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

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

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

and from (1.2)

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

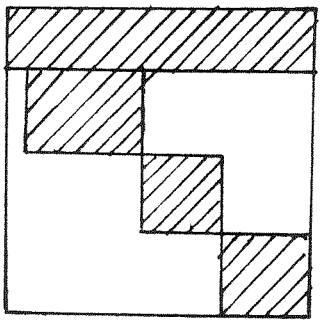
We can write for (1.2)

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

and hence

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

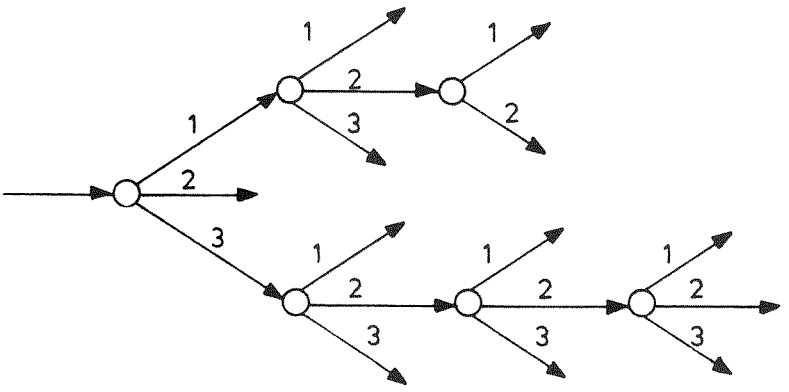
In the nested factorizations that will be considered here, at least for one j B_j has the structure of a blockangular problem with coupling constraints, i.e. of the type:



(5.3)

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

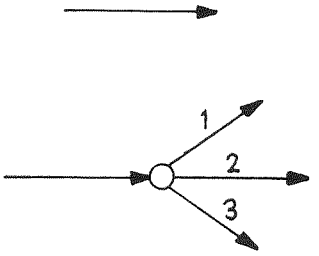
The basis factorization developed in Chapter 2 will be referred to as a 1 level factorization (or the level 1 factoization) and accordingly its Working Basis will be called the level 1 Working Basis and its block basis' the level 1 block basis'. If one or more of the block basis' are factorized again, then each one of them gives rise to a level 2 Working Basis and to level 2 block basis'. 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' with an outgoing arc.

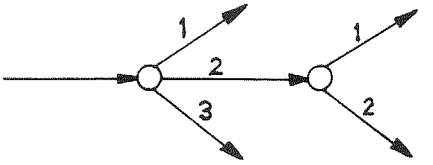
For example, for a blockangular basis B_T with three blocks



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 be the outgoing column, and

