Update procedures for the parallel revised simplex method

J. A. J. Hall and K. I. M. McKinnon

30^{th} September 1992

Technical Report MSR 92-13

Department of Mathematics and Statistics
University of Edinburgh

Update procedures for the parallel revised simplex method

J. A. J. Hall K. I. M. McKinnon

30^{th} September 1992

Abstract

In the parallel revised simplex method proposed by Hall *et al.* in [8], the inversion of basis matrices is perfomed in parallel with the simplex iterations. As a result the update procedure must accommodate the set of basis changes which have occurred since commencing the inversion. This paper shows that update procedures which modify the factors of the matrix which is inverted are inappropriate for the parallel revised simplex method and that either the product form or an update based on the use of a Schur complement should be used. The essential difference between these approaches is highlighted and the implementation of the product form within the parallel revised simplex method is shown to be straightforward. Two approaches to the problem of using an update based on a Schur complement in the parallel revised simplex method are described.

1 Introduction

The simplex method remains the most flexible approach to the solution of linear programming (LP) problems despite the application of interior point methods introduced by Karmarkar in 1984. The standard simplex method has been implemented in parallel but the comparisons of the performance with a good serial implementation of the revised simplex method have not been published. It is expected that an efficient serial code for the revised simplex method will have superior performance for large sparse LP problems on all practical parallel machines. For these problems, parallel methods based on the revised simplex method with a factored inverse are proposed by Hall *et al.* in [8].

A LP problem has the form

$$\begin{array}{rl} \text{maximize} & f = c^T x \\ \text{subject to} & l \leq x \leq u \\ & Ax = b \\ \text{where} & x \in I\!R^n \quad \text{and} \quad b \in I\!R^m. \end{array} \tag{1}$$

BTRAN	Form $\pi_k^T = c_k^T B_k^{-1}$.
FTRAN	Form $s_k = B_k^{-1} v_k$.
UPDATE	Update the representation of B_k^{-1} using $B_{k+1} = B_k + (v_k - B_k e_{p_k}) e_{p_k}^T$.

Table 1: Operations involving B_k^{-1}

At any stage in the simplex method, the indices of the variables are partitioned into two sets. The basic variables x_B satisfy $l_B \leq x_B \leq u_B$ and each of the nonbasic variables x_N is at its upper or lower bound. If the problem is partitioned accordingly then the objective function is $f = c_B^T x_B + c_N^T x_N$ and the constraints are $A_B x_B + A_N x_N = b$, where the *basis matrix* A_B is nonsingular. This allows a new set of basic variables to be determined such that the objective function is increased. This process requires the vector $\hat{c}_N^T = c_N^T - c_B^T A_B^{-1} A_N$ and a particular column from the matrix $\hat{A}_N = A_B^{-1} A_N$.

The standard simplex method provides the necessary information by maintaining the vector $\hat{c}_N$ and the matrix $\hat{A}_N$ explicitly and updates them with a procedure which parallelises easily. Unfortunately the matrix $\hat{A}_N$ is unlikely to retain any sparsity in the matrix A in (1). The revised simplex method obtains the necessary information by solving linear systems involving A_B and forming a matrix-vector product with the matrix A_N. This requires a representation of A_B^{-1} which can be updated efficiently. If this representation and the corresponding solution procedure exploits sparsity efficiently then for large problems, the performance of a serial code based on the revised simplex method is likely to be superior to that of a parallelization of the standard simplex method.

It is convenient to identify iterations in the revised simplex method by the index k. When this is done, the vector c_B in iteration k is referred to as c_k and the basis matrix is referred to as B_k. The column chosen to enter the basis is denoted by v_k and this replaces column p_k of B_k, that is the vector $B_k e_{p_k}$. The operations in each iteration of the revised simplex method which involve the representation of B_k^{-1} are given in Table 1.

