Mathematical Programming 1 (1971) 195–216. North-Holland Publishing Company

REINVERSION WITH THE PREASSIGNED PIVOT PROCEDURE *

Eli HELLERMAN
Bureau of the Census, Washington, D.C., U.S.A.

and

Dennis RARICK
Management Science Systems, Rockville, Maryland, U.S.A.

Received 8 October 1970
Revised manuscript received 27 February 1971

Mathematical programming computer systems using the product form in the inverse (PFI) must periodically resort to a reinversion with the current basis in order to reduce the amount of work to be done in the succeeding iterations.

In this paper, we show the consequences of column, pivot selection and sequence upon the transformation vector (ETA) density and give an algorithm which will tend to minimize eta density and work done per iteration.

The algorithm has been implemented and tested as a replacement for the previous inversion algorithm on the OPTIMA system for the CDC 6000 computers and on the MPS/III mathematical programming system for the IBM 360 computer. A comparative performance table is given.

1. Introduction

Mathematical programming production codes generally use the product form of the inverse (PFI) in the course of obtaining solutions. Studies have shown that PFI or some variant of PFI seems to be one of the more efficient techniques available for use on large problems [3, 4]. However, a characteristic of PFI is that with each pivot an increasing additional amount of work must be done in order to compute the

* This paper was presented at the 7th Mathematical Programming Symposium 1970, The Hague, The Netherlands.

"pricing" vector and in transforming the "selected" vector to its representation in terms of the current basis. This comes about because each pivot adds an additional transformation vector (η) to the set of transformation vectors (ETA) which imply the inverse. It is easy to see that after some number of pivots the amount of work done per iteration may be significantly greater than what it was at the initial iteration. At some point the economics of the situation demands that the larger set of ETA be replaced by a smaller set representing the same basis. This replacement is usually accomplished by a program called "INVERT" or "REINVERT" which resides within the mathematical programming system. The mathematical problem which "INVERT" tries to solve may be stated in the following manner:

Given – a set of basic variables

Find – a set of transformation vectors (ETA) which imply the inverse of the basis in such a way as to

a) minimize the number of nonzero elements in ETA and

b) minimize the work done in forming the ETA.

Markowitz's [1] observations in connection with the minimization of the number of nonzero elements when forming the ETA have long been the starting point for reinversion techniques in mathematical programming.

In this paper we

a) Review some pertinent aspects of PFI

b) Show the consequences of pivot sequence upon ETA nonzero density

c) Give an algorithm which seeks to find a pivot sequence which will tend to minimize the ETA nonzero density and the work which must be done.

The algorithm discussed is called "Preassigned Pivot Procedure" and it has been implemented and tested as a replacement for the previous inversion scheme on the OPTIMA mathematical programming system for the CDC 6000 series computers and on the MPS/III mathematical programming system for the IBM 360 computer. It performs the reinversion some six to ten times faster than its predecessor on OPTIMA. The Appendix contains a performance comparison between this procedure and the IBM MPS/360 inversion.

It has recently been called to our attention that our work is somewhat related to that of Steward [2].

2. Pertinent aspects of PFI

It is well known that under suitable conditions a matrix A may be transformed to the identity matrix I by a series of elementary row transformations. Furthermore an elementary row transformation on A may be expressed as the product EA where E is an elementary matrix whose unique column is in the column index position corresponding to the pivot row index position. By way of illustration, suppose we have the vector

$$\begin{bmatrix} 1 \\ -1 \\ 5 \\ 2 \\ 12 \\ -7 \end{bmatrix}$$

and we wish to apply an elementary row transformation so as to make it

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

what must E become?

Another way of expressing the question is

$$\begin{bmatrix} 1 & & & ? & \\ & 1 & & ? & \\ & & 1 & ? & \\ & & & 1 & ? \\ & & & ? & \\ & & & ? & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ 5 \\ 2 \\ 12 \\ -7 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$E \qquad A = I$$

Upon solution we find

$$\begin{bmatrix} 1 & & & -1/12 & \\ & 1 & & 1/12 & \\ & & 1 & -5/12 & \\ & & & 1 & -2/12 \\ & & & 1/12 & \\ & & & 7/12 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ 5 \\ 2 \\ 12 \\ -7 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

and we see immediately that the rule for forming the elements of the

unique column of E^p (the superscript denotes the pivot row-column index) is

$$\eta_p = 1/a_p \qquad \text{where } p = \text{pivot index}$$

$$\eta_i = -a_i/a_p \qquad (i \neq p)\,.$$

This unique column is indeed the η we previously mentioned. In actual practice it is only necessary to record the η and its pivot index rather than the full E^p in order to apply E^p as needed. Also the zeros within an η need not be recorded. There are other more abbreviated ways to carry a modified η but for the sake of exposition we will adhere to the rules above.

The rule for applying an η to an arbitrary vector V is as follows:

a) Extract $v_p = V_p$, then set $V_p = 0$. (The p comes from E^p and v_p is now called the scalar multiplier.)

b) Compute $\overline{V}_i = V_i + v_p\,\eta_i$.

Note that if $v_p = 0$ than $\overline{V}_i = V_i$ and no element of V_i will change. The equivalence of our rules and the arithmetic in extenso of $E_p V$ may easily be verified by the reader.