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

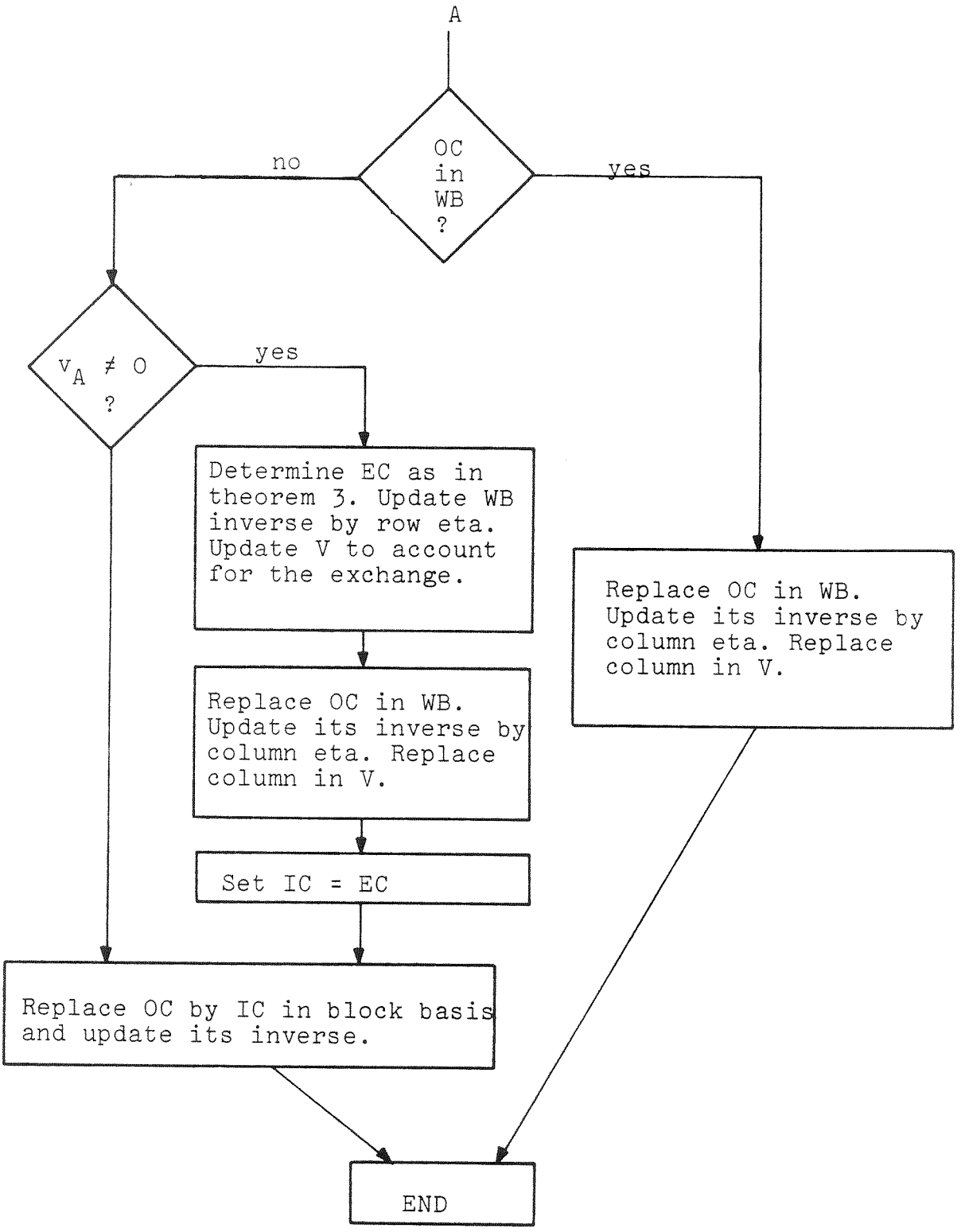
It will be convenient to modify the Information Flow-Sheet of the Updating Procedure for Block-angular Linear Problems with Coupling Constraints (see Figure 2), so that the updating of the inverse of a block basis is done as a last step (when the case arises). This modified Information Flow-Sheet is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