In the revised simplex method, the initial basis matrix is inverted and an update procedure is used which maintains a representation of B_k^{-1}. The representation increases in size with the number of iterations and may become unstable if an ill-conditioned basis matrix is encountered. For both these reasons it is necessary to reinvert some basis matrix B_r. This operation is termed INVERT and, in order to exploit sparsity, a factored inverse is formed. When this is complete it is typical for the index k to be reset to zero. Thus, in general, B_0 refers to any basis matrix which is inverted.

A fundamental difference between the parallel and serial revised simplex methods is that the parallel version performs iterations whilst INVERT is

proceeding. Thus it must be possible to perform FTRAN and BTRAN whilst the new factored inverse is unavailable. It is also important to be able to update the new representation to incorporate the basis changes occurring during INVERT with minimal delay.

In serial implementations of the revised simplex method the frequency of INVERT is limited since it requires the sequence of iterations to be halted whilst it is performed. Indeed, when numerical considerations allow, it is only the growth in the data required to represent B_k^{-1} which makes the investment in reinversion worthwhile. However, in the parallel revised simplex method it would be possible to perform INVERTs continually, that is to start inverting the current basis matrix immediately after the previous INVERT was completed.

Update procedures determine a representation of B_k^{-1} which may be classified according to whether the original factored inverse of B_0 is retained or modified. It will be seen that this distinction is particularly important when considering a parallel implementation. Most representations of B_k^{-1} build up a set of *eta vectors* whose density is problem-dependent. These vectors are generally of dimension m, with one produced for each update. There are four main approaches to updating which are used to implement the revised simplex method on serial machines. These are outlined in Section 2 where the two which are generally used in serial implementations are shown to be inappropriate for the parallel revised simplex method. Update procedures based on the other two approaches are developed in Section 3.

The parallel revised simplex method of Hall *et al.* [8] performs minor iterations in which the standard simplex method is applied to a subproblem of (1) formed from a small set of columns $v_{k1}, \ldots, v_{kt}$ of the matrix A_N. The corresponding set of columns $s_{k1}, \ldots, s_{kt}$ from $\hat{A}_N$ are obtained by a *multiple* FTRAN of the form $s_{kj} = B_k^{-1} v_{kj}, j = 1, \ldots, t$ which can be distributed so that it takes the same time as a single FTRAN. Although the minor iterations do not require operations with B_k^{-1}, it is needed to return to the revised simplex method. Thus it must be possible to incorporate the basis changes occuring during minor iterations without significant delay. The implications for the update procedures introduced in Section 3 are discussed in Section 4. A summary of conclusions is given in Section 5.

2 Update procedures

This section outlines the four main approaches to updating a representation of B_k^{-1} which are used in serial implementations of the revised simplex method. The two approaches which modify the factored inverse of B_0 are shown to be inappropriate for the parallel revised simplex method.

2.1 The Bartels-Golub and Forrest-Tomlin updates

The Bartels-Golub (BG) update [2] modifies the upper triangular factor of B_0 and produces a large set of short eta vectors. The Forrest-Tomlin (FT) update [6] is closely related to the BG update but the modifications of the upper triangular factor are restricted to a single row. One eta vector is produced for each update and although it may be short, its dimension is generally of order m.

The BG and FT updates are inappropriate for use within the parallel revised simplex method since they modify the upper triangular factor of B_0 and the corresponding factor for B_r is unavailable until INVERT is complete. Thus the updates corresponding to the basis changes which occurred during INVERT would have to be performed after it was completed. These updates can only be performed sequentially and if this were done before carrying out further iterations it would result in a bottleneck which severely restricts the performance of the parallel revised simplex method. This situation could be eased by continuing to use the inverse corresponding to B_0 and performing the updates on the new inverse in parallel with simplex iterations. However, the speed with which iterations are preformed may be such that it is not possible to catch up in this way.

It is envisaged by Hall *et al.* in [8] that an efficient parallel implementation of the revised simplex method requires parallelism to be exploited when performing FTRAN and BTRAN. This issue is addressed by McKinnon and Plab in [9] and requires an investment during INVERT which will only pay off if the inverse is applied many times. If the factors are modified then the return on this investment is lost. Thus any update procedure which requires the modification of the original is inappropriate for the parallel revised simplex method.