An example of the rules is now given. Suppose we have

$$X = \begin{bmatrix} 1 \\ -1 \\ 5 \\ 2 \\ 12 \\ -7 \end{bmatrix} \qquad \text{and } Y = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 0 \\ 6 \end{bmatrix} \qquad \text{and } \eta^5 = \begin{bmatrix} -1/12 \\ 1/12 \\ -5/12 \\ -2/12 \\ 1/12 \\ 7/12 \end{bmatrix}$$

Then the computation of $\eta^5 X$ and $\eta^5 Y$ appears as

$$\begin{bmatrix} 1 \\ -1 \\ 5 \\ 2 \\ 0 \\ -7 \end{bmatrix} + 12 \begin{bmatrix} -1/12 \\ 1/12 \\ -5/12 \\ -2/12 \\ 1/12 \\ 7/12 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \qquad \text{and} \qquad \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 0 \\ 6 \end{bmatrix} + 0 \begin{bmatrix} -1/12 \\ 1/12 \\ -5/12 \\ -2/12 \\ 1/12 \\ 7/12 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 0 \\ 6 \end{bmatrix}$$

$$X_i + x_5 \eta^5 = \overline{X}_i \qquad (i \neq 5)$$

$$Y_i + y_5 \eta^5 = \overline{Y}_i = Y_i \qquad (i \neq 5)\,.$$

In the case of X we verify our original transformation. The case of Y shows that if the vector to be transformed has a zero in the position corresponding to the pivot of the η being applied then no element is changed because η has a zero multiplier.

It is necessary to call attention to just one more aspect of PFI. We note that if

$$E^p_m \cdot E^p_{m-1}{}^{-1} \ \ldots \ E^p_1{}^1 A = I \text{ then}$$

$$(E^p_m \cdot E^p_{m-1}{}^{-1} \ \ldots \ E^p_1{}^1) = A^{-1} .$$

The implication here is that normally we have a succession of ETA in sequence applied to a column of A (say A_j) to transform it to its representation in terms of the current basis.

3. Consequences of pivot sequence of ETA nonzero density

The basic procedure in developing the set of ETA representing the inverse of the basis is as follows:

a) Select a column of the basis not already used for pivoting
b) Transform the column by applying the current set of ETA in the order of generation.
c) Choose a pivot element for the transformed column in a row (p) where no column has pivoted previously.
d) Form an η^p from the transformed column and add to the ETA set.
e) Repeat steps a through d until all columns have been pivoted.

It is understood that column selection may be done on a random basis and within the rules we may pivot on any nonzero because the effect of these actions is to merely give us a different permutation of I and this is acceptable.

If we do have such freedom of choice in column and pivot row selection, does the sequence in which we use columns and rows have any impact on ETA nonzero density? A small example will show that it does. Take for example the matrix

	$A.1$	$A.2$	$A.3$
$A1.$	1	0	2
$A2.$	0	0	6
$A3.$	2	4	2

If we were to use the pivot sequence

Col	Row
$A.3$	$A3.$
$A.2$	$A1.$
$A.1$	$A2.$

ETA set would be $\eta^3 = \begin{bmatrix} -1 \\ -3 \\ 1/2 \end{bmatrix}$ $\eta^1 = \begin{bmatrix} -1/4 \\ -3 \\ 1/2 \end{bmatrix}$ $\eta^2 = \begin{bmatrix} 1/12 \\ -1/3 \\ 1/6 \end{bmatrix}$

On the other hand if our pivot sequence is

Col	Row
$A.3$	$A2.$
$A.1$	$A1.$
$A.2$	$A3.$

then the ETA set would be $\eta^2 = \begin{bmatrix} -1/3 \\ 1/6 \\ -1/3 \end{bmatrix}$ $\eta^1 = \begin{bmatrix} 1 \\ 0 \\ -2 \end{bmatrix}$ $\eta^3 = \begin{bmatrix} 0 \\ 0 \\ 1/4 \end{bmatrix}$

To verify the equivalence of both sets let us apply each set to an arbitrary vector e.

η^3 η^1 η^2

$\begin{bmatrix} -1 \\ -3 \\ 1/2 \end{bmatrix} \begin{bmatrix} -1/4 \\ -3 \\ 1/2 \end{bmatrix} \begin{bmatrix} 1/12 \\ -1/3 \\ 1/6 \end{bmatrix}$ applied to $\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ yields $\begin{bmatrix} -1/6 \\ 2/3 \\ 1/6 \end{bmatrix}$ $\begin{matrix} A.2 \\ A.1 \\ A.3 \end{matrix}$

η^2 η^1 η^3

$\begin{bmatrix} -1/3 \\ 1/6 \\ -1/3 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ -2 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1/4 \end{bmatrix}$ applied to $\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ yields $\begin{bmatrix} 2/3 \\ 1/6 \\ -1/6 \end{bmatrix}$ $\begin{matrix} A.1 \\ A.3 \\ A.2 \end{matrix}$

We note that the values associated with a column are the same in both cases – only a permutation has taken place. We also observe that for the first sequence of ETA we have 100% density of nonzeros while the second sequence has 66.6% density which is exactly the density of the original matrix.

