Permuting Spiked Matrices to Triangular Form and its Application to the Forrest-Tomlin Update

Lukas Schork* Jacek Gondzio†

School of Mathematics, University of Edinburgh, Edinburgh EH9 3FD, Scotland, UK

Technical Report, July 2017

1 Introduction

This paper is concerned with the problem of permuting a spiked matrix to triangular form. A spiked matrix results from changing one column or one row in a triangular matrix. In this paper we focus on changing one column in an upper triangular matrix, see Figure 1(a). Spiked matrices arise in updating the LU factors of a matrix after a column change. The LU update methods of Bartels and Golub [1] and Forrest and Tomlin [3] use algebraic operations to transform a spiked matrix to triangular form. We present an LU update method which does the transformation by permutation alone whenever this is possible and falls back to the Forrest-Tomlin update otherwise.

A similar idea is presented by Reid [7]. Reid's method is a variant of the Bartels-Golub update which permutes the spiked matrix before applying algebraic operations in order to reduce the bump size. (The "bump" is the single block on the diagonal when the spiked matrix is regarded as a block triangular matrix.) The algorithm seems unsuitable for large scale matrices, however, since its runtime is proportional to the number of nonzeros in the bump. In contrast, the runtime of the algorithm presented here is not lower bounded by the bump size or the matrix dimension.

The problem is relevant for a number of computational mathematics applications which require computing and updating LU factors of an unsymmetric matrix. Possibly the most important application is the revised simplex method, which is used for the numerical comparisons in Section 5.

The mathematical description of the permutation method requires some terminology from graph theory and sparse matrices. The sparsity pattern of an $m \times m$ matrix A defines a directed graph $G = (V, E)$ with nodes $V = \{1, \ldots, m\}$ and edges $(i, j) \in E$ when $a_{i,j} \neq 0$. Self-edges for nonzero diagonal elements are included in E. A path $j_0 \rightsquigarrow j_k$ is a sequence of nodes $(j_0, \ldots, j_k)$ where an edge exists from each node to the next in the sequence. The nodes in the path other than j_0 and j_k are called *intermediate nodes*. A cycle is a path $j \rightsquigarrow j$ with at least one intermediate node. (A self-edge is not a cycle.) $\mathrm{Reach}_A(j)$ is the set of all nodes i for which there exists a path $j \rightsquigarrow i$ in the directed graph of A. Note that j might not be contained in $\mathrm{Reach}_A(j)$ if the self-edge is not present. A symmetric permutation of matrix A corresponds to a renumbering of the nodes in its graph G. We

*L.Schork@ed.ac.uk
†J.Gondzio@ed.ac.uk

say that A is *symmetrically permuted triangular* when there exists a permutation matrix P such that PAP^T is triangular. A is of this form if and only if G is acyclic; i. e. it does not contain a cycle.

The following lemma shows how a column permutation changes the directed graph of a matrix.

Lemma 1.1. *Let A be an $m \times m$ matrix and Q be the permutation matrix which reorders columns $j_0, \ldots, j_n$ in a cycle; i. e. column j_{k+1} of A becomes column j_k of AQ for $0 \le k \le n$, where $j_{n+1} = j_0$. Denote $G = (V, E)$ the directed graph of A and $G_Q = (V, E_Q)$ the directed graph of AQ. Then for any node $i \in V$*

$$\begin{aligned} (i, j_k) \in E_Q &\iff (i, j_{k+1}) \in E \quad \text{for } 0 \le k \le n, \\ (i, j) \in E_Q &\iff (i, j) \in E \qquad \text{for } j \notin \{j_0, \ldots, j_n\}. \end{aligned}$$

A note regarding notation: We make use of permutation matrices and permutation vectors. A permutation vector is a permutation of indices $1, \ldots, m$. When $\boldsymbol{p}$, $\boldsymbol{q}$ are permutation vectors, then $A(\boldsymbol{p}, \cdot)$ is the matrix whose i-th row is the p_i-th row of A and $A(\cdot, \boldsymbol{q})$ is the matrix whose j-th column is the q_j-th column of A. The corresponding permutation matrices P, Q are permutations of the identity matrix such that $A(\boldsymbol{p}, \boldsymbol{q}) = PAQ^T$.