2.2 The product form update

The product form (PF) update which is due to Dantzig and Orchard-Hays [4] maintains the factored inverse of B_0 and a set of eta vectors. These are the same in number as the FT update but each is of dimension m.

The product form update stems from the observation that each basis matrix is obtained from the previous one by the rank one update.

$$\begin{aligned} B_{j+1} &= B_j + (v_j - B_j e_{p_j}) e_{p_j}^T, \quad j = 0, \ldots, k-1 \\ &= B_j (I + (s_j - e_{p_j}) e_{p_j}^T) \quad \text{where} \quad B_j s_j = v_j \\ &= B_j S_j. \end{aligned} \tag{2}$$

$$\text{Hence} \quad B_k = B_0 S_0 \ldots S_{k-1} \tag{3}$$

$$\text{so} \quad B_k^{-1} = S_{k-1}^{-1} \ldots S_0^{-1} B_0^{-1}. \tag{4}$$

Since s_j is required by the revised simplex method in iteration j, it is available at no additional cost. The matrix S_j is the identity matrix with column p_j

replaced by the vector

$$s_j = \left[s_{1p_j}, \ldots, s_{p_jp_j}, \ldots, s_{np_j}\right]^T$$

so it is easy to show that S_j^{-1} is of the same form, where column p_j consists of the *eta vector*

$$\eta_j = \left[\frac{-s_{1p_j}}{s_{p_jp_j}}, \ldots, \frac{1}{s_{p_jp_j}}, \ldots, \frac{-s_{np_j}}{s_{p_jp_j}}\right]^T.$$

Thus the PF representation of B_k^{-1} consists of the factored inverse of B_0 and the vectors $\eta_j, j = 0, \ldots, k-1$.

The numerical properties of the PF representation of B_k^{-1} depend on the condition of basis matrices $B_0, \ldots, B_{k-1}$. If any basis matrix B_r is ill-conditioned then there may be a large relative error in the vector s_r and/or growth in the eta vector η_r. In this case, operations with B_r^{-1} (and hence operations with B_k^{-1} for $k > r$) are unstable. If serious growth occurs it is necessary to reinvert the basis matrix B_r and if this were to occur too frequently it would seriously impair the efficiency of the revised simplex method.

2.3 The block LU update

Update procedures based on the use of a Schur complement were first proposed by Bisschop and Meeraus [3] for large scale LP problems. It is sufficient to maintain the factored inverse of B_0 and update a small square dense Schur complement. This gives a uniquely low storage requirement amongst update procedures. However, for reasons of efficiency, one of two possible sets of eta vectors akin to those of the PF update is generally retained. This yields either the block LU (BLU) update described by Gill *et al.* [1], which has been implemented by Eldersveld and Saunders [5], or the alternative block LU (ABLU) update described by Hall in [7]. The latter is more appropriate to active set methods for LP so it is the BLU update which is considered in this paper.

The update procedures which make use of a Schur complement relate the current basis matrix B_k to the basis matrix B_0 by a single update rather than by a sequence of rank one updates of the form (2). This multiple rank update is of the form

$$B_k = B_0 + (V_k - B_0 I_k^T) I_k. \tag{5}$$

Clearly V_k need not contain any column of B_0 which has been removed and subsequently returned to the basis. Thus (5) represents $l \leq k$ net column interchanges by which B_k may be obtained from B_0. If $l < k$ then it is said that *cancellation* of basis changes has occurred. Column p_j of B_0 is replaced by the vector v_j, for $j = 0, \ldots, l-1$ so

$$V_k = [v_0, \ldots, v_{l-1}] \quad \text{and} \quad I_k^T = \left[e_{p_0}, \ldots, e_{p_{l-1}}\right].$$

In iteration k, cancellation happens if the column entering the basis left it since inversion, or if the column leaving the basis entered it since inversion. That is the event

$$v_k = B_0 e_{p_j}, \qquad \text{for some } p_j \in \{p_0, \ldots, p_{l-1}\} \tag{6}$$

$$\text{or} \quad B_k e_{p_k} = v_j, \qquad \text{for some } v_j \in \{v_0, \ldots, v_{l-1}\}. \tag{7}$$

If both of these events occur then l is reduced by one, if exactly one occurs then l is unchanged and otherwise l increases by one. Cancellation tends to occur when the simplex method is at a degenerate vertex and may be significant within a short sequence of basis changes.

