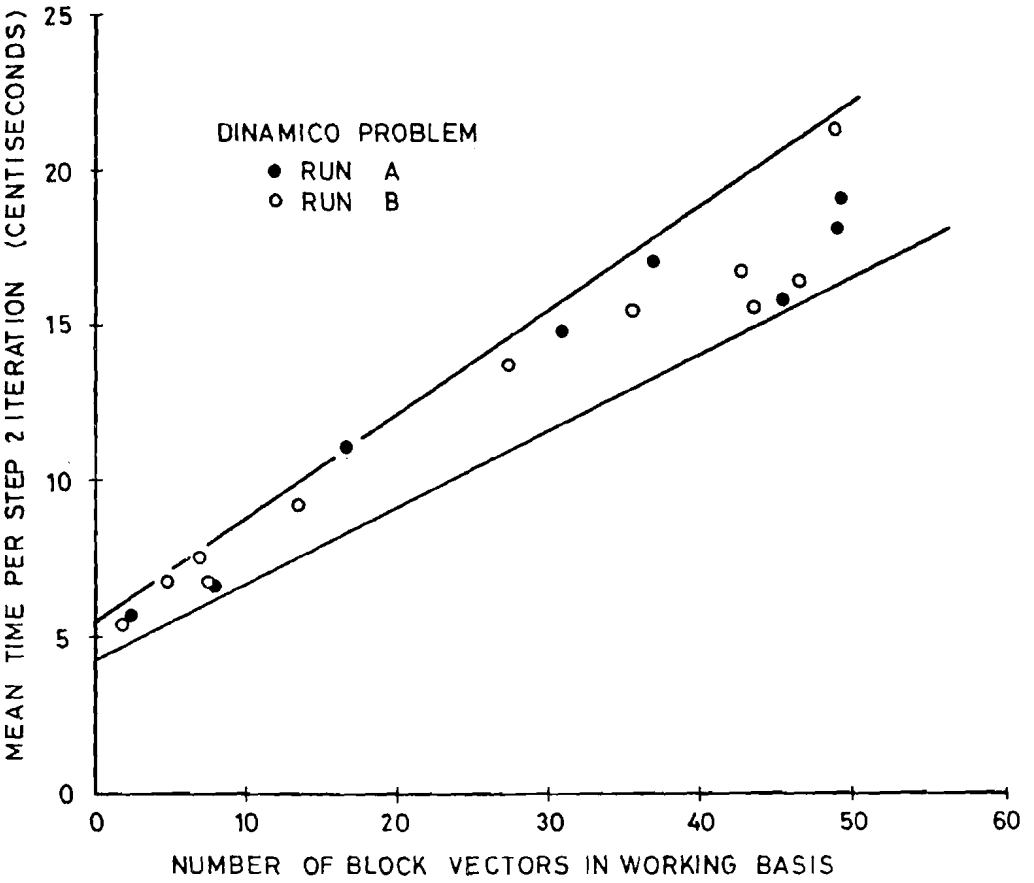


FIGURE A3. MEAN TIME PER STEP 2 ITERATION VS. NUMBER OF BLOCK VECTORS IN WORKING BASIS.



BIBLIOGRAPHY

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

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

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

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

[5] Dantzig, G.B., (1973) "Solving Staircase Linear Programs by a Nested Block-Angular Method", Technical Report 73-1, Department of Operations Research, Stanford University, Stanford.

[6] Dantzig, G.B., et al. (1972) "On the Need for a System Optimization Laboratory", Department of Operations Research Technical Report No. 72-11, Stanford University, Stanford.

[7] Dantzig, G.B., (1963) Linear Programming and Extensions, Princeton University Press, Princeton, New Jersey.

[8] Dantzig, G.B., (1963) "Compact Basic Triangularization for the Simplex Method", in R. Graves and P. Wolfe (eds.), Recent Advances in Mathematical Programming, McGraw-Hill Book Company, New York.

[9] Dantzig, G.B., (1955) "Upper Bounds, Secondary Constraints and Block Triangularity in Linear Programming", Econometrica, Vol. 23, No. 2, pp. 174-183.

[10] Dantzig, G.B., (1951) "Maximization of a Linear Function of Variables subject to Linear Inequalities", in T.C. Koopmans (ed.), Activity Analysis of Production and Allocation, Cowles Commission Monograph No. 13, J. Wiley, New York.