2 The Forrest-Tomlin Update

Let B be an $m \times m$ matrix and $\bar{B} = B + (\boldsymbol{a} - B\boldsymbol{e}_j)\boldsymbol{e}_j^T$, where $\boldsymbol{e}_j$ is the j-th unit vector. Let B and $\bar{B}$ be nonsingular. It is assumed that B has been permuted so that the factorization $B = LU$ into a unit lower triangular matrix L and an upper triangular matrix U exists. The Forrest-Tomlin update [3] computes a row eta matrix R and a symmetrically permuted triangular matrix $\bar{U}$ so that $\bar{B} = LR\bar{U}$. A row eta matrix has the form

$$R = \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ r_{j1} & \cdots & 1 & \cdots & r_{jm} & \\ & & & \ddots & & \\ & & & & 1 & \end{bmatrix}. \tag{1}$$

To derive formulae for R and $\bar{U}$, the expression for $\bar{B}$ is manipulated as follows (see [6]).

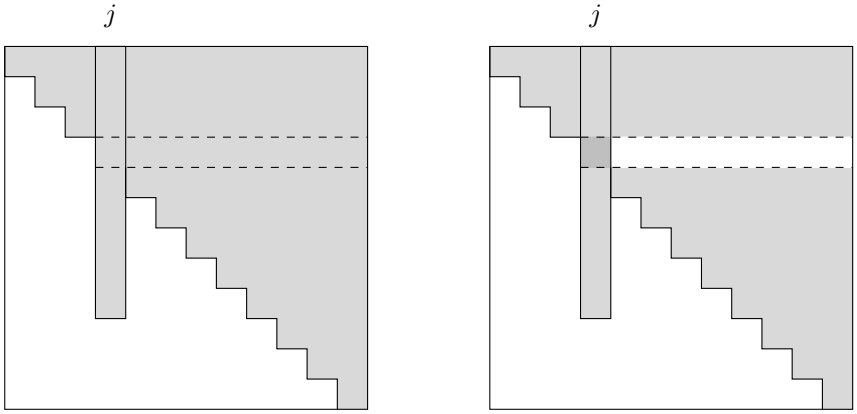
$$\begin{aligned} \bar{B} &= B + (\boldsymbol{a} - B\boldsymbol{e}_j)\boldsymbol{e}_j^T \\ \implies L^{-1}\bar{B} &= U + \left(L^{-1}\boldsymbol{a} - U\boldsymbol{e}_j\right)\boldsymbol{e}_j^T \\ &= U + (\hat{\boldsymbol{a}} - \boldsymbol{u}_j)\,\boldsymbol{e}_j^T = \hat{U}. \end{aligned}$$

$\hat{U}$ is a spiked upper triangular matrix as illustrated in Figure 1(a). The Forrest-Tomlin update eliminates the off-diagonal elements in row j of $\hat{U}$ by a row transformation. Let $\boldsymbol{w}^T$ be the row vector corresponding to row j of U without the diagonal element and let $\boldsymbol{r}^T = \boldsymbol{w}^T U^{-1}$. Because $r_j = 0$ the matrix $R = I + \boldsymbol{e}_j\boldsymbol{r}^T$ is of the form (1). Computing

$$R^{-1}\hat{U} = \left(I - \boldsymbol{e}_j\boldsymbol{r}^T\right)\hat{U} = \bar{U}$$

yields a matrix $\bar{U}$ in which row j is a singleton row with diagonal element $\bar{u}_{jj} = \hat{a}_j - \boldsymbol{r}^T\hat{\boldsymbol{a}}$. $\bar{U}$ is illustrated in Figure 1(b). It can be permuted to upper triangular form by a cyclic permutation of rows and columns j to m.

