TR/13/88 December 1988

ALTERNATIVE METHODS FOR REPRESENTING THE INVERSE OF LINEAR PROGRAMMING BASIS MATRICES

Gautam Mitra and Mehrdad Tamiz

ALTERNATIVE METHODS FOR REPRESENTING THE INVERSE OF LINEAR PROGRAMMING BASIS MATRICES

Gautam Mitra

Brunel University

and

Mehrdad Tamiz†

Brunel University

†Dr Mehrdad Tamiz is supported by a collaborative research grant of the Science and Engineering Research Council (SERC) UK and Numerical Algorithms Group Limited (NAG), UK.

TABLE OF CONTENTS

0 Abstract

1. Introduction

2. Background and an Analysis of the Issues

3. Lower Block Triangular (LBT) Factorization and Basis Inverse Representation

4. Data Structures for Factorization.

5. A Comparison of Alternative Representations for Basis Inverse

6. Update of the Basis Inverse

7. Acknowledgments

8. References

TITLE:

Alternative Methods for Representing the Inverse of Linear Programming Basis Matrices

Gautam Mitra and Mehrdad Tamiz
Brunel University, U.K.

Methods for representing the inverse of Linear Programming (LP) basis matrices are closely related to techniques for solving a system of sparse unsymmetric linear equations by direct methods. It is now well accepted that for these problems the static process of reordering the matrix in the lower block triangular (LBT) form constitutes the initial step. We introduce a combined static and dynamic factorisation of a basis matrix and derive its inverse which we call the partial elimination form of the inverse (PEFI). This factorization takes advantage of the LBT structure and produces a sparser representation of the inverse than the elimination form of the inverse (EFI). In this we make use of the original columns (of the constraint matrix) which are in the basis. To represent the factored inverse it is, however, necessary to introduce special data structures which are used in the forward and the backward transformations (the two major algorithmic steps) of the simplex method. These correspond to solving a system of equations and solving a system of equations with the transposed matrix respectively. In this paper we compare the nonzero build up of PEFI with that of EFI. We have also investigated alternative methods for updating the basis inverse in the PEFI representation. The results of our experimental investigation are presented in this paper.

1. Introduction

Computer solution of large sparse systems of linear equations and the related problem of computing the inverse of the corresponding coefficient matrix, both efficiently and in an efficient (compact) form assumes a key role in the study of large systems which arise in many contexts [DFREID79], [DGEPLE82]. Commercial exploitation of the equation solving technologies for such systems is most widespread in the field of linear programming. This is because formulation of large planning and scheduling problems and also their solutions have been well investigated since the late sixties. Also solution methods for LPs have continued to keep pace with developments in the field of algorithms, software and computer hardware. Series of special conferences on the topic of sparse matrices and sparse equation solving methods were organised during the 60s and all through the 70s [WILLBY69], [REIDJK71], [ROSWIL72], [BUNROS76], [DUFSTW79], and there is a wide body of literature which cover these developments. Research covering algorithms, data structures, software implementation and exploitation of machine architecture

Our research interests concern the solution of large Linear Programming problems robustly and efficiently [MTTAMZ88]. At the heart of such a system lies suitable basis inversion and basis inverse update procedures. In this paper we set out our arguements for the choice of the inversion and update procedures; we describe the mathematical methods and the related implementation issues and finally present some preliminary results of our investigations. The rest of the paper is organised as follows. In section 2 we expand on the background of the developments and present our analysis of the issues. In section 3 we discuss why the lower block triangular (LBT) form is a desirable structure, and the corresponding factored form of the inverse. In section 4 we outline the data structures which we have adopted to represent the factorisation and we describe how it is used in the solution process. In section 5 we compare the elimination form of the inverse with the partial elimination form of the inverse (PEFI) as derived by us. In section 6 the alternative ways of updating representation are presented. Section 7 contains a discussion and concluding remarks based on our findings.

2. Background and an Analysis of the Issues

The methods of computing and updating the inverse basis matrices of large sparse linear programming problems are dominated by the following major considerations. (a) The nonzero structure of the matrix needs to be analyzed and the rows and columns of the matrix reordered: ANALYZE phase. (b) Pivots have to be chosen for (Gaussian) elimination and alternative methods of factorization of the matrix have to be considered: FACTOR phase. (c) A method has to be derived whereby the inverse can be used in the solution process implicit in the "backward and forward transformations": SOLVE procedures. (d) Suitable data structures have to be designed to represent the original sparse matrix and the sparse basis inverse. (e) The implications of the algorithms and the data structures on the main memory of the computer and the architecture of the processor have to be taken into account. (f) In addition to the efficient inverse representation an efficient update procedure has to be designed taking into account (b), (d), (e) above. We refer the readers to papers [BARGOL69], [FORTOL72], [HELRAR71], [HELRAR72], [REIDJK76], [REIDJK82], [GMSAWR84] which have addressed, expanded and dealt with most of these issues. The point (e) concerning implications of these algorithms on the current generation of multiprocessing architecture has been barely discussed in the literature.

(e) ANALYZE phase

Given a nonsingular matrix B of order m the reordering algorithms make use of the corresponding graph of the adjacency matrix derived from the nonzero elements. In the symmetric case the aim is to derive specifications for symmetric row and column

interchanges given by permutation matrices P, P^T respectively such that we obtain a reordered matrix B, $B = PBP^T$ which is presented as compactly as possible in band diagonal form. The heuristic methods exploit the underlying undirected graph and make use of the degrees of the vertices. In the case of unsymmetric matrices, the general situation with basis matrices, it is well established [DARMIT81], [DUFFIS84] that LBT is the most desirable form. This LBT reordering is achieved by considering the underlying directed graph and applying the following algorithms.