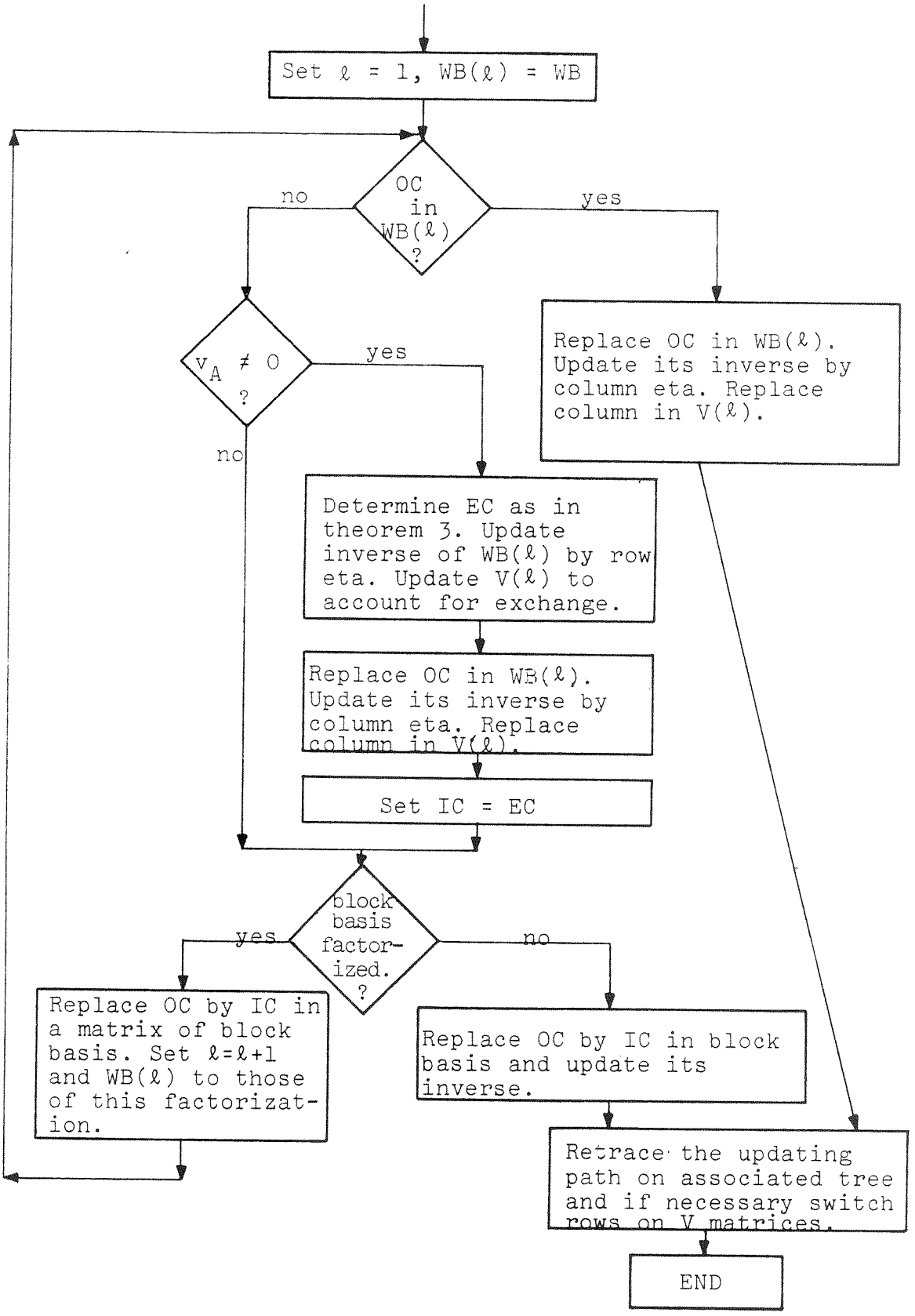
Recall from section 2.3 that

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

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'. 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 nonzero 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 on 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

$$\Pi_O = \hat{\Pi}_O$$

and

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

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

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

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 nonzero and Step 1 has to be performed no matter if 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 $\ell = 1,\ldots,\ell$, $\ell \geq 2$. 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

$$d = \begin{pmatrix} d_0 \\ d_1 \\ \vdots \\ d_i \\ \vdots \\ d_{\bar{k}} \end{pmatrix}$$

Then if d belongs to block i $d_i = 0$ for $j = 1,\ldots,\bar{k}$, $j \neq i$

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

Thus to compute $\Pi^t d$ for columns associated to 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 calculations 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

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

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

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

Thus as a first step

$$\hat{d}_i = B_i^{-1}d_i \qquad (5.8)$$

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, and 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 to 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. ∴ True for ℓ ⟶ 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 ℓ + 1 block basis'. 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 it 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 non-zeros 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 nonbasic 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

$$\max Z$$

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

$$\text{(SP)} \quad \begin{aligned} A_1X_1 &= b_1 \\ D_1X_1 + A_2X_2 &= b_2 \\ D_2X_2 + A_3X_3 &= b_3 \\ &\ddots \\ D_{n-1}X_{n-1} + A_nX_n &= b_n \end{aligned}$$

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

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 nonzeros in its matrix. The SP can be expressed more compactly as

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

$$\text{s.t.} \quad A_1 X_1 = b_1$$

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

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

Relations $D_{i-1}X_{i-1} + A_iX_i = b_i$ will be referred 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 preceeding 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.

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

$$\text{s.t.} \quad A_1 X_1 = b_1$$

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

$$(SP_1) \qquad X_i \geq 0 \qquad i = 1,\ldots,k-1$$

and

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

$$\text{s.t.} \quad D_{i-1} X_{i-1} + A_i X_i = b_i \qquad i = k+1,\ldots,n$$

$$(SP_2) \qquad X_i > 0 \qquad i = k,\ldots,n$$

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 which 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 $i_k + 1 \leq i \leq i_F$ have 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

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

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' in the final level of factorization and the rest to Working Basis' 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 on 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 in 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 nonzeros (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.