It follows from (5) that B_k may be represented as

$$B_k = B_0(I + (Y_k - I_k^T)I_k), \tag{8}$$

where the columns of Y_k are the vectors $y_j = B_0^{-1}v_j$, for $j = 0, \ldots, l-1$. Whilst these vectors are akin to the eta vectors of the PF update in terms of number, length and density, a fundamental difference is that their numerical properties are determined by the condition of just B_0. By inverting (8) and applying the Sherman-Morrison-Woodbury formula,

$$B_k^{-1} = (I + (Y_k - I_k^T)I_k)^{-1}B_0^{-1} = (I - (Y_k - I_k^T)C_k^{-1}I_k)B_0^{-1}, \tag{9}$$

where the *Schur complement* $C_k = I_k Y_k$ is well-defined if B_0 is nonsingular. Clearly C_k is nonsingular if and only if B_k is nonsingular.

There is no scope for cancellation within the PF representation of B_k^{-1} since this is formed by inverting the product (3) of B_0 with k elementary matrices rather than (implicitly) inverting the matrix $I + (Y_k - I_k^T)I_k$ in (8) during each iteration.

An implementation of the revised simplex method using the BLU update is described by Eldersveld and Saunders in [5]. They show that when it is used to solve the *netlib* test set, with INVERT performed every 100 updates, the average number of eta vectors is only 25 to 40, with a mean of 34. This should be compared with an average of 50 eta vectors for the PF update, if stability requirements allow 100 updates. Thus the reduction in operations with eta vectors due to cancellation when the BLU update is used is significant. For large problems this is expected to more than make up for the additional work required to operate with the inverse of the Schur complement.

2.4 FTRAN and BTRAN for the PF and BLU updates

When using the PF update, the FTRAN operation in Table 1.1 is performed by operating on v_k with B_k^{-1} given by (4). For the PF update, the event analogous to (6) is

$$v_k = B_j e_{p_j}, \quad \text{for some } j \in \{0, \ldots, k-1\}. \tag{10}$$

If this occurs then FTRAN reduces to

$$s_k = S_{k-1}^{-1} \dots S_j^{-1} e_{p_j}. \tag{11}$$

It is clear from (4) that BTRAN is given by

$$\pi_k^T = c_k^T S_{k-1}^{-1} \dots S_0^{-1} B_0^{-1}. \tag{12}$$

For the BLU update, FTRAN is performed by operating on v_k with B_k^{-1} given by (9). If event (6) occurs then FTRAN reduces to

$$s_k = (I - (Y_k + I_k^T) C_k^{-1} I_k) e_{p_j}.$$

Otherwise the first operation in FTRAN forms $y_k = B_0^{-1} v_k$ and it is observed that this is the eta vector which is required by the BLU update. It follows from (9) that BTRAN is given by

$$\pi_k^T = c_k^T (I - (Y_k - I_k^T) C_k^{-1} I_k) B_0^{-1}. \tag{13}$$

It is convenient to express $c_k = d_0 + I_k^T d_k$, in which case (13) reduces to

$$\pi_k^T = (d_0^T + [d_0^T (Y_k - I_k^T) + d_k^T] C_k^{-1} I_k) B_0^{-1} \tag{14}$$

In phase II of the simplex method it is possible to write $c_k = c_0 + I_k^T d_k$, where c_0 is the vector c_B corresponding to basis matrix B_0 and the components of d_k update c_0 according to the basis changes. In this case the components of $Y_k^T d_0$ in (14) only require updating when event (6) occurred in the previous iteration, requiring at most a single scalar product. It is clear from (12) that this saving is not available when using the PF update.