If we consider how this reduction in nonzeros comes about it becomes apparent that we took advantage of zero multipliers. Indeed we see that if we pivot in sequence down the diagonal of a matrix of the form $L + D$ (lower triangle plus nonzeros on the diagonal) then no new nonzeros are added to ETA because each pivot element has nothing but

zeros to its right and hence zero multipliers. Therefore, as in fig. 1,

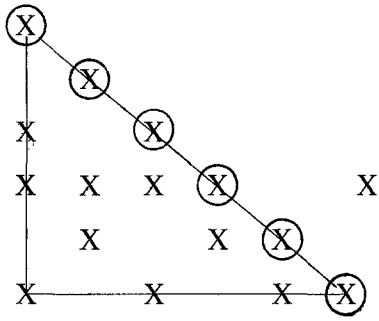

Fig. 1.

if we can select the sequence of columns and pivot rows in an arbitrary sparse matrix in such a way as to find this structure we will save all the work of transforming each vector by the existing set of ETA. The question now becomes – how can we find this structure or how close to this structure can we get?

4. The preassigned pivot procedure

We immediately notice the following:

a) The first or topmost pivot is characterized by the fact that this is the only nonzero element in the row.

b) The last or lowest pivot is characterized by the fact that this is the only nonzero element in the column.

Therefore a recursive technique with the following logic is suggested:

1. Initialization:

All references to columns and rows are made via indices.

Let $\bar{\mu}$ = the number of pivotal positions required.

a) Set the following parameters:

(i) $\mu = \bar{\mu}$ to be used for indexing the backward triangle

(ii) $\nu = 1$ to index the forward triangle

(iii) $L = 0$ to index spikes

b) Allocate space for the following vectors of dimension $\bar{\mu}$:

(i) C sequence of column pivots

(ii) R sequence of row pivots

(iii) I column counts

(iv) J row counts

(v) S spikes

2. Compute the number of nonzeros in each column and each row and record in I and J respectively.
3. Scan column counts (array I)
   a) if there is no count equal to one, go to 4.
   b) if a count of one is found, then
      (i) record the column (index) in C_μ
      (ii) record the row (index) of the nonzero element in R_μ
      (iii) reduce the column counts (I) of those columns containing nonzero elements in the row recorded in R_μ
      (iv) mark the column and row of C_μ and R_μ as no longer available.
      (v) set $\mu = \mu - 1$
      (vi) if $\mu = 0$, go to 5; otherwise go to start of 3.
4. Scan row counts (array J but exclude rows marked unavailable)
   a) if there is no count equal to one, go to 5
   b) if a count of one is found, then
      (i) record the row (index) in R_ν
      (ii) record the column (index) containing the nonzero element in C_ν
      (iii) reduce the row counts (J) of those rows for which the column recorded in C_ν has nonzeros
      (iv) mark the column and row of C_ν and R_ν as no longer available
      (v) set $\nu = \nu + 1$
      (vi) if $\nu > \mu$, go to 5.
      (vii) if $L = 0$, go to start of 4; otherwise go to 9.
5. Test for completion or start of bump.
   Note that at step 5 we have one of two possible situations: either the matrix has been completely triangularized or at some point each remaining row and column has two or more nonzero elements. In the latter case the number of vectors for which we must still find pivots is $\mu - \nu + 1$.

To bring the concept into better focus we illustrate with a matrix (fig. 2) which we will latter carry on to completion.

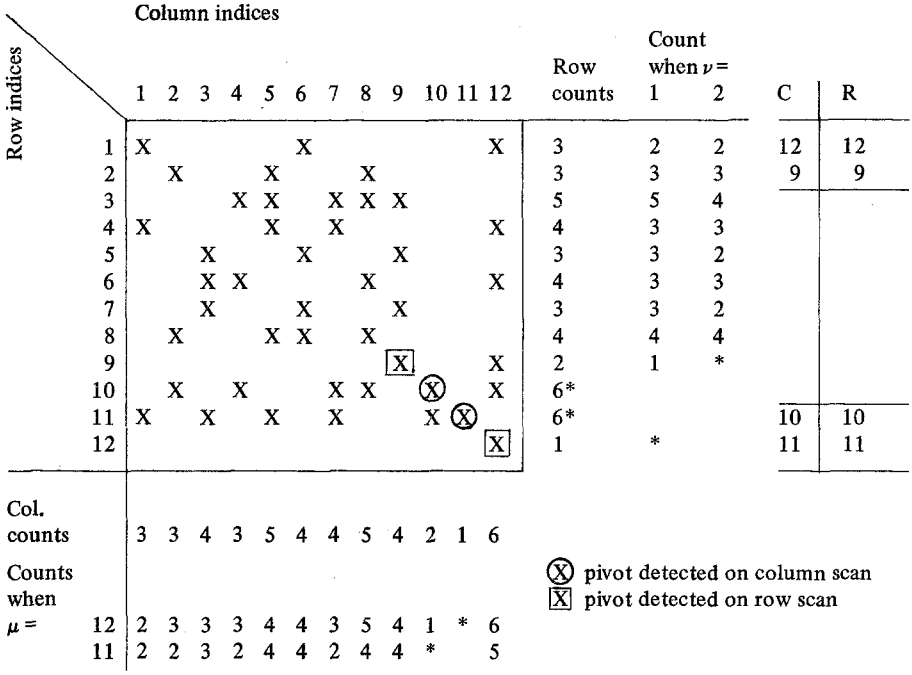

Fig. 2.

We have now reached a point where all row and column counts are two or more. At this point $\mu - \nu + 1 = 8$ so we know that we still have eight columns to pivot and these columns are referred to as the "bump". The reason for this designation becomes clear if we display the matrix in its rearranged form as in fig. 3.