- Maximal Matching

We can apply the maximum matching algorithm of Hall [HALLML56] whereby the matrix B is reordered to B* which has a zero-free diagonal. As a result B* = BQ; the permutation matrix Q which specifies the reordering is derived by this algorithm.

- Finding the Strong Components

To the matrix B* which has a zero-free diagonal the celebrated algorithm of Tarjan can be applied to obtain the strong components (the diagonal blocks of the matrix) of the directed graph. This leads to the derivation of orderings given by the permutation matrices P, P^T whereby the original matrix B can be reordered into the lower block triangular form B

$B = PB^*P^T$ or $B = P(BQ)P^T$. We note that for the symmetric case the problem of bandwidth minimisation is NP-complete [PAPDIM76]. Hence only heuristic methods are applied. In contrast the above mentioned LBT algorithms have the complexity of low order polynomials [DUFFIS84], [DARBYK80]. Beale [BEALEM84] conjectured that a band form for the unsymmetric case may be equally attractive as the LBT form. Although we have done some preliminary work on this topic [JUMITZ87] this has not been properly followed up and we are not considering the issue of symmetric matrices and the band form any further in this paper. It is interesting to note that Hellerman and Rarick [HELRAR71] did not make use of Tarjan's work as these results were not known at that time. They also combined the ANALYZE and at least a part of the FACTOR phase as they preassigned pivot positions; the implications of this are discussed in the next subsection,

(b) FACTOR phase

The main motivation of this phase is to compute a sparse yet accurate LU decomposition of the matrix B. If the LU decomposition is computed in the product form of the L and U transformation matrices then this reduces the nonzero growth. If pivot choice is made dynamically during factorization with a view to control accuracy then this can lead to a stable solution. Hellerman and Raricks P^4 algorithm finds spikes which are postponed for pivoting and constructs pivot agenda based entirely on a static analysis. The main aim is to minimise fill in. Unfortunately, however, it is necessary to modify the algorithm to deal with all possible contingencies otherwise the method is known [LINMAH77], [ERGRLP85] to lead to structural singularity as well as unstable solutions. Erisman et al have studied the comparative performance of a modified and improved P^5 (precautionary partitioned preassigned pivot procedure) with the modified Markowitz procedure with threshold tolerance due to Reid. The authors conclude that not only is the latter superior in non-LP applications, even in LP applications, the latter is equally good if not superior, taking into account accuracy and robustness. It is well known that the choice of the largest admissible nonzero entry at each step of Gaussian elimination can provide accurate LU decomposition but this strategy will result in considerable fill-in. Tomlin [TOMLIN72] suggested a relative tolerance of 0.01 of maximum column entry based on experimental evidence of processing LP problems. We have found Reid's recommendation of a threshold tolerance with restricted Markowitz merit criteria very

effective indeed [DEREID86], According to Reid at the k^{th} elimination step pivots are chosen by the criteria.

$$\min_{i,j} \quad (r_i - 1)\ (c_j - 1) \tag{2.1}$$

$$\text{and} \quad |b^k_{ij}| > u\ (\max_{\ell} |b^k_{\ell j}|) \quad \ldots\ 0 < u < 1$$

In (2.1) b^k_{ij} denotes the $(ij)^{th}$ entry in the remaining matrix at the k^{th} elimination step, r_i, c_j are row and column nonzero counts of row(i), column(j), and u is a threshold tolerance.

Thus the pivot chosen is b^k_{pq} such that $|\frac{b^k_{pq}}{u}|$ is greater than the largest absolute entry of the nonzero elements available for pivot in the column q of the matrix at the k^{th} step. Following Duff Erisman and Reid's suggestion we use a near minimum as opposed to exact minimum merit count and a threshold value of u = 0.1 which leads to fairly good results both for sparsity and accuracy. The data structures adopted by us are briefly discussed in section 4.

(e) The SOLVE procedures

Let e denote an m component vector with ± 1 as one or more of its components (e is known as the form vector). Let a_i denote a column of the A matrix and Π the vector of shadow prices. There are two well known steps of the revised simplex, namely backward Transformation and Forward Transformation, [BEALEM68], [MITRAG76], which make use of the B^{-1} representation. These transformation steps can be expressed as two equivalent equation solving steps given by the relations

$$eB^{-1} = \Pi \quad \text{or} \quad \Pi = (B^{-1})^T e \tag{2.2}$$

and

$$\bar{a}_j = B^{-1} a_j \tag{2.3}$$

Having computed the inverse in the factored form it becomes necessary to compute the solutions shown in (2.2) (2.3) making use of the data structures and the factored inverse representation.

(d) Choice of Data Structure

In the sixties and seventies in main frame as well as mini computers the main memory modules (also known as core store) were very expensive and for an LP system to work over a range of machines with varying memory sizes both the A matrix and the B^{-1} matrix were stored in backing magnetic stores in the form of sequential files. The main memory is no longer at a premium even for desk top machines. Yet it is still necessary to design highly compact data structures to represent the A and B^{-1} matrices. There are well known storage schemes for sparse and supersparse matrices [KALANJ71], [DARMIT83], [MITAMZ89]. Since we wish to have both A and B^{-1} memory resident it is to our advantage to use as much of A as possible to represent B^{-1}. We believe that although it might lead to some slowing down of the processing speed the ability to accommodate larger models in memory considerably enhances the upward limit of the resulting system. We take up this issue again in section 5 where we discuss some statistics concerning storage.

(e) Computer Memory and Processor Architecture

We have not studied in depth the implications of different parallel processing architectures and memory couplings on different LP and linear equation solving algorithms. We are, however, conscious of the strong trends in the design of algorithms which require transfer of serial algorithms for such computation intensive tasks to parallel machines. The main reason for setting this item for consideration is that it is a factor that has to be taken into account in the redesign of algorithms which must cover such aspects as memory coupling, data structure design and process synchronisation steps [HOCJES88].