[11] Dantzig, G.B. and W. Orchard-Hays, (1954) "The Product Form of the Inverse in the Simplex Method", Mathematical Tables and Aids to Computation, Vol. 8, pp. 64-67.

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

[13] Dantzig, G.B. and P. Wolfe, (1960) "Decomposition Principle for Linear Programs", Operations Research, Vol. 8, No. 1, pp. 101-111. See also Econometrica, Vol. 23, No. 4, pp. 767-778.

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

[15] Gass, S.T., (1966) "The Dualplex Method of Large-Scale Programs", Operations Research Center Report 66-15, University of California, Berkeley.

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

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

[18] Grigoriadis, M.D. and K. Ritter, (1969) "A Decomposition Method for Structured Linear and Nonlinear Programs", J. Computer and Systems Sciences, Vol. 3, No. 4, pp. 335-360.

[19] Grigoriadis, M.D. and W.W. White, (1973) "A Framework for the Experimental Study of Partitioning Methods for Structured Linear Programs". Presented at 8th International Symposium on Mathematical Programming, Stanford.

[20] Hartman, J.K. and L.S. Lasdon, (1970) "A Generalized Upper Bounding Method for Doubly Coupled Linear Programs", NLRQ, pp. 411-429.

[21] Heesterman, A.R.G. and J. Sandee, (1965) "Special Simplex Algorithm for Linked Problems", Management Science, Vol.11, No. 3 pp. 420-428.

[22] Kaul, R.N., (1965) "An Extension of Generalized Upper Bounded Techniques for Linear Programming", Operations Research Center Report 65-27, University of California, Berkeley.

[23] Knowles, T.W., (1973) "An Artifical-Variable Elimination Method for Block-Diagonal Programming Problems", Operations Research, Vol. 21 No. 3 pp. 712-727.

[24] Lasdon, L.S., (1970) Optimization Theory for Large Systems, Macmillan.

[25] Manne, A.S., (1974) "Waiting for the Breeder", to appear in Review of Economic Studies.

[26] McBride, R.D., (1973) "Factorization in Large-Scale Linear Programming", Western Management Science Institute Working Paper No. 200, University of California, Los Angeles.

[27] Müller-Merbach, H., (1965) "Das Verfahren der direkten Dekomposition in der Linearen Planungsrechnung", Ablauf-und-Planungsforschung 6, pp. 306-322.

[28] Ohse, D., (1973) "A Dual Decomposition Method for Block Diagonal Linear Programs", Zeitschrift für Operations Research, 17.

[29] Orchard-Hays, W., (1973) "Practical Problems in LP Decomposition", in D.M. Himmelblau, (ed.) Decomposition of Large Scale Problems, North-Holland

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

[31] Ritter, K., (1967) "A Decomposition Method for Linear Programming Problems with Coupling Constraints and Variables", Mathematics Research Center Technical Summary Report No. 739, University of Wisconsin, Madison.

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

[33] Rutten, D.P., (1972) "A Theoretical Comparison of Some Partitioning Methods for Solving Structured Linear Programs", Graduate School of Business, Indiana University, presented at the ORSA Meeting, New Orleans, La.

[34] Saigal, R. (1969) "Compact Basis Representation for Dynamic Linear Programs", Center for Research in Management Working Paper No. 266, University of California, Berkeley.

[35] Saigal, R., (1966) "Block-Triangularization of Multi-Stage Linear Programs", Operations Research Center Report 66-9, University of California, Berkeley.

[36] Sakarovitch, M. and R. Saigal, (1967) "An Extension of Generalized Upper Bounding Techniques for Structured LP Problems", SIAM Journal on Applied Mathematics, Vol. 15 No. 4, pp. 906-914.

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

[38] Winkler, C., (1974) "On a Generalized GUB Basis Factorization Algorithm for Block-Angular Linear Problems with Coupling Variables". (Report to be published by Stanford's SOL).

[39] Webber, D.W. and W.W. White, (1968) "An Algorithm for Solving Large Structured Linear Programming Problems", IBM, New York, Scientific Report No. 320-2946, New York.

[40] Wolfe, P. and L. Cutler, (1963) "Experiments in Linear Programming", in R.L. Graves and P. Wolfe (eds.), Recent Advances in Mathematical Programming, McGraw-Hill Book Company, New York.