Figure 1:



(a) spike column in position j (b) after eliminating off-diagonals

Let B^r be the matrix after r column changes to B. Applying the Forrest-Tomlin update r times yields the factorization

$$B^r = LR^1 \cdots R^r U^r,$$

where each of the R^k is a row eta matrix and U^r is symmetrically permuted triangular. The permutation vector $\boldsymbol{p}$ which makes $U^r(\boldsymbol{p}, \boldsymbol{p})$ upper triangular is updated with each Forrest-Tomlin update by moving the index of the spike column to the end of $\boldsymbol{p}$.

3 Permutation to Triangular Form

When updating the LU factorization of very sparse matrices it frequently happens that the spiked matrix $\hat{U}$ is already permuted triangular. In this case the row operations of the Forrest-Tomlin update are unnecessary to restore triangularity. This section presents a procedure to find a permutation of $\hat{U}$ to triangular form or to prove that no such permutation exists. For simplicity it is assumed that $\hat{U}$ is obtained by inserting the vector $\hat{\boldsymbol{a}}$ into column j of an upper triangular matrix U, but the procedure also works when U is symmetrically permuted triangular. We have to distinguish whether or not the diagonal element of the spike column is zero.

3.1 Spike has a nonzero diagonal element

We first consider the case $\hat{a}_j \neq 0$. Because U and $\hat{U}$ differ only in column j, and because element (j, j) is nonzero in both matrices, $\text{Reach}_U(j) = \text{Reach}_{\hat{U}}(j)$. An example for this is shown in Figure 2.

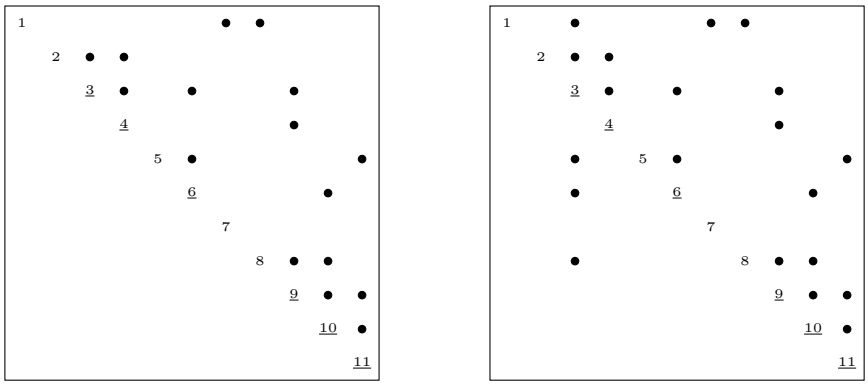
In the following $\mathcal{U}_j$ denotes the nonzero pattern of column j of U.

Theorem 3.1. *Let $\hat{U}$ have a zero-free diagonal and be upper triangular except for column j. Then $\hat{U}$ is permuted triangular if and only if*

$$\hat{\mathcal{U}}_j \cap \text{Reach}_{\hat{U}}(j) = \{j\}. \tag{2}$$

The permutation to triangular form is symmetric.

Proof. It is first shown that any permutation of a matrix with a zero-free diagonal to triangular form is symmetric. The proof is by contradiction. Assume that U has a zero-free



(a) original matrix U (b) $\hat{U}$ with spike in column 3

Figure 2: Example matrix U and spiked matrix $\hat{U}$. The dark circles represent off-diagonal nonzeros. The diagonal is assumed zero-free and numbered for easy reference. Nodes in Reach(3) are underlined.

diagonal and $U(\boldsymbol{p}, \boldsymbol{q})$ is upper triangular, where $\boldsymbol{p} \neq \boldsymbol{q}$ are permutation vectors. W.l.o.g. let $p_1 \neq q_1$ (otherwise consider a submatrix of U). Because $U(\boldsymbol{p}, \boldsymbol{q})$ is upper triangular, column q_1 of U is a singleton. However, because U and $U(\boldsymbol{p}, \boldsymbol{q})$ both have a zero-free diagonal, the column has nonzeros in positions q_1 and p_1, which yields the contradiction.