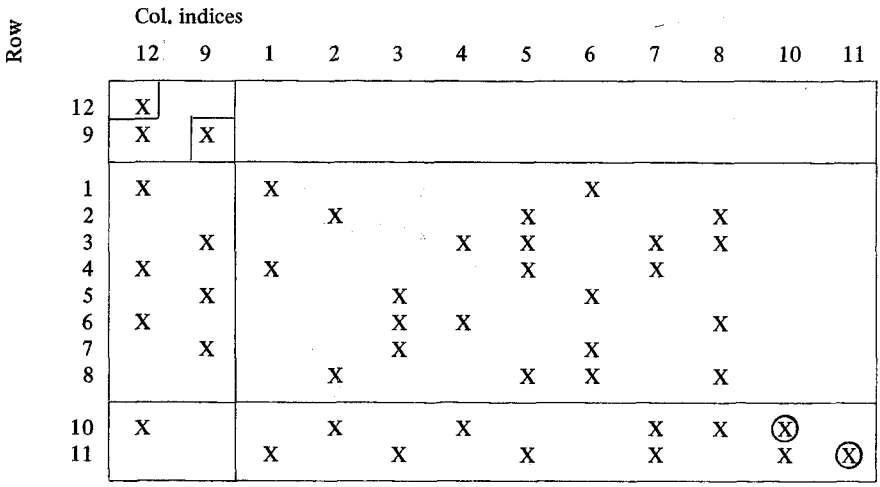

Fig. 3.

In general we can present any matrix at this stage (if it has not been completely triangularized) as in fig. 4,

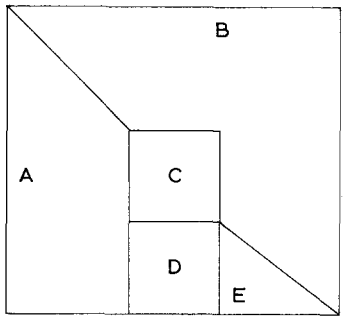

Fig. 4.

where in sections A and E we have found pivots on the main diagonal and all other nonzero elements are below. Section B is all zero. The "bump" section C has at least two elements in each row and column within the section. Section D contains the elements below section C.

It has been found that the greatest amount of work and the buildup of nonzero elements in ETA occur as a consequence of the technique used in processing sections C and D. This can be understood immediately when we see that sections A and E have zero multipliers as we proceed down the diagonal. Therefore the major problem at hand is how to cope with the nonzero buildup in sections C and D.

A method of reducing such nonzero buildup would be to break section C into two or more bumps, so that the nonzero buildup would be limited to these smaller bumps. Such bumps we call external bumps. Fig. 5 shows sections C and D broken up into three external bumps, sections G, H, and K.

Section F, like section B, is all zero. We can treat sections L and M just like sections A and E.

Remembering the success of the zero multiplier technique in sections A, L, M, and E, we would like to arrange the rows and columns of each external bump (sections G, H, and K) so as to have as many columns as possible with only zeros above the main diagonal. In fig. 6, everything above the main diagonal is zero except where the vertical solid lines are shown.

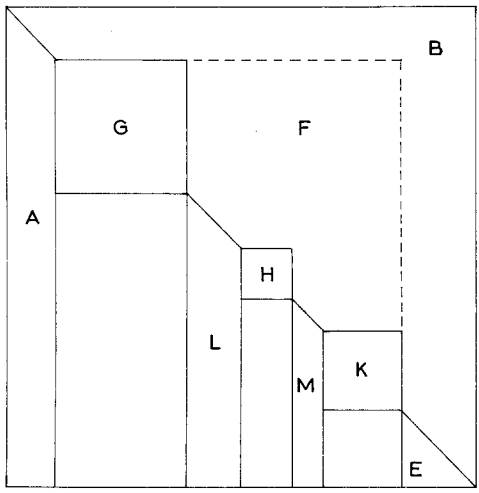

Fig. 5.

The columns of the matrix which contain these vertical lines are called spikes. The external bumps G, H, and K have been indicated by the dotted lines. The right hand edge of each external bump is always a spike with a nonzero in the upper position. This can always be achieved by a suitable arrangement of rows and columns within an external

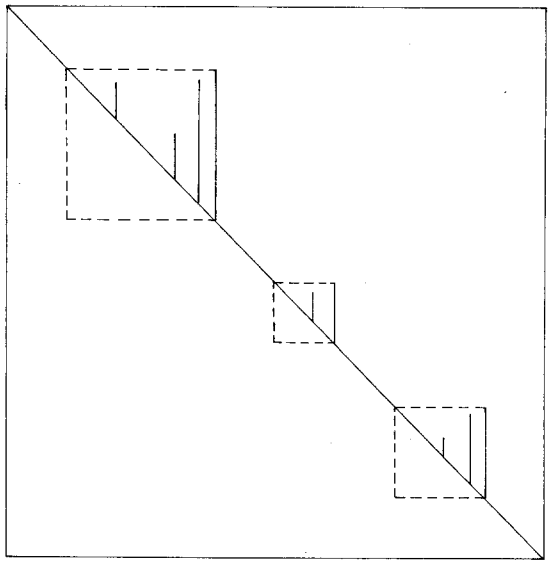

Fig. 6.

bump. The other spikes (which are the right hand edges of internal bumps) should be as short as possible so that they will be transformed with the smallest subset of ETA.