(f) Update Procedures

Bartels and Golub [BARGOL69] developed a basis inverse update procedure which requires the original basis to be factored in the LU form. Their work and subsequent studies [REIDJK82] demonstrated that this was superior to product form update in all considerations of sparsity, stability and speed. Forrest and Tomlin [FORTOL72] introduced a clever variation of this update procedure which is better than the latter in speed and sparsity but is poorer on a stability consideration. This issue and its close relationship to the original problem of computing a sparse yet stable inverse was well addressed by Reid who produced the celebrated LAO5 subroutines with modified Bartels Golub update. These subroutines have been widely quoted in many studies and have been adopted as the main compute engine in successful LP systems [MARSTN81], [SMTTHB87]. Gill et al [GMSAWR84] have also considered this issue in considerable depth. They put forward a cogent argument (for studying this problem in its own right) which can be stated as follows. Considerable progress continues to be made in sparse equation solving methods which may be looked upon as "blackbox" procedures. Not all of these provide iterative update procedure. Hence methods for sparse and stable updates to work with sparse and stable solvers can be of considerable value. They have described how Bisschop and Meeraus's work [BISMER77] interpreted as Schur complement update can be exploited towards this purpose. We have found these arguments very compelling indeed. We describe in section 6 the mathematics of this update procedure as adopted by us and the framework and findings of our experiment.

3. Lower Block Triangular (LET) Factorization and Basis Inverse Representation

After reordering the matrix B into the LBT form B we can factorize it in the LU form and derive the elimination form of the inverse (EFI) as it is derived by the Gaussian elimination operations. Alternatively we can first factorise the matrix and then apply the elimination operation only partially to the diagonal blocks of the matrix B and call this the partial elimination form of the inverse (PEFI) [MITRAG73], [DARBYK80]. Erisman, Duff and Reid [ERGRLP85], [DEREID86] call the same representation the reducible factors of the original (reducible) matrix. If we do not take into account possible cancellation of nonzero elements to zero values then it is easy to show that PEFI will always produce a sparser inverse than EFI. Our computational experience ([DARBYK80], also see next section) bears this out and Erisman et al, as above, make the same statement. We illustrate this point by considering a reordered B matrix with two block diagonal matrices and four partitions. Consider the matrix B in the reducible form

$$B = \begin{bmatrix} L_1 & & & \\ B_{21} & B_{22} & & \\ B_{31} & B_{32} & B_{33} & \\ B_{41} & B_{42} & B_{43} & L_{44} \end{bmatrix} \qquad (3.1)$$

It is easily seen that the matrix can be expressed in a factored form as

$$B = \begin{bmatrix} L_{11} & & & \\ B_{21} & I & & \\ B_{31} & & I & \\ B_{41} & & & I \end{bmatrix} \times \begin{bmatrix} I & & & \\ & B_{22} & & \\ & & I & \\ & & & I \end{bmatrix} \times \begin{bmatrix} I & & & \\ & I & & \\ & B_{32} & I & \\ & B_{42} & & I \end{bmatrix} \times$$

(3.2)

$$\begin{bmatrix} I & & & \\ & I & & \\ & & B_{33} & \\ & & & I \end{bmatrix} \times \begin{bmatrix} I & & & \\ & I & & \\ & & I & \\ & & B_{43} & I \end{bmatrix} \times \begin{bmatrix} I & & & \\ & I & & \\ & & I & \\ & & & L_{44} \end{bmatrix}$$

The submatrices B_{22}, B_{33} can be expressed in the elimination form as $B_{22} = L_{22}U_{22}$, $B_{33} = L_{33}U_{33}$. This leads to the further factored form

$$B = \begin{bmatrix} L_{11} & & & \\ B_{21} & I & & \\ B_{31} & & I & \\ B_{41} & & & I \end{bmatrix} \times \begin{bmatrix} I & & & \\ & L_{22} & & \\ & & I & \\ & & & I \end{bmatrix} \times \begin{bmatrix} I & & & \\ & U_{22} & & \\ & & I & \\ & & & I \end{bmatrix} \times \begin{bmatrix} I & & & \\ & I & & \\ & B_{22} & I & \\ & B_{42} & & I \end{bmatrix}$$

(3.3)

$$\times \begin{bmatrix} I & & & \\ & I & & \\ & & L_{33} & \\ & & & I \end{bmatrix} \begin{bmatrix} I & & & \\ & I & & \\ & & U_{33} & \\ & & & I \end{bmatrix} \begin{bmatrix} I & & & \\ & I & & \\ & & I & \\ & B_{43} & & I \end{bmatrix} \begin{bmatrix} I & & & \\ & I & & \\ & & I & \\ & & & L_{44} \end{bmatrix}$$

We note that the PEFI can be easily derived from the above representation and only nonzero build up takes place in the L_{22}, U_{22}, L_{33}, U_{33} matrices as obtained by the elimination operations.

If a full EFI was derived B - LU then there will be further nonzero build up in the position of the blocks occupied b $\left|\frac{B_{32}}{B_{42}}\right|$ $|B_{43}|$ whereas in the PEFI no non zero build up takes place in these positions. In the MA28 subroutines distributed by Harwell [DEREID86] this can be specified as an option and the corresponding reducible factorization obtained. The apparent drawback of this method is that Bartels and Golub's update procedure cannot be applied to this form. We consider this issue further in section 6.

4. Data Structures for Factorization

The Analyze phase only makes use of static data structres. At the end of reordering, these data areas are discarded and they have no influence on the performance of the iterative steps of the simplex. For a description of these data structures we refer the readers to [DARBYK80], [TAMIZM86], [DEREID86]. In contrast the data structures used in the FACTOR phase require dynamic update, they also determine the overall usage of main memory and influence the processing speed in the simplex steps.