Secondly it is shown that $\hat{U}$ is permuted triangular if and only if (2) holds. Let $G = (V, E)$ be the directed graph of $\hat{U}$. Then $\hat{U}$ is permuted triangular if and only if G is acyclic. Because only column j of $\hat{U}$ has entries below the diagonal, any cycle in G must contain node j. Such a cycle exists if and only if $i \in \mathrm{Reach}_G(j)$ for some $i \neq j$ and $(i, j) \in E$. □

Figure 2(b) illustrates the theorem. Because the intersection of $\hat{\mathcal{U}}_3 = \{1, 2, 3, 5, 6, 8\}$ and Reach(3) $= \{3, 4, 6, 9, 10, 11\}$ is $\{3, 6\}$, $\hat{U}$ is not permuted triangular. Indeed, its graph contains the cycle $3 \rightsquigarrow 6 \rightsquigarrow 3$. On the other hand, if the entry in row 6 of the spike was not present, then $\hat{U}$ would be permuted triangular.

To implement the triangularity test, $\mathrm{Reach}_U(j)$ must be computed, usually by a graph traversal of U starting from node j. In some applications this can be omitted. When $\boldsymbol{e}_j^T B^{-1}$ has been computed before the update, then the (structural) nonzero pattern of $\boldsymbol{e}_j^T U^{-1}$ has been determined, which is $\mathrm{Reach}_U(j)$. Therefore $\hat{\mathcal{U}}_j$ and $\mathrm{Reach}_U(j)$ are available anyway and condition (2) can be tested at negligible cost. This applies, for example, to the revised simplex method.

When $\hat{U}$ is permuted triangular, then the permutation to triangular form is obtained as follows.

Lemma 3.2. *Let $\hat{U}$ have a zero-free diagonal and be upper triangular except for column j. When $\hat{U}$ is permuted triangular, then $\hat{U}(\boldsymbol{p}, \boldsymbol{p})$ is upper triangular, where $\boldsymbol{p}$ is obtained from the identity permutation $(1, \ldots, m)$ by removing the indices in $\mathrm{Reach}_{\hat{U}}(j)$ and appending them in topological order to the end.*

Proof. A matrix is upper triangular if the nodes in its graph $G = (V, E)$ are numbered in topological order, i. e. if $(i, k) \in E$ only if $i \leq k$.

Let $\boldsymbol{p} = (p_1, \ldots, p_s, j, p_{s+2}, \ldots, p_m)$, where $(j, p_{s+2}, \ldots, p_m) = \mathrm{Reach}_{\hat{U}}(j)$. Then the following holds true:

(i) The leading $s \times s$ block of $\hat{U}(\boldsymbol{p}, \boldsymbol{p})$ is upper triangular because it is the remainder after removing rows and columns from the spiked upper triangular matrix $\hat{U}$, including the spike column.

(ii) The trailing $(m-s)\times(m-s)$ block of $\hat{U}(\boldsymbol{p},\boldsymbol{p})$ is upper triangular because $(j, p_{s+2}, \ldots, p_m)$ are in topological order with respect to $\hat{U}$.

(iii) There are no edges (p_l, p_k) in the graph of $\hat{U}$ for $1 \le k \le s$ and $s+1 \le l \le m$, since otherwise p_k would have been in $\mathrm{Reach}_{\hat{U}}(j)$. Hence $\hat{U}(\boldsymbol{p},\boldsymbol{p})$ has a zero block of dimension $(m-s)\times s$ below the diagonal.

Consequently $\hat{U}(\boldsymbol{p},\boldsymbol{p})$ is upper triangular. □

3.2 Spike has a zero diagonal element

Secondly we consider the case $\hat{a}_j = 0$. Provided that $\hat{U}$ is nonsingular, it cannot be the symmetric permutation of a triangular matrix (a symmetric permutation does not permute off-diagonal elements on the diagonal). It may be an unsymmetric permutation of a triangular matrix, however.