The remainder of this paper is devoted to the techniques we developed to find the spikes and external bumps.

At this time we know that no matter which column we choose to pivot next, there will be at least one additional column that has an element in the pivot row and thus will become a spike. Therefore, instead of selecting the column to pivot we address ourselves to the problem of spike selection. It soon becomes evident that the best vector to select as a spike is one which

a) when its effect is removed from the row counts, will create a maximum number of row counts of unity (as vectors selected on the basis of unity row counts need not be updated), and
b) will be able to pivot as soon as possible (thereby being updated by the smallest subset of ETA).

It is not always possible to achieve these goals with one spike, so to be realistic, we state our goal to be generally – select the spike in such a way as to make as many small row counts as possible become even smaller. In order to do this we will need a tally function defined as follows:

$t_k(n)$ = the number of nonzeros that column n has in rows whose row counts are less than or equal to k.

Thus: $t_2(6) = 3$ means that column 6 has 3 elements in rows with a row of two or less.

We can now give the rules for spike selection. We shall also consider that these rules are a continuation of the technique already given for processing the forward and backward triangles of the matrix (section A and E of fig. 5). Therefore we continue at step 5 of the procedure as follows:

5. Test for completion or start of bump:
   a) if $\nu > \mu$, go to 14 (END)
   b) if $\nu \leq \mu$, then
      (i) set k = minimum nonzero row count (J)
      (ii) define N as the set of vectors not found in C or S
6. Compute the tally functions $t_k(n)$ for all $n \in N$
7. Spike decisions:
   a) if maximum $t_k(n) > 1$, then
      (i) if unique, label it n_s, go to 8

(ii) if not unique, select n_s from the maximum tally columns as the column with the greatest column count or if these are equal make an arbitrary selection from among them and go to 8.

b) if maximum $t_k(n) = 1$, then

(i) define N to be the set of vectors for which $t_k(n) = 1$

(ii) redefine k = min. row count greater than previous value of k

(iii) go to 6.

8. Spike array entry:

a) set $L = L + 1$

b) enter n_s into S_L

c) reduce row counts of J in all rows for which n_s contains nonzeros.

9. Scan row counts (J) for path decisions

a) if min $J > 1$, go to 5.b.i. (another spike)

b) if min $J = 1$ and is unique, or if $L = 0$, go to 4.b.

Applying these rules to the bump of our sample problem we have:

C | R	S		1	2	3	4	5	6	7	8	J	
	6	1	X					X			2	$k = 2$
		2		X			X			X	3	$N = \{1, 2, 3, 4, 5, 6, 7, 8\}$
	L=1	3				X	X		X	X	4	$t_2(1) = 1$ $t_2(5) = 0$
		4	X				X		X		3	$t_2(1) = 0$ $t_2(6) = 3$
		5			X			X			2	$t_2(3) = 2$ $t_2(7) = 0$
		6			X	X				X	3	$t_2(4) = 0$ $t_2(8) = 0$
		7			X			X			2	$n_s = 6$
		8		X			X	X		X	4	
		I	2	2	3	2	4	4	2	4		

Fig. 7.

If we now reduce the row counts in J according to the nonzeros of column 6 we have:

	1	2	3	4	5	7	8	6	J
1	X							X	1
2		X			X		X		3
3				X	X	X	X		4
4	X				X	X			3
5			X					X	1
6			X	X			X		3
7			X					X	1
8		X			X		X	X	3
I	2	2	3	2	4	2	4		

Fig. 8.

We note that the minimum nonzero $J = 1$ but it is not unique. This motivates us to think that if we can find a single vector which will reduce two or more counts of J to zero then we would have an opportunity to assign a pivot row to the spike. In order to find such a vector, it turns out that the tally function is the best tool to use. Therefore we may continue with the statement of the algorithm rules at step 9.c. in the following manner:

9. c) if min. $J = 1$ and is not unique, then
   (i) set $k = 1$
   (ii) define N as the set of vectors not listed in C or S
10. Compute the tally function $t_k(n)$ for all $n \in$ N.
11. Vector and pivot selection
   a) if maximum $t_k(n) > 1$ then
      (i) if unique, label it n_p, go to 11.c.
      (ii) if not unique, select n_p from the maximum tally columns as the column with the greatest column count, or if these are equal make an arbitrary selection from among them and go to 11.c.
   b) if maximum $t_k(n) = 1$, then
      (i) define N to be the set of vectors for which $t_k(n) = 1$.
      (ii) redefine k = minimum row count greater than previous value of k.
      (iii) go to step 10.
   c) define $q = t_1(n_p)$

12. Make pivot list entry.
    a) record n_p in C_ν
    b) record in R_ν a row of n_p which is nonzero and for which $J = 1$.
    c) mark n_p and its pivot row as no longer available
    d) reduce the row counts in J for all rows that have nonzeros in n_p
    e) set $\nu = \nu + 1$
    f) if $\nu > \mu$, go to 14
    g) set $q = q - 1$
    h) if $q = 0$, go to 9; otherwise go to 13
13. Enter spike into basis list.
    a) record the vector of S_L in C_ν
    b) record in R_ν a row that was reduced to zero in J
    c) set $L = L - 1$
    d) go to 12.e.
14. End.