There are altogether three data structures which have important bearing in the factorization phase. These are ETA representation, LBT structure (sub-bump sequence) indicator, and structure for Markowitz pivot selection.

- Eta Representation and Sub Bump Sequence

In general we can subdivide these in following groups: Unit and non unit etas. Unit etas may or may not take up any space depending on implementation strategy but non unit etas can be further subdivided into direct and indirect etas. The indirect etas are in turn subdivided into full and split columns of the orginal A-matrix. These eta categories are shown in hierarchical form in diagram 4.1.

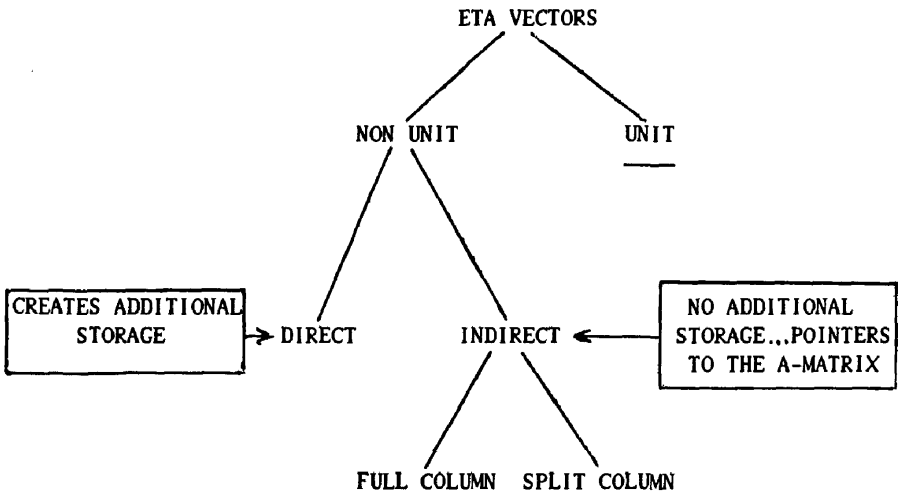

Diagram 4.1

The sturctures of these Etas are shown in diagram 4.2. Last and next provide forward and backward linking information NONZER and MROPIV indicate nonzeros and pivot rows respectively. They all occupy 2 bytes - 16 bits. The row indices I_1, I_2 etc occupy 2 bytes, whereas pivot value (PIVVAL) and elements VAL_1, VAL_2 are in double precision and occupy 8 bytes. ICOLAD is a 4 byte integer pointing to an A-matrix column. IBLOCN indicates the sub bump with which the split eta is connected. The A-matrix columns themselves make up the indirect etas. Their storage is shown in diagram 4.3.

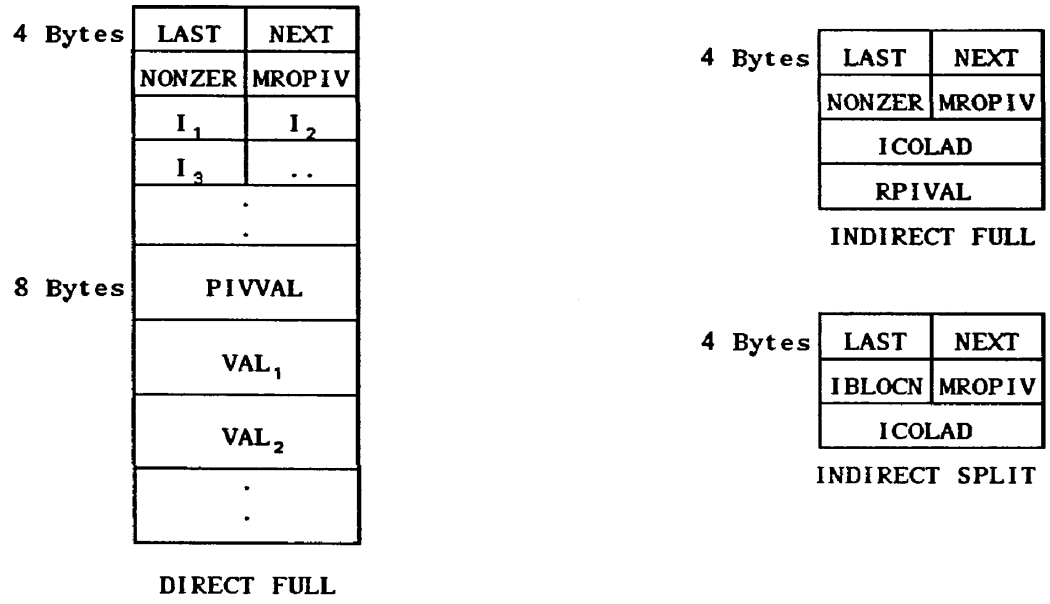

Diagram 4.2

We use the supersparse representation of Kalan [KALANJ71] and initially construct a unique element pool for the nonzero elements of the columns. As most real-life LP models have a prevalence of plus and minus ones the data structure is designed to store these values implicitly. A typical packed column format is shown in diagram 4.3. Where column header contains information regarding the bound and a pointer to the bound value. $I_,$, I_2, I_3 are the row numbers of the nonzero coefficients. P_1 and P_2 are the pointers to the unique element pool. I_3 inidcates a row number of 9 and the corresponding row coefficient has a value of +1.