Lemma 3.3. *Let Q be a permutation matrix such that $\hat{U}Q$ has a zero-free diagonal. When $\hat{U}$ is the permutation of an upper triangular matrix U, then $\hat{U}Q$ is a symmetric permutation of U.*

Proof. Assume that $P_1\hat{U}P_2 = U$ for permutation matrices P_1, P_2. Then $P_1(\hat{U}Q)Q^TP_2 = U$ and since $\hat{U}Q$ has a zero-free diagonal it follows from the proof of Theorem 3.1 that $P_1^T = Q^TP_2$. Hence $\hat{U}Q$ is indeed a symmetric permutation of U. □

It follows that testing if $\hat{U}$ is permuted triangular can be broken down into two steps. The first step is to find a column permutation Q that makes the diagonal of $\hat{U}Q$ zero-free. Such a permutation exists when $\hat{U}$ has full structural rank. The second step is to test if $\hat{U}Q$ is symmetrically permuted triangular. By Lemma 3.3 the result of step two is independent of the specific choice of Q.

A column permutation Q that makes the diagonal of $\hat{U}Q$ zero-free is found by computing a maximum matching. A maximum matching pairs each row i with a column j such that element (i,j) of $\hat{U}$ is nonzero and no row or column is paired twice. The diagonal of $\hat{U}$ already defines a partial matching in which each row i is matched to column i except for row j. This matching is increased by 1 by finding an alternating augmenting path [2]. In our setting an alternating augmenting path is defined by a sequence of column indices $(j_0, j_1, \ldots, j_n)$ where $j_0 = j$ and elements (j_k, j_{k+1}) for $0 \le k < n$ as well as (j_n, j_0) of $\hat{U}$ are nonzero. The permutation matrix Q which cycles columns $j_0, \ldots, j_n$ makes the diagonal of $\hat{U}Q$ zero-free.

Figure 3(a) shows a spiked upper triangular matrix in which the spike has a zero diagonal element. The arrows illustrate an alternating augmenting path through columns $(3, 5, 9)$. Figure 3(b) shows the matrix after a cyclic permutation of these columns.

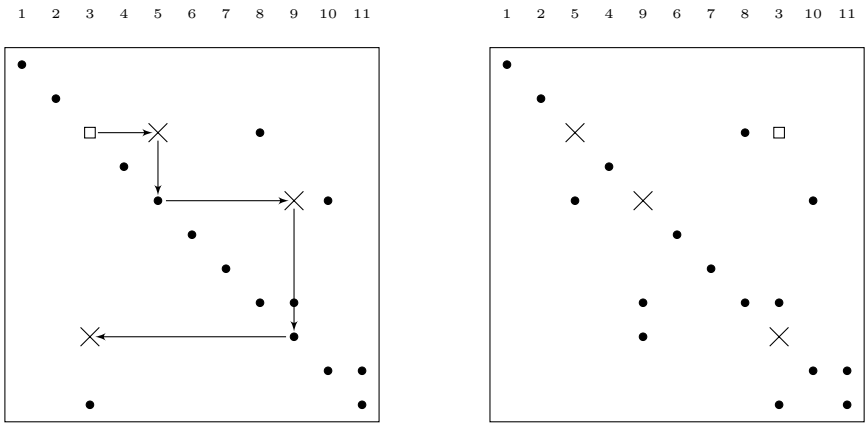
There are different techniques for finding an alternating augmenting path. One possible choice is a depth first search with look-ahead as described by Duff [2].

The column permutation Q defined by the augmenting path $(j_0, \ldots, j_n)$ makes $\hat{U}Q$ a spiked upper triangular matrix with spikes in columns $j_0, \ldots, j_n$, see Figure 3(b). Theorem 3.1 easily adapts to a matrix with multiple spike columns:

Theorem 3.4. *Let $\bar{U}$ have a zero-free diagonal and be upper triangular except for columns $j_0, \ldots, j_n$. Then $\bar{U}$ is permuted triangular if and only if*

$$\bar{\mathcal{U}}_{j_k} \cap \mathrm{Reach}_{\bar{U}}(j_k) = \{j_k\} \quad \textit{for } 0 \le k \le n.$$

Proof. This is shown analogously to the proof of Theorem 3.1 by using that any cycle in the directed graph of $\bar{U}$ must contain at least one of the nodes $j_0, \ldots, j_n$. □



(a) augmenting path in $\hat{U}$ (b) columns $(3, 5, 9)$ cycled

Figure 3: Example matrix $\hat{U}$ and permutated to zero-free diagonal. Dark circles and crosses are nonzeros, the white square is a zero. Columns are numbered by their index in $\hat{U}$.