We can now finish the example according to the algorithm in the following manner:

C	R	S		1	2	3	4	5	7	8	6	J	
		6	1	X							X	1	$k = 1$
			2		X			X		X		3	$N = \{1, 2, 3, 4, 5, 7, 8\}$
		L = 1	3				X	X	X	X		4	$t_1(1) = 1$ $t_1(5) = 0$
			4	X				X	X			3	$t_1(2) = 0$ $t_1(7) = 0$
			5			X					X	1	$t_1(3) = 2$ $t_1(8) = 0$
			6			X	X			X		3	$t_1(4) = 0$
			7			X					X	1	
			8		X			X		X	X	3	$n_p = 3$ $q = 2$
			I	2	2	3	2	4	2	4			

Fig. 9.

Therefore vector number 3 is selected to pivot in either row 5 or row 7. We pivot vector 3 on row 5 and this leaves row 7 available as a pivot row for the spike, vector 6.

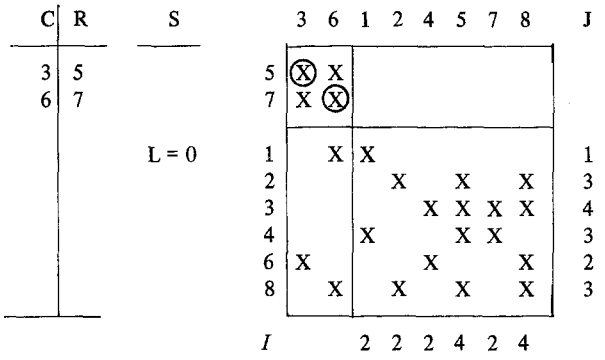

Fig. 10.

Now we have a unique minimum count of one in J, so rule 9.b returns us to forward triangle (section 4.b). As a result we have:

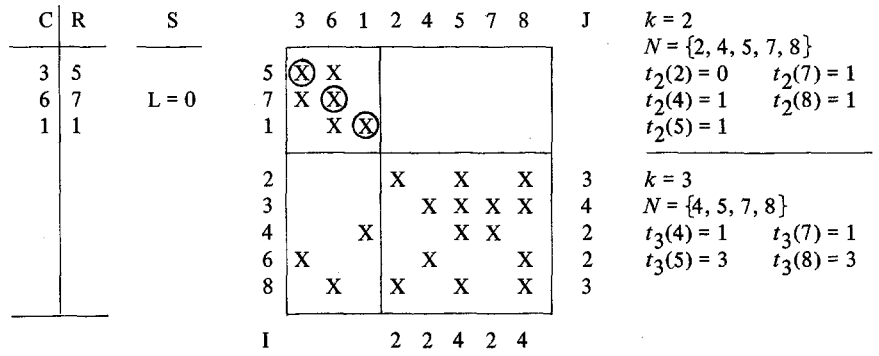

Fig. 11.

Here we have shown that the minimum row count in J is greater than unity so we must select a spike. The use of the tally function with $k = 2$ is not adequate to select the spike, so we raise k to a value of 3 and work with the reduced set N and find that vectors 5 and 8 are equally good in tally and in column counts. If we select vector 5 as the spiké we display our matrix and carry on as follows:

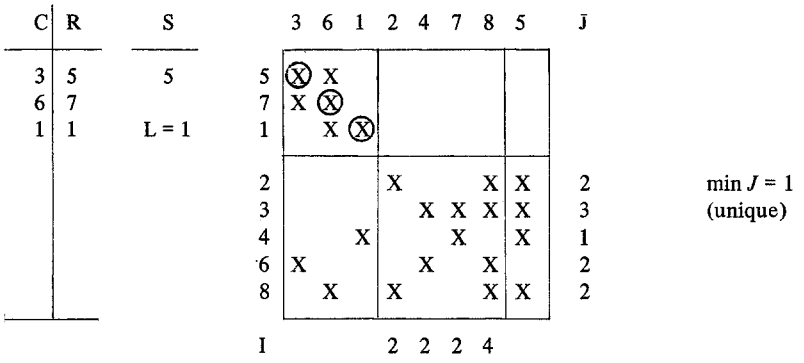

Fig. 12.

We see that the minimum J is one and is unique, so we can assign row 4 as the pivot row for column 7. We show this and continue as follows:

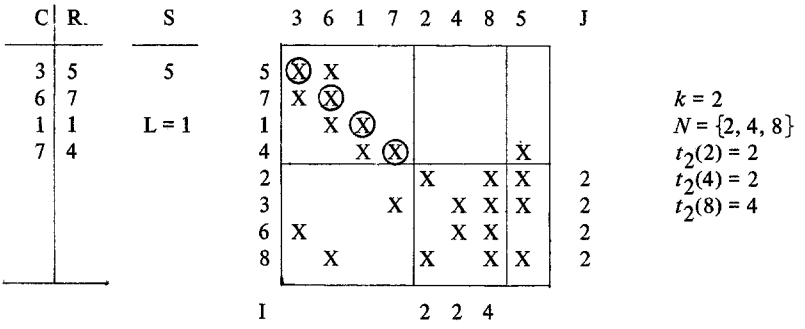

Fig. 13.