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 tell whether or not this limitation is outweighted 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 to unify existing Partitioning and Decomposition methods (not based on 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, and all incore 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 somehow inefficient. From the view point of analysing experimental results however, this is not serious. For an experimental code like 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. 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 criterium 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 thus recomputed X_B 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) use 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 iter.	9.30	--
Comp. time/ LPM1 iter.	--	10.1

Analysis of Results

1) The CCA algorithm can produce substantial reductions in overall computation time for block-angular linear problems with coupling constraints, with respect to comparable general LP's, 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, where 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 noticed 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 BW 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

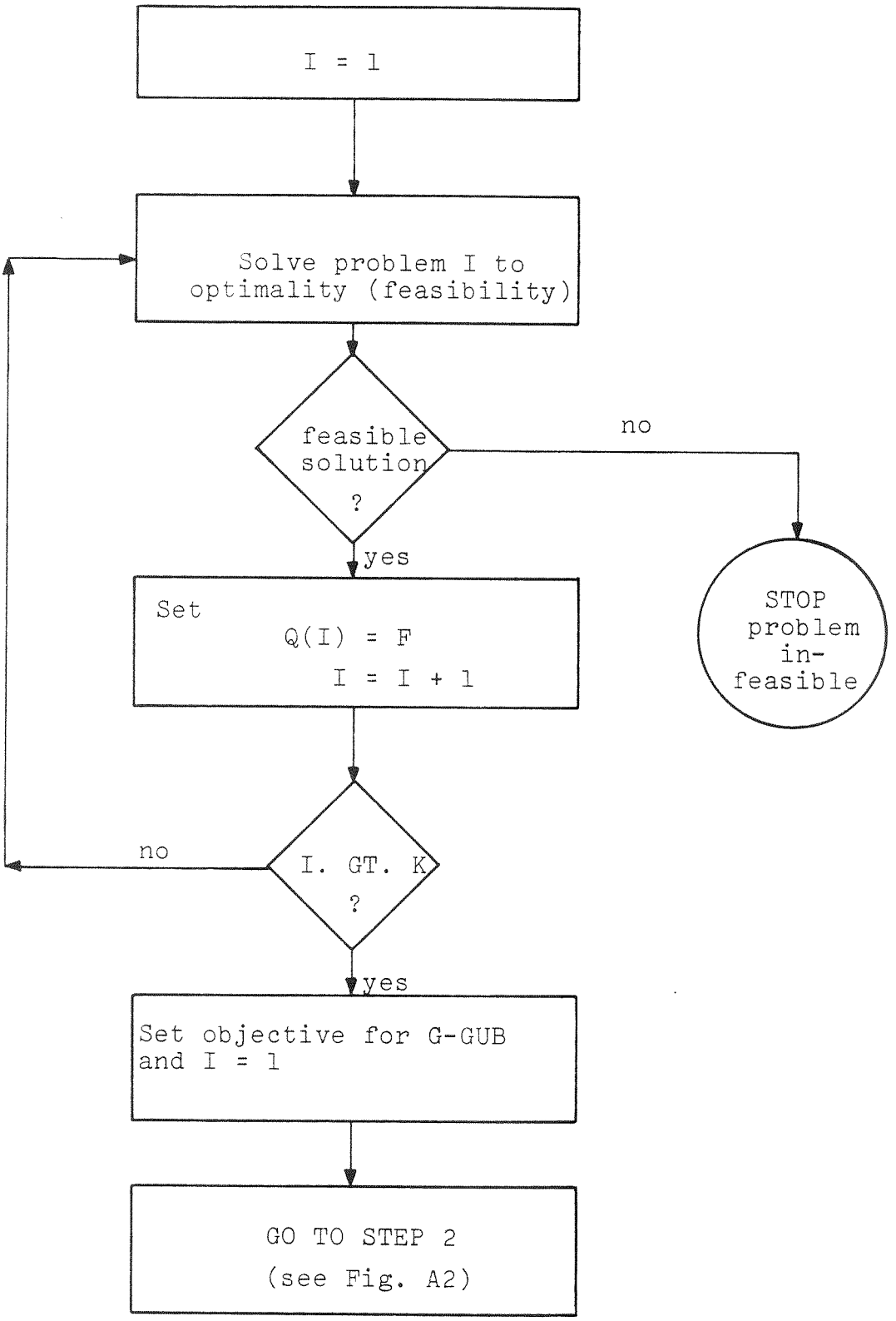


FIGURE A2

Coupling Constraints Algorithm: Step 2

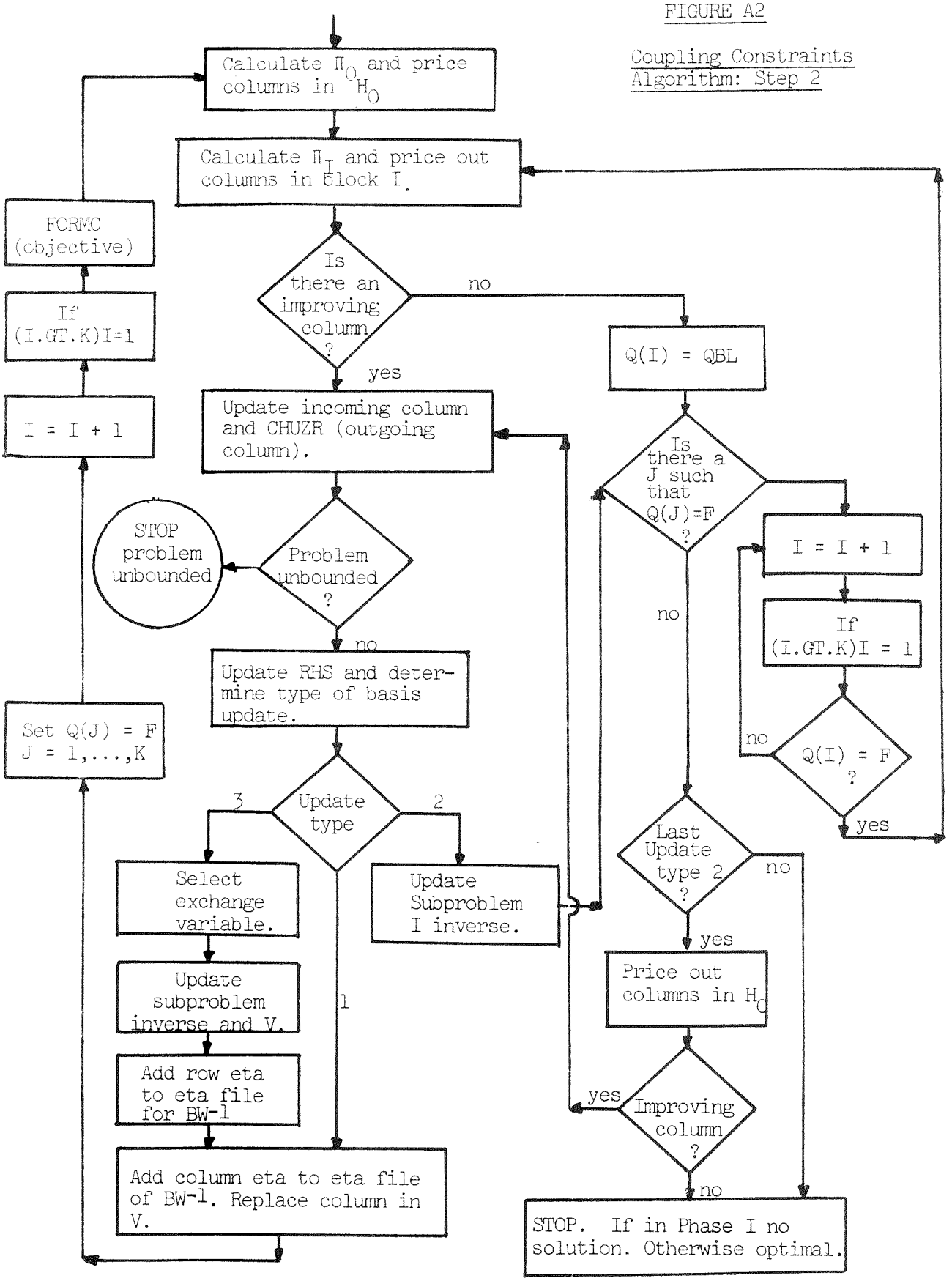
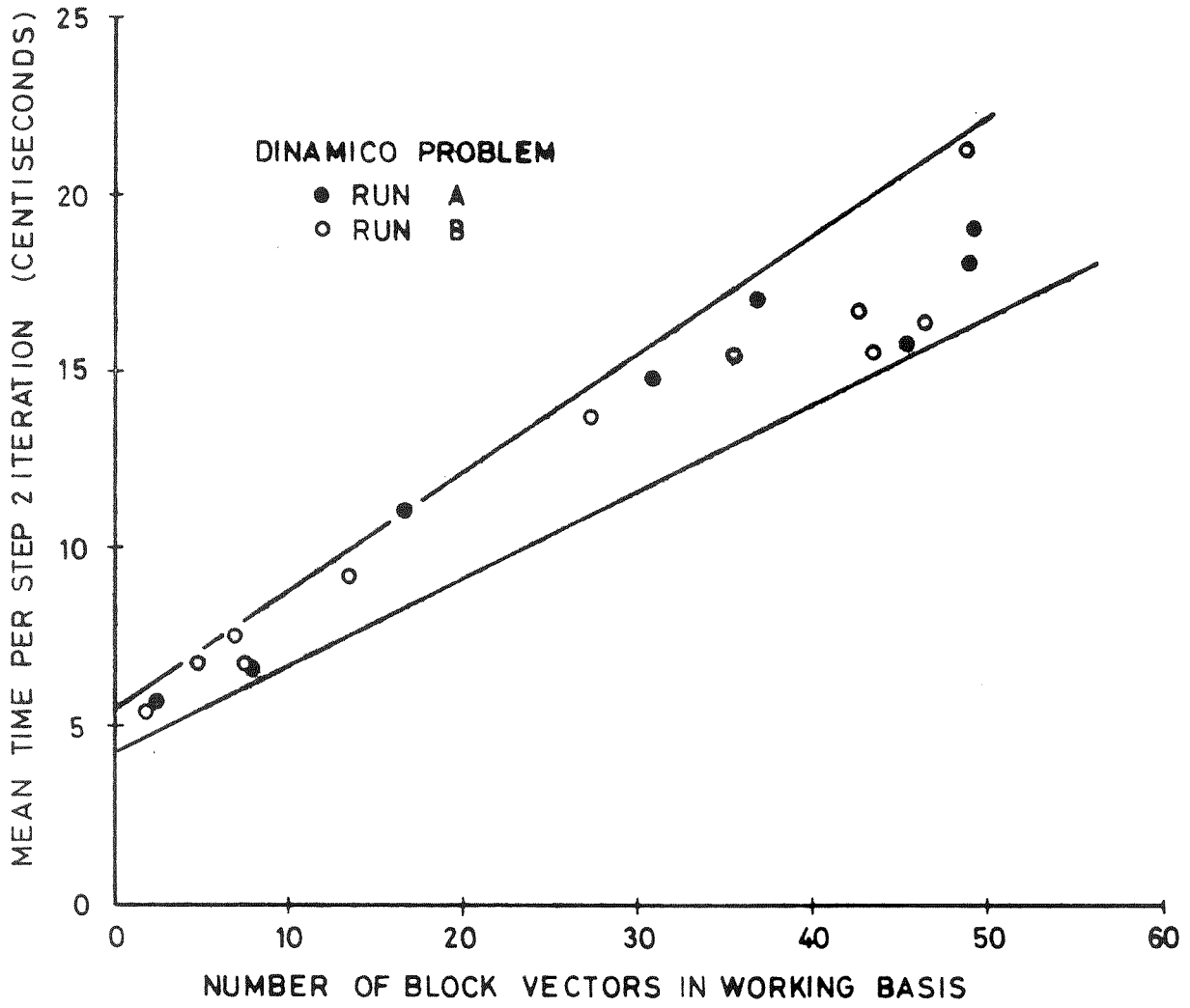