Theorem 3.4 gives a simple characterization of whether $\hat{U}Q$ (and hence $\hat{U}$) is permuted triangular. It is inconvenient to implement, however, since computing $\mathrm{Reach}_{\bar{U}}(j_k)$ requires the pattern of the column permuted matrix $\bar{U} = \hat{U}Q$. In the implementation we would like to apply the column permutation only for an update, i. e. after determining that $\hat{U}$ is permuted triangular.

There is another characterization of whether $\hat{U}$ is permuted triangular given an augmenting path. Stated in the following theorem, this result requires more effort to derive, but is convenient to implement.

Theorem 3.5. *Let $\hat{U}$ be upper triangular except for column j_0, which has a zero diagonal element and one or more nonzeros below. Let $(j_0, j_1, \ldots, j_n)$ be an augmenting path in $\hat{U}$ and $j_{n+1} = j_0$. Let $G' = (V, E')$ be the directed graph of $\hat{U}$ without the edges (j_k, j_{k+1}) for $0 \le k \le n$. Then $\hat{U}$ is permuted triangular if and only if*

$$j_{k+1}, \ldots, j_n \notin \mathrm{Reach}_{G'}(j_k) \quad \text{for } 0 \le k \le n-1, \tag{3a}$$

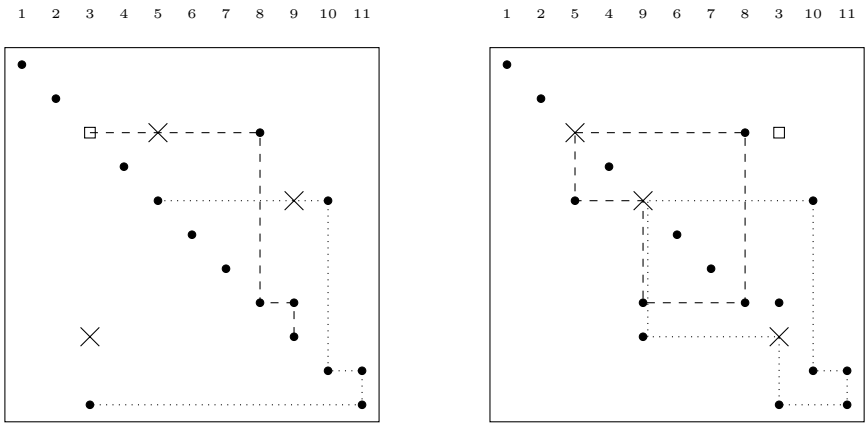
$$\mathrm{Reach}_{G'}(\{j_0, \ldots, j_n\}) \cap \hat{\mathcal{U}}_{j_0} = \{j_n\}. \tag{3b}$$

Before proving the theorem, it should be illustrated at an example. For the matrix in Figure 3(a) G' is the graph given by the pattern of dark circles. The crosses are the edges in the augmenting path which are excluded from G'. The matrix is plotted again in Figure 4. The dashed line in part (a) is a path $3 \rightsquigarrow 9$, meaning that $j_2 \in \mathrm{Reach}_{G'}(j_0)$. Hence condition (3a) is violated. It is seen in part (b) how this path together with the self-edges of diagonal elements of $\hat{U}$ becomes a cycle in the column permuted matrix, confirming that $\hat{U}$ is indeed not permuted triangular. Condition (3b) is also violated because $11 \in \mathrm{Reach}_{G'}(5)$ and $11 \in \hat{\mathcal{U}}_3$. The dotted line in Figure 4 shows how that path in G' becomes a cycle in the column permuted matrix.