It is important to observe that the operation with the set of eta vectors Y_k for the BLU update is merely a matrix-vector product. Thus it is easy to exploit parallelism by distributing the columns of Y_k amongst a set of processors. This natural parallelism is not available when using the PF update. Depending on the sparsity of the eta vectors, it is shown by McKinnon and Plab [9] that it is possible to exploit parallelism to a lesser extent when performing operations with the eta vectors of the PF update.

3 Parallel update procedures

In this section update procedures for the parallel revised simplex method based on the use of the product form or a Schur complement are developed. Numerical considerations dictate that there is only one practical procedure based on the PF update and it is shown that its implementation within the parallel revised simplex method is straightforward. A continuum of procedures which make use of a Schur complement are developed which has the parallel product form approach as one extreme and a parallelisation of the serial BLU update at the other.

Basis matrices	B_0	B_1	...	B_r	B_{r+1}	...	B_k
Before completing INVERT	B_0	S_0	...	S_{r-1}	S_r	...	S_{k-1}
After completing INVERT				B_r	S_r	...	S_{k-1}

Table 2: Representation of B_k using the product form

3.1 Parallel product form updates

If the serial PF update is continued whilst INVERT is proceeding then, it follows immediately from (3) that once it is complete, B_k may be expressed as

$$B_k = B_r S_r \dots S_{k-1} \tag{15}$$

$$\text{so} \quad B_k^{-1} = S_{k-1}^{-1} \dots S_r^{-1} B_r^{-1}. \tag{16}$$

Thus no numerical operations are required to obtain the PF representation once INVERT is complete. Unfortunately the eta vectors required to perform operations with B_k^{-1} depend upon the condition of all the previous basis matrices since inverting B_0 and, inductively, upon the condition of all the basis matrices since commencing the revised simplex method. Thus the effect of growth in any eta vector affects the stability of all subsequent iterations. The numerical consequences of this may be unacceptable.

The following procedure addresses this problem and is termed the parallel product form (PPF) update. Once INVERT is complete, rather than continue with the existing eta vectors, they are recalculated using the inverse of B_r. This requires another multiple FTRAN whose cost is not prohibitive when performed in parallel. Whilst the new eta vectors are identical to the old using exact arithmetic, the dependence on the condition of the basis matrices preceeding B_r is lost. Unfortunately, serious growth in some eta vector η_R, $R \geq r$, requires the inversion of B_{R+1}, before there has been any opportunity to use the inverse of B_r! It is easy to see that this situation could lead to most advantages of parallelism being lost.

3.2 Parallel block LU updates

When INVERT is complete for B_r, the expression

$$B_k = B_r(I + (Y_K - I_K^T)I_K), \tag{17}$$

where $Y_K = B_r^{-1}V_K$ and I_K correspond to the (net) basis changes occurring during INVERT, may be inverted to give

$$B_k^{-1} = (I - (Y_K - I_K^T)C_K^{-1}I_K)B_r^{-1}. \tag{18}$$

Thus the aim of the BLU update is to perform FTRAN as

$$s_k = (I - (Y_K - I_K^T)C_K^{-1}I_K)B_r^{-1}v_k. \tag{19}$$

An update procedure must allow operations with B_k^{-1} to be performed before INVERT is complete and allow the representation corresponding to (19) to be obtained without a significant overhead once INVERT is complete.

The first approach, termed the PBLU-1 update merely continues the serial BLU update whilst INVERT is proceeding. However, in order to use (19) when INVERT is complete, it is necessary to form $Y_K = B_r^{-1}V_K$, then form and invert $C_K = I_K Y_K$. This requires a multiple FTRAN which, when performed in parallel, takes the same time as a single FTRAN. Forming C_K requires the appropriate components to be extracted from the columns of Y_K. The cost of inverting C_K depends upon the number of iterations of the simplex method which have ocurred since commencing INVERT. This may be large, particularly if minor iterations have been performed, in which case parallel dense Gaussian elimination may be necessary.