FIGURE A3. MEAN TIME PER STEP 2 ITERATION VS. NUMBER OF BLOCK VECTORS IN WORKING BASIS.



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.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: SOL 74-19
2. GOVT ACCESSION NO.:
3. RECIPIENT'S CATALOG NUMBER:

4. TITLE (and Subtitle): BASIS FACTORIZATION FOR BLOCK-ANGULAR LINEAR PROGRAMS: UNIFIED THEORY OF PARTITIONING AND DECOMPOSITION USING THE SIMPLEX METHOD

5. TYPE OF REPORT & PERIOD COVERED: Technical Report

6. PERFORMING ORG. REPORT NUMBER:

7. AUTHOR(s): Carlos WINKLER

8. CONTRACT OR GRANT NUMBER(s): N-00014-67-A-0112-0011

9. PERFORMING ORGANIZATION NAME AND ADDRESS:
Department of Operations Research
Stanford University
Stanford, CA 94305

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: NR-047-064

11. CONTROLLING OFFICE NAME AND ADDRESS:
Operations Research Program Code 434
Office of Naval Research
Arlington, Virginia 22217

12. REPORT DATE: December 1974

13. NUMBER OF PAGES: 128

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office):

15. SECURITY CLASS. (of this report):

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT (of this Report)

This document has been approved for public release and sale; its distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

LARGE-SCALE SYSTEMS,
LINEAR PROGRAMMING BASIS
FACTORIZATION
PARTITIONING AND DECOMPOSITION

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

SEE OTHER SIDE

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

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

Carlos Winkler

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.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)