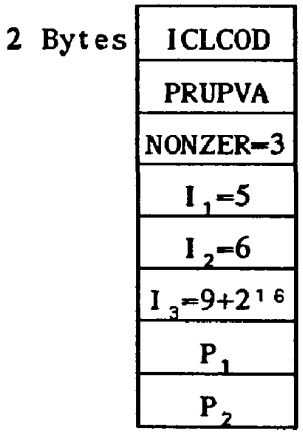

Diagram 4.3

Finally we note that a split eta which corresponds to the factored partial column of the A-matrix has a unit pivot and its entries are the original column entries less those entries which are marked in the sub bump indicators. Thus as shown in diagram 4.4 rows in a sub bump sequence are indicated by sub bump number. The column a_k belongs to this

sub bump ISB = 3. Hence the split part of the column does not include the entries with these row number

.

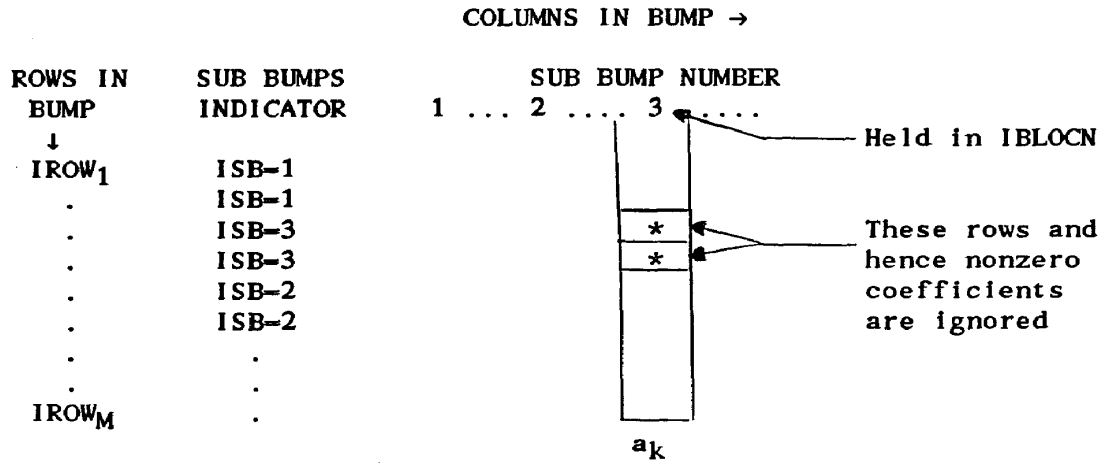

Diagram 4.4

This information is used during the forward and backward transformation processes.

- Data Structures for Markowitz Pivoting

The pivot choice strategy for the dynamic factorization of the sub bump based on merit count is already described in section 2.1. We have followed the storage scheme of Reid et al [DEREID86] and employ forward and backward linked lists of row counts and column counts which are set up initially. At each pivotal step these lists are updated. We do not search these lists fully for minimum merit count but accept near minimum merit count. After pivot operation Reid et al store the remaining updated matrix using a threaded list structure. We have adopted a different strategy whearby we update the remaining matrix in packed form and use two alternating temporary work areas. The growth of these work areas during these steps are indicated in table 4.1.

PROBLEM	MAXIMUM SUB BUMP	MAXIMUM AMOUNT USED	MAXIMUM AREA USED AT PIVOT STEP
AT1	166		
AT2	156		
BP1	356		
BP2	423		
GA1	55		
GA2	43		

TABLE 4.1

5. A Comparison of Alternative Representations for Basis Inverse

In this section we compare the PEFI method with the EFI method which is otherwise well established. For the EF1 method we have made use of the MINOS inveresion procedure.

The six test problems summarized in Table 5.1 are basis metrices collected from three well known LP Benchmark test problems [BRUNEL87], [GAYM85]. The three problems were partly processed by MINOS [MURTSU83] and their bases saved at given iterations. The basis files were then read back into FORTLP [MTTAMZ88] which in turn computed the invert. The invert statistics produced by FORTLP are presented in a summary form in table 5.2.

In these experiments we were mainly interested in the non zero build up in the inverse representation and in determining the amount of main computer storage saved by using the indirect eta vectors

NAME	PROBLEM	MATRIX ORDER	ITERETION NUMBER	BASIS NONZERO
ATLAS	AT1	315	900	1406
ATLAS	AT2	315	1200	1472
BPTEST	BP1	822	2500	4765
BPTEST	BP2	822	5500	4777
GANGES	GA1	1310	303	3453
GANGES	GA2	1310	603	5374

SUMMARY INFORMATION FOR THE TEST BASES
TABLE 5.1

In order to evaluate the stability of PEFI with sub bumps processed by Markowitz merit count and threashold pivoting, we solve the following system of equations and compute an error term in the following way.

Let

$$Bx = b \tag{5.1}$$

Where

$$b = Be \tag{5.2}$$

And e is a vector with unit components. If $\bar{x}$ is the computed solution then the accuracy of the algorithms is measured by the quantity.

$$\text{ERROR} = \| \bar{x} = e \|_\infty = \max_i | \bar{x}_i - 1 | .$$

This error term follows closely the measure suggested by Reid et al [DEREID86].

PROBLEM	TOTAL NONZERO	BUMP SIZE	NO.OF SUB-BUMPS	LARGEST SUB-BUMPS	TOTAL NO. OF NONZERO IN B^{-1}	A-MATRIX COEFF. USED				TOTAL NO. OF NON-ZERO IN U^{-1}
						ABOVE	SPLIT	BELOW	TOTAL	
AT1	1406	179	11	166	1697	199	172	155	526	442
AT2	1472	170	9	156	1729	284	168	171	623	493
BP1	4765	445	34	356	6298	1157	424	492	2073	1655
BP2	4777	460	19	423	6433	1184	448	369	2001	1905
GA1	3453	94	4	55	3424	312	135	2411	2858	140
GA2	5374	303	21	43	5369	397	353	2026	2776	339

DETAILED STATISTICS OF INVERSION INFORMATION
TABLE 5.2

f japanta.rep

A comparison of nonzero build up and inversion time using MINOS and FORTLP is presented in Table 5.3. The error term for the invert in FORTLP is also shown in this table.