It is evident that column 8 is the best choice for the spike so it is entered in the S array with $L = 2$. This leads to the following tableau:

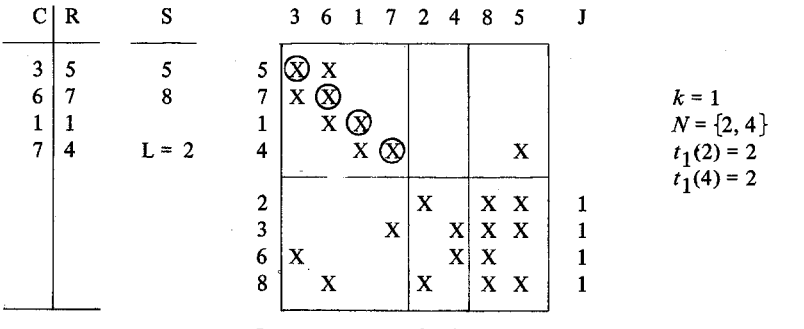

Fig. 14.

The tableau shows that vectors 2 and 4 are equally good for the basis at this time. Furthermore, either one will give q a value of two so the spike vector in position S_2 may follow either vector into the basis. This is illustrated by:

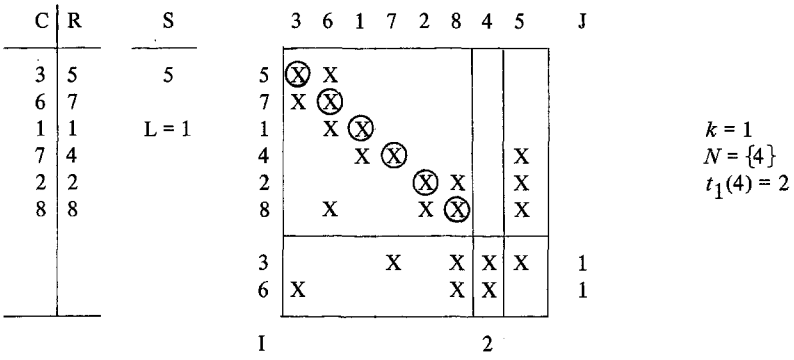

Fig. 15.

Vector 4 will be pivoted next and this generates a pivot row for the final spike, vector 5. The complete pivot sequence is now displayed as:

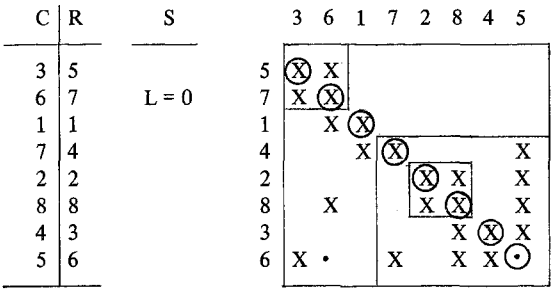

Fig. 16.

We have designated by a • the creation of a nonzero element in that position.

The complete structure is displayed as:

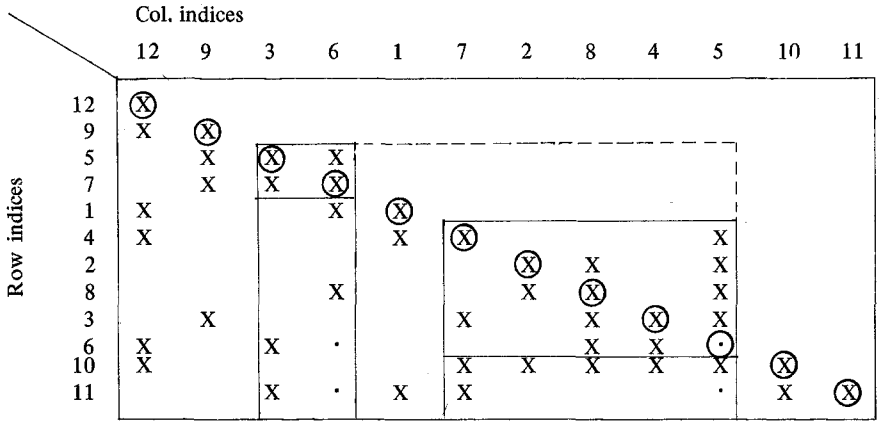

Fig. 17.

Notice that nonzeros are created only within spikes. However it is possible to prevent the creation of nonzeros outside the rows and columns of an external bump by partitioning the η into the rows within the external bump and those outside. Thus:

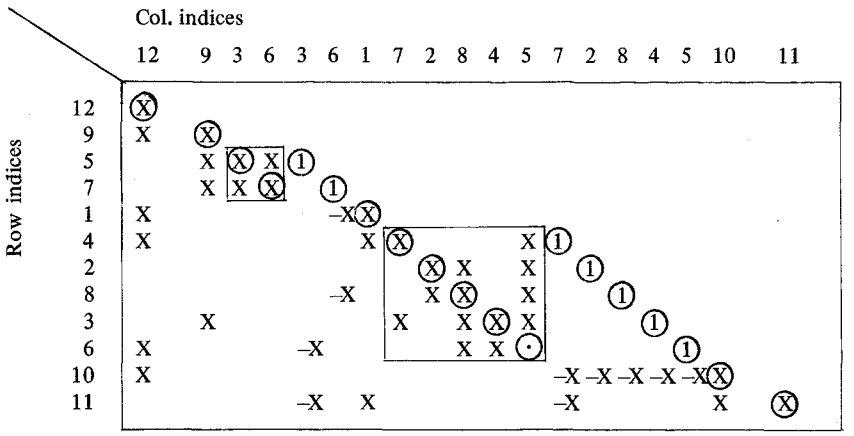

Fig. 18.