The second approach avoids the multiple FTRAN by forming a second Schur complement so is termed the PBLU-2 update. Putting $k = r$ in (9) and using the expression to substitute for B_r^{-1} in (18) gives

$$B_k^{-1} = (I - (Y_K - I_K^T)C_K^{-1}I_K)(I - (Y_r - I_r^T)C_r^{-1}I_r)B_0^{-1}. \tag{20}$$

Hence, whilst INVERT is proceeding, FTRAN may be performed as the operation

$$s_k = (I - (Y_K - I_K^T)C_K^{-1}I_K)(I - (Y_r - I_r^T)C_r^{-1}I_r)B_0^{-1}v_k. \tag{21}$$

The eta vector required to update Y_K and C_K in (20) is the vector $y_k = B_r^{-1}v_k$ which is formed naturally as an intermediate stage in (21). Clearly Y_K and C_K in (19) are available when INVERT is complete so the multiple FTRAN and Schur complement inversion of the PBLU-1 update may be avoided.

The PBLU-1 and PBLU-2 updates have an important difference in terms of numerical stability. For the PBLU-1 update, since the matrix Y_K is formed using the inverse of B_r, the eta vectors (and hence the Schur complement C_K) are independent of the condition of B_0. However the eta vectors which form Y_K for the PBLU-2 update are obtained from (21) and, as such, depend upon the condition of B_0 and, inductively, upon the condition of all basis matrices which have been inverted. This dictates that Y_K should be reformed, requiring a multiple FTRAN, and C_K formed and inverted.

Whilst the PBLU-1 update requires a multiple FTRAN, it works with the full set of basis changes since completing the inversion of B_0. The PBLU-2 update uses two sets of basis changes which form a partition of the full set. It is clear from (20) that it is only possible to take advantage of cancellation within these sets. Hence, if Y_k, Y_r and Y_K contain l_k, l_r and l_K eta vectors respectively, then $l_k \leq l_r + l_K$. The compensation for the PBLU-2 update requiring the retention of more eta vectors is the requirement for operations with the inverse of two smaller Schur complements rather than one larger Schur complement. The total

Basis matrices	B_0	B_1	...	B_r	B_{r+1}	...	B_k
Before completing INVERT (PBLU-1)	B_0	Y_k C_k					
Before completing INVERT (PBLU-2)	B_0	Y_r C_r			Y_K C_K		
After completing INVERT				B_r	Y_K C_K		

Table 3: Representation of B_k using the block LU update

work required by the former is $l_r^2 + l_K^2$ operations whereas the latter requires l_k^2 operations and it is likely that $l_r^2 + l_K^2 < l_k^2$.

By working with two sets of eta vectors, the possibility for exploiting parallelism by distributing the corresponding operations may be reduced significantly when using the PBLU-2 update. This is due to the fact that there will only be a few eta vectors in Y_K soon after commencing INVERT. With the PBLU-1 update there are always at least l_r scalar products to be distributed.

The representation of B_k using the PBLU-1 and PBLU-2 updates is summarised in Table 3.

An extension of the PBLU-2 update is the PBLU-4 update which uses four Schur complements of still smaller dimension corresponding to partitions of the sets of basis changes occuring during iterations $0, \ldots, r-1$ and during INVERT into two subsets. Whilst this may reduce the cost of performing the update and using the invertible representation, it depends on the condition of twice as many basis matrices as the PBLU-2 update. The number of Schur complements in the representation of B_k^{-1} may be increased to a maximum of k, each of unit dimension. If this is done then the representation of B_k^{-1} reduces to that of the product form. Thus a continuum of update procedures from the single Schur complement PBLU-1 update to the product form PPF update may be considered.

Experience reported by Eldersveld and Saunders [5] suggests that the serial BLU update is more efficient than the PF update due to the reduction in the number of operations with eta vectors. This is likely to be magnified in the parallel implementation since a product involving BLU eta vectors may be distributed. Although a continuum of methods is available, the gain in efficiency by distributing the operations with a single set of eta vectors Y_k may be such that the PBLU-1 update is preferable. This will be determined by practical experience.