		MINOS		FORTLP		FORTLP
PROBLEM	TOTAL NONZERO	TIME	NONZERO	TIME	NONZERO	ERROR
AT1	1406	6.7	1793	8.6	1694	0.25×10^{-9}
AT2	1472	6.8	1857	7.4	1736	0.81×10^{-12}
BP1	4765	40.6	6550	38.0	6191	0.50×10^{-9}
BP2	4777	45.2	6707	50.1	6402	0.65×10^{-10}
GA1	3453	11.0	3424	6.8	3424	0.76×10^{-12}
GA2	5374	14.3	5367	27.0	5369	0.33×10^{-13}

COMPARISON OF INVERSION TIME AND NONZEROS

TIMES IN SECONDS ON IBM PC RT6150

TABLE 5.3

6. Update of the Basis Inverse

Bartels Golub update within PEFI

The product form update appears very poor indeed compared with the Bartels-Golub and Forrest-Tomlin schemes on account of speed and accuracy. Unfortunately the PEFI or the reducible factorization in spite of producing, a sparser inverse (see section 5), is not suitable for applying the latter updates as we do not possess a B = LU decomposition. In the Bartels-Golub scheme [BARGOL69] at the k^{th} iteration after inversion we can express the factorization of the basis matrix B_k and its inverse as

$$B_k = LT_1\ T_2\ \ldots\ T_{k-1}\ U_k \quad \text{or} \tag{6.1}$$

$$(B_k)^{-1} = U_k^{-1}\ T^{-1}{}_{k-1}\ \ldots\ T_2\ T_1^{-1} L^{-1} \tag{6.2}$$

In (6.1) (6.2) we make use of the original L and derive T, ... T_{k-1} and U_k from the original U. The method of computing these is well established [GMSAWK84] and not repeated here. The Forrest-Tomlin update can be considered to be similar to Bartels and Golub with an additional row interchange.

The representation of $(B_k)^{-1}$ can be expressed as

$$(B_k)^{-1} = Q_k\ \ldots\ Q_1\ V_k\ R_{k-1}\ \ldots\ R_1\ L^{-1} \tag{6.3}$$

Where V_k is obtained from U_k with row deletion and row interchange which in each step is specified by the permutation matrices Q_1 ,Q_2 ....

- Bartels-Golub update within PEFI

We explain here how the PEFI (reducible factorization) can be adopted to derive an equivalent EFI and then used to introduce Bartels-Golub update. Consider the factorization of (3.3) and re-express this as

$$B = F_1 \times F_2 \times \ldots F_8 \tag{6.4}$$

- Deriving the U of the B = LU factor

It is easy to show that U of the LU factor is given by

$$U = F_3 \times F_6 \tag{6.5}$$

where $F_3 = \begin{bmatrix} I & & & \\ & U_{22} & & \\ & & I & \\ & & & I \end{bmatrix}$ $F_6 = \begin{bmatrix} I & & & \\ & I & & \\ & & U_{33} & \\ & & & I \end{bmatrix}$

The generalization of this method is to simply multiply the matrices with U^s of the diagonal block to obtain the full U, and in practice it is a simple copy (trivial multiplication) process. The statistics given in section 5 show that only (**) nonzeros are used to represent the U part of the matrix of order 822 and the total of (**) inverse nonzeros. Thus it is a small price to pay to take a copy of the U (the original U_{ii} are required to compute the L, see next section) in order to implement Bartels-Golub scheme.

- Deriving the L of the B = LU factor

We note that the L is used to solve a system in the form

$$L\bar{v} = v \tag{6.6}$$

where $\bar{v}$ is the solution for the given v. Consider the LU decomposition of the partitioned matrix

$$\begin{bmatrix} I & & \\ B_{22} & & \\ B_{32} & I & \end{bmatrix} = \begin{bmatrix} I & & \\ L_{22} & & \\ B_{32} & I & \end{bmatrix} \times \begin{bmatrix} I & & \\ & U_{22} & \\ & & I \end{bmatrix} \tag{6.7}$$

We wish to compute $\bar{v}$ expressed in the partitioned form, whereby

$$\begin{bmatrix} I & & \\ L_{22} & & \\ B_{32} & I & \end{bmatrix} \times \begin{bmatrix} \bar{v}_1 \\ \bar{v}_2 \\ \bar{v}_3 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \tag{6.8}$$

then $\bar{v}_1 = v_1$ (6.9)

$L_{22}\bar{v}_2 = v_2$ (6.10)

and $B_{32}\bar{v}_2 + \bar{v}_3 = v_3$ (6.11)

From (6.7) we have $B_{32} = (B_{32}U_{22}^{-1})$
Hence we can write (6.11) as

$B_{32}U_{22}^{-1}\bar{v}_2 + \bar{v}_3 = v_3$ (6.12)

(6.9), (6.10) and (6.12) allow us to compute $L\bar{v} = v$. Using $L_{11}, B_{32}, U_{22}^{-1}$ as opposed to using L made up of L_{22}, B_{32}.

- The implementation issues

At first sight this appears rather strange to go through this long process and not compute L as in the elimination form. The justification of going through these steps are set out below. We wish to make use of as much of the A matrix as possible in the representation of the B^{-1} since this saves storage. In this scheme the nonzeros of the columns in the explicit transformation matrices are not introduced and stored as double precision (*8) floating point values and the corresponding row indices of short integer (*2) are not duplicated. The statistics given in section 5 show that upto *** of the B^{-1} matrix can be represented in this way. It is easily seen that computation of $\bar{v}_3$ in (6.12) requires a second temporary
work area where $B_{32}U_{22}^{-1}v_2$ computed and this added to v_3.