In the figure we have two external bumps, one of dimension 2 and one of dimension 5. By partitioning as suggested by E.M.L. Beale, we actually form two sets of η vectors, one for the rows in the bumps and one

for the full vector. However, we know that if the set of η for elements within the bump are applied to the full vector we will have unity in each position for which there was a pivot and zero elsewhere within the external bump. Knowing this we can take advantage of the situation and save the calculations by merely inserting the 1 where it is needed. The $-X$ represents the negatives of the original elements of those columns and rows and this is one of the easiest kinds of transformations to make. In the example shown there is no real savings to be realized by partitioning, for the number of unity elements created exceeds the number of new elements that would be created otherwise. In the general case, however, there are many more elements below the base line of the external bump, so the partition does prevent buildup of nonzeros. The procedure as presented is concerned only with the construction of the column and row pivot sequence. Tests may be added in steps 3 and 4 to detect singularities. During the scan of column counts (step 3), a column with a count of zero should be dropped from the basis. During the row count scan (step 4), the appearance of zero count in an unassigned row indicates a redundant constraint.

When the ETA are ultimately formed in the pivot list sequence, it is necessary to guard against pivoting on an element that is too small. However, if we assume that all matrix elements are pivotable (this can be arranged by column and row scaling), then it is obvious that all non-spike columns may pivot without difficulty. If the updated pivot element of a spike becomes too small to be used as a pivot, it becomes necessary to "swap" spikes among the set of spikes within the external bump. We assert that if a matrix is not singular, there exists an arrangement of the set of spikes within each external bump such that every spike within the bump may pivot upon an element of suitable size *. In practice we have found that very few spike interchanges are made per inversion.

5. Conclusions

The preassigned pivot procedure, while heuristic in nature, tends to give very good results in terms of speed and minimizing the nonzero buildup of ETA.

* A proof of this statement will be contained in a forthcoming paper.

This algorithm performs well for several reasons:

a) Having the pivots preassigned before any η is created, we know in what order the basis matrix will be needed, and so can sharply reduce basis matrix I/O.

b) We have carried triangularization of the matrix much farther than the old schemes, not just at the front and back of the matrix, but between the bumps and even within the bumps themselves. For all of these non-spike vectors, the η can be created directly from the basis vectors themselves, thus saving a considerable amount of work.

c) The few vectors that do need to be updated, i.e., the spikes, are usually updated only by a small subset of ETA, and so the updating goes quite fast.

Another feature of this algorithm is that the amount of arithmetic done is very small, and so the round-off error is smaller than with the older schemes.

Finally we would like to point out that since the nonzero buildup in ETA is substantially less than with the older schemes, the iterating routines run noticeably faster. And since the invert routine is faster, we cas afford the overhead of an invert more often and hence keep the ETA small. As a result, most problems solve in 1/2 to 2/3 the time previously required.

Acknowledgement

This work was essentially completed while the authors were employed by CEIR Professional Services Division of CDC, Washington.

References

[1] H.M. Markowitz, "The elimination form of the inverse and its application to linear programming," *Management Science* 3, No. 3 (1957) 255–269.

[2] D.V. Steward, "On an approach to techniques for the analysis of the structure of large systems of equations," *SIAM Review* 4, No. 4 (1962) 321–342.

[3] P. Wolfe and Leola Cutler, "Experiments in linear programming," *Recent advances in mathematical programming* (McGraw-Hill Book Co. Inc., New York, 1963) pp. 177–200.

[4] G. Zoutendijk, *Methods of feasible directions* (Elsevier Publishing Co., Amsterdam) pp. 52–55.

Appendix

Performance comparison between preassigned pivot procedure (MPS/III) and IBM (MPS/360) inversions

PROBLEM NUMBER		1	2	3	4	5
NUMBER OF ROWS		589	782	838	903	977
NUMBER OF BASIC STRUCTURALS		513	651	522	822	782
NUMBER OF BASIS NON ZEROS		4946	7976	5942	3883	4343
NUMBER OF ETA NON ZEROS	MPS/III	5734	9485	5833	4176	4780
	MPS/360	6543	19879	6212	8651	7806
DENSITY INCREASE	MPS/III	0.16	0.19	−0.02	0.08	0.10
	MPS/360	0.32	1.49	0.05	1.23	0.80
TIME TAKEN	MPS/III	0.14	0.38	0.22	0.21	0.28
	MPS/360	0.45	2.14	0.69	1.33	1.29
FACTOR OF SPEED IMPROVEMENT		3.2	5.6	3.1	6.3	4.6
NUMBER OF BUMPS	MPS/III	3	19	16	15	22
NUMBER OF SPIKES	MPS/III	23	108	24	32	68

Notes:
1) DENSITY INCREASE = (ETA NON ZEROS − BASIS NON ZEROS)/BASIS NON ZEROS
2) BASIS NON ZEROS includes the slacks for all logicals in the basis.
3) ETA NON ZEROS includes the slacks for only the logicals in the basis which are on greater than rows.
4) The MPS/360 invert used is the fast invert (i.e., XINVERT = 0) as provided by IBM in their version 2, mod 9 MPS/360 system.
5) For each problem, the MPS/III and the MPS/360 inverts were executed in the same job step of the same run so that the timing could be compared. A 360 model 50 and region sizes of 150K and 200K were used.