4 Accommodating the basis changes during minor iterations

When performing minor iterations of the standard simplex method on a subset of the columns of the LP problem the representation of B_k^{-1} is not required. However, in order to return to the revised simplex method, the representation of

B_k^{-1} should either be updated during the minor iterations or the basis changes should be retained and the corresponding updates performed after completing the minor iterations. In this section the resulting implications for the update procedures introduced in Section 3 are discussed.

The columns of the matrix $\hat{A}_N$ in the minor iterations are the vectors $s_{kj} = B_k^{-1}v_{kj}, j = 1, \ldots, t$ corresponding to the current basis matrix B_k. As such the eta vector required to perform the product form update is available naturally so the minor iterations pose no problems when using the PPF update.

The eta vectors required by the PBLU updates are the vectors $y_k = B_0^{-1}v_k$ which are not readily available during the minor iterations. To accumulate these vectors during the minor iterations requires a single operation with the inverse of B_0 during each minor iteration which would inhibit performance significantly. It is therefore clear that the basis changes occurring during minor iterations should be accumulated and the corresponding eta vectors calculated as a multiple FTRAN when the minor iterations are complete. It is possible that INVERT is completed during the multiple FTRAN, in which case a second multiple FTRAN corresponding to the new factored inverse would be necessary. Although this occurrence would be unfortunate, it is not expected to occur sufficiently often to reduce performance significantly.

5 Conclusions

The possibilities for the implementation of the full range of serial update procedures within the parallel revised simplex method has been discussed and the following conclusions can be made. Update procedures which modify the factored inverse are inappropriate since the updates corresponding to basis changes which have occurred during INVERT must be applied sequentially before the representation of the current basis matrix may be used. The inherent numerical problems of the PF update are magnified when implemented in the simplest and most efficient manner so an alternative procedure is suggested which alleviates this problem somewhat. Even with this approach, there is no guarantee that instability cannot lead to the PF representation being be prohibitively inefficient. A continuum of methods based on the use of a Schur complement has been constructed which ranges from the stable PBLU-1 update to the unstable PPF update. The gain in efficiency when the operations with the set of eta vectors corresponding to the PBLU-1 update are distributed may be such that the only competition is the PBLU-2 update without a multiple FTRAN which is inferior numerically. Practical experience will allow further conclusions to be drawn.

References

[1] P. E. G. and W. Murray, M. A. Saunders, and M. H. Wright. Sparse matrix methods in optimization. *SIAM J. Sci. Stat. Comput.* , 5:562–589, 1984.

[2] R. H. Bartels. A stabilization of the simplex method. *Numer. Math.* , 16:414–434, 1971.

[3] J. Bisschop and A. J. Meeraus. Matrix augmentation and partitioning in the updating of the basis inverse. *Math. Prog.* , 13:241–254, 1977.

[4] G. B. Dantzig and W. Orchard-Hays. The product form for the inverse in the simplex method. *Math. Comp.* , 8:64–67, 1954.

[5] S. K. Eldersveld and M. A. Saunders. A block-lu update for large-scale linear programming. Technical Report SOL 90-2, Systems Optimization Laboratory, Stanford University, 1990.

[6] J. J. H. Forrest and J. A. Tomlin. Updated triangular factors of the basis to maintain sparsity in the product form simplex method. *Math. Prog.* , 2:263–278, 1972.

[7] J. A. J. Hall. *Sparse matrix algebra for active set methods in linear programming.* PhD thesis, University of Dundee Department of Mathematics and Computer Science, 1991.

[8] J. A. J. Hall, K. I. M. McKinnon, and F. Plab. Parallel methods for linear programming based on the revised simplex method with a factored inverse. Technical report, Department of Mathematics and Statistics, University of Edinburgh, In preparation.

[9] K. I. M. McKinnon and F. Plab. Exploiting parallelism when solving linear systems using a factored inverse. Technical report, Department of Mathematics and Statistics, University of Edinburgh, In preparation.