- Schur complement update

Let the columns $a_{j_1} \ldots a_{j_k}$ be introduced in k simplex iterations after inversion and let these be denoted as columns $w_1 \ldots w_k$ of the matrix W which is of order (m×k). Let $e_{i_1} \ldots e_{i_k}$ denote unit vectors which indicate columns of B which were replaced in these k iterations. Let E denote a matrix of order (k × m) made up of corresponding row vectors $(e_{i_1}, e_{i_2} \ldots e_{i_k})^T$. If B_k denotes the basis matrix at the kth step then the solution to the system $B_k x = b$ can be related directly to the solution of the system

$$\begin{bmatrix} B & W \\ E & O \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix} \quad (6.13)$$

It is easily seen that the matrix

$$B_k = B + (W - BE^T)E, \quad (6.14)$$

and the system of k equations

$$Ex = 0 \tag{6.15}$$

set to zero value the variables which were made nonbasic in the simplex steps.

The (m+k) × (m+k) matrix in 6.13 can be factorized and re-expressed as

$$\begin{bmatrix} B & W \\ E & O \end{bmatrix} = \begin{bmatrix} B & O \\ E & S \end{bmatrix} \times \begin{bmatrix} I & Y \\ & I \end{bmatrix} \tag{6.16}$$

where $S = -(EB^{-1} W)$ and is known as the Schur complement [COTTLE74] of the matrix B. As in the LU solution we introduce an intermediate variable ν and write down the set of three equations

$$B\nu = b \quad \text{or} \quad \nu = B^{-1}b \tag{6.17}$$

$$E\nu + Sz = 0 \quad \text{or} \quad z = S^{-1}(-E\nu) \tag{6.18}$$

$$x + Yz = \nu \quad \text{or} \quad x = \nu - Yz \tag{6.19}$$

which are derived from the two systems

$$\begin{bmatrix} B & O \\ E & S \end{bmatrix} \begin{bmatrix} \nu \\ z \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix} \tag{6.20}$$

and

$$\begin{bmatrix} I & Y \\ & I \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix} = \begin{bmatrix} \nu \\ z \end{bmatrix} \tag{6.21}$$

We note that it is not necessary to compute the columns of the matrix Y ($Y=B^{-1} W$) explicitly. Since $Bx = b - Wz$ we can compute x by applying a second solve operation $x = \nu - B^{-1}(Wz)$. For a discussion of this and the method of solving a transposed system (backward transformation) see [GMSAWR84]. For the implementation of the method we use S^{-1} in the explicit form and a second work area. It is interesting to note that the use of such partitioned inverse representation has been known to other investigators [GEORGE74] but the application of this method to the update procedure is novel.

- Framework of Experimentation

The framework of experiments carried out by us is described here. We have taken the results of EFI and Bartels-Golub update as in MINOS to compare against the PFI, PEFI with Bartels Golub and PEFI with Schur complement, updates as implemented within FORTLP. The results illustrating relative speed and nonzero build up (to be completed) are set out in Table 6.1.

Problem No and Basis	Information	Initial	EFI and B-G Update MINOS After n iterations		Schur Complement FORTLP After n iterations		PEFI with B-G update FORTLP After n iterations		PFI update FORTLP After n iterations	
			n = 40	n = 80	n = 40	n = 80	n = 40	n = 80	n = 40	n = 80
BPA	Nonzero	xxx	xxx	xxx						
	Time	--	----	----						
BPB	Nonzero	xxx								
	Time	-								

Table 6.1 A comparison of alternative update methods

7. Acknowledgments

We are extremely grateful to NAG Ltd for their continued interest in this project and their collaborative financial support. We also thank SERC for their support. We had approached Dr J K Reid of Harwell Laboratories and Dr P Gill of Stanford Optimization Laboratories for information on LAO5 and LUSOL subroutines respectively. They were immediately forthcoming: after closely studying these two codes we have devised our implementation strategy. Dr J Judice had worked on a related part of the project [JUMITZ87] and was supported again by an SERC visiting fellowship. We acknowledge many stimulating ideas put forward by him, not the least for his suggestion that we adopt Schur complement update procedure. Dr K Darby-Dowman [DARBYK80] had earlier worked on algorithms for restructuring basis matrices and we have drawn heavily on the algorithms and subroutines developed by him.

REFERENCES

[BARGOL69] Bartels, R H, and Golub, G H, (1969), The simplex method of linear programming using LU decomposition, Communications of the ACM, pp 266-268, 275-278.

[BEALEM68] Beale, E M L, (1968), Mathematical programming in practice, Pitman, London.

[BEALEM84] Beale, E M L, (1984), private communication.

[BISMER77] Bischop, J, and Meeraus, A, (1977), Matrix augmentation and partitioning in the updating of the basis inverse, Mathematical Programming, vol 13, pp 241-254.

[BRUNEL87] A collection of test problems held by the mathematical programming group at Brunei University

[BUNROSE76] Bunch, J R, and Rose, D J, (1976) (eds), Sparse matrix computation, Academic Press.

[COTTLE74] Cottle, R W, (1974), Manifestations of the Schur complement, Linear Algebra Applic, vol 8, pp 189-211.

[DARBYK80] Darby-Dowman, K, (1980), The exploitation of sparsity in large scale linear programming problems and data structures and restructuring algorithms for basis matrices, PhD thesis, Brunei University, UK

[DARMIT81] Darby-Dowman, K, and Mitra, G, (1981), An investigation of algorithms used in restructuring of linear programming basis matrices prior to inversion, in Studies of Graphs and Discrete Programming, (ed) Hanson, P, North Holland, pp 69-73.

[DARMIT83] Darby-Dowman, K, and Mitra, G, (1983), Matrix Storage Schemes in Linear Programming, SIGMAP, Bulletin of the ACM.

[DEREID86] Duff, I S, Erisman, A M, and Reid, J K, (1986), Direct method for sparse matrices, Oxford Science Publications, Clarendon Press, Oxford.

[DFREID79] Duff I S, and Reid, J K, (1979), Performance evaluation of codes for sparse matrix problems, in Fosdick, L O, (ed) Performance evaluation of numerical software, North Holland, 1979.

[DGEPLE82] Duff, I S, Grimes, R G, Lewis, J G, and Poole, W G, Jr, (1982), Sparse matrix test problems, SIGNUM Newsletter, Association of Computing Machinery, NY, vol 17 (2), 22.

[DUFFIS84] Duff, I S, (1984), Direct methods for solving sparse systems of linear equations, SIAM Journal of SCI STAT COMPUT, vol 5, No 3, pp 609-619.

[DUFSTW79] Duff, I S, Stewart, G W, (1979) (eds) Sparse matrix proceedings, 1978, SIAM, Philadelphia, PA.

[ERGRLP85] Erisman, A M, Grimes, R G, Lewis, J G, and Poole, W G (1985), A structurally stable modification of Hellerman-Rerick's P^4 algorithm for reordering unsymmetric sparse matrices, SIAM Journal of NUMER ANAL, vol 22, No 2, pp 369-384.

[FORTOL72] Forrest, J J H and Tomlin, J A, (1972), Updated triangular factors of the basis to maintain sparsity in the product form simplex method, Mathematical Programming, vol 2, No 3, pp 263-278.

[GAYM85] Gay, D M, (1985), Electronic mail distribution of linear programming test problems, Mathematical Programming Society COAL Newsletter.

[GEORGE74] George, J A, (1974) On block elimination for sparse matrices, SIAM Journal of Numer. Anal., vol 11, pp 452-455.

[GMSAWR84] Gill, P E, Murray, W, Saunders, M A, and Wright, M H, (1984), Sparse matrix methods in optimisation, SIAM Journal of SCI STAT COMPUT, vol 5, No 3, pp 562-589.

[GRGLIU80] George, A and Liu, J W H, (1981), Computer solution of large sparse positive definite systems, Prentice Hall, New Jersey.

[HALLML56] Hall, M (1956), An algorithm for district representatives, American Mathematical Monthly, vol 63, p616-617.

[HELRAR71] Hellerman, E, and Rarick, D, (1971), Reinversion with the preassigned pivot procedure, Mathematical Programming, vol 1, pp 195-216.

[HELRAR72] Hellerman, E, and Rarick, D, (1972), The partitioned preassigned pivot procedure, in [ROSWIL72].

[HOCJES88] Hockney, R W, and Jesshope, C R, (1988), Parallel Computers, 2nd edition, Adam Hilger, UK.

[JUMITZ87] Judice, J D, Mitra, G, and Tamiz, M, (1987), A program to reorder and solve sparse unsymmetric linear systems using the envelope method, Technical Report/04/87, Dept of Maths & Stats, Brunel University, UK.

[KALANJ71] Kalan, J E, (1971), Aspects of large scale in-core linear programming, proceedings of the annual conference of the ACM, pp304-313.

[LINMAH77] Lin, T D, and Mah, R H, (1977), Hierarchical partition a new optimal pivoting algorithm, Mathematical Programming, vol 12, pp 260-278.

[MARSTN81] Marsten, R E, (1981), The design of XMP linear programming library, ACM Transactions on Mathematical Software.

[MITRAG73] Mitra G, (1973), Sparse inverse in the factored form and maintaining sparsity during simplex iterations, in, Software for Numerical Mathematics, Evans, D J, (ed), Academic Press.

[MITRAG76] Mitra, G, (1976), Theory and application of mathematical programming, Academic Press, London.

[MITAMZ88] Mitra G, and Tamiz, M, (1988), FORTLP user manual, Brunei University and NAG Ltd.

[MITAMZ89] Mitra G, and Tamiz M, (1989), Algorithms and software techniques for constructing linear and integer programming system for large sparse problems, Academic Press, Forthcoming.

[MURTSU83] Murtagh, B A, and Saunders, M A, (1983), MINOS 5.1 user's guide, Technical Report SOL 83-20R, Department of Operational Research, Stanford University.

[PAPDIM76] Papadimitriou, C H, (1976), The N-P completeness of the bandwidth minimisation problem, COMPUTING, vol 16, pp 263-270.

[REIDJK71] Reid, J K, (1971) (ed), Large sparse sets of linear equations, Academic Press.

[REIDJK76] Reid, J K, (1976), Fortran subroutines for handling sparse linear programming bases, Report AERE-R8269, Harwell.

[REIDJK82] Reid, J K, (1982), A sparsity-exploiting variant of the Bartels-Golub decomposition for linear programming bases, Mathematical Programming, vol 24, No 1, pp 55-69.

[ROSWIL72] Rose, D J, and Willoughby R A, (1972) (eds), Sparse matrices and their applications, Plenum Press.

[SMITHB87] Smith, B, (1987), Private communication concerning LP system developed at Leeds University for Crew Scheduling.

[TAMIZM86] Tamiz, M, (1986), Design development and testing of a general linear programming system, PhD thesis, Brunel Universit.

[TOMLIN72] Tomlin, J A, (1972), Pivoting for size and sparsity in linear programming inversion routines, IMA Journal, vol 10, pp 289-295.

[WILLBY69] Willoughby, R A, (1969) (ed), Proceedings of the symposium on sparse matrices and their applications, IBM Report RA1, Yorktown Heights, NY.

fjapan. rep