For the following proof of Theorem 3.5 some quantities are fixed: Q is the permutation matrix corresponding to the augmenting path $(j_0, \ldots, j_n)$, $j_{n+1} = j_0$, $G_Q = (V, E_Q)$ is the directed graph of $\hat{U}Q$, and $G' = (V, E')$ is defined in Theorem 3.5.

Lemma 3.6. *For $0 \le k < l \le n$ there exists a path $j_l \rightsquigarrow j_k$ in G_Q.*

Proof. For $0 \le k \le n-1$, (j_{k+1}, j_{k+1}) is a self-edge in $\hat{U}$. By Lemma 1.1 (j_{k+1}, j_k) is an edge in G_Q. Consequently, for $0 \le k < l \le n$ there is a path $j_l \rightsquigarrow j_k$ in G_Q. □



(a) example matrix $\hat{U}$ (b) columns $(3, 5, 9)$ cycled

Figure 4: Both paths in $\hat{U}$ become cycles after the column permutation.

Lemma 3.7. *If there exists a path $j_k \rightsquigarrow j_l$ in G' for some $k \neq l$, then G_Q contains a cycle.*

Proof. Assume that for some $k \neq l$ there is a path $j_k \rightsquigarrow j_l$ in G'. We can assume w.l.o.g. that the intermediate nodes do not contain any node of $\{j_0, \ldots, j_n\}$.

If $l = k + 1$ then the path contains at least one intermediate node j because (j_k, j_{k+1}) is not an edge in G'. Hence $j_k \rightsquigarrow j \rightsquigarrow j_{k+1}$ is a path in G'. It follows from Lemma 1.1 that these edges form paths $j_k \rightsquigarrow j$ and $j \rightsquigarrow j_k$ in G_Q, so that G_Q contains a cycle.

If $l \neq k + 1$ then we can say that $k + 2 \leq l \leq n + 1$ since $\hat{U}$ is upper triangular except for column j_{n+1}. It follows from Lemma 1.1 that $j_k \rightsquigarrow j_{l-1}$ is a path in G_Q. Since by Lemma 3.6 $j_{l-1} \rightsquigarrow j_k$ is also a path in G_Q, G_Q contains a cycle. □

Proof of Theorem 3.5. Since the diagonal of $\hat{U}Q$ is zero-free, $\hat{U}$ is permuted triangular if and only if G_Q is acyclic. It will be shown that G_Q is acyclic if and only if (3a) and (3b) hold.

"$\Longrightarrow$" For the "only if" implication assume that (3a) or (3b) are violated. If (3a) is violated, then there exists a path $j_k \rightsquigarrow j_l$ in G' for some $l > k$. By Lemma 3.7 G_Q contains a cycle.

Now assume that (3b) is violated. Then there exists an index $j \neq j_n$, $j \in \hat{\mathcal{U}}_{j_0}$, and some $0 \leq k \leq n$ such that $j \in \mathrm{Reach}_{G'}(j_k)$. We have to distinguish two cases:

(i) If $j = j_k$ then $(j_k, j_0) \in E'$ and by Lemma 1.1 $(j_k, j_n) \in E_Q$. Because $j_n \rightsquigarrow j_k$ is a path in G_Q by Lemma 3.6, G_Q contains a cycle.

(ii) If $j \neq j_k$ then we can assume that $j \notin \{j_0, \ldots, j_n\}$ since otherwise Lemma 3.7 applies. Moreover, we can assume w.l.o.g. that $j_k \rightsquigarrow j$ is a path in G' whose intermediate nodes do not contain any node of $\{j_0, \ldots, j_n\}$. By Lemma 1.1 these edges form a path $j_k \rightsquigarrow j$ in G_Q. Because $(j, j_0) \in E'$ we have that $(j, j_n) \in E_Q$ and therefore $j_k \rightsquigarrow j \rightsquigarrow j_n$ is a path in G_Q. On the other hand $j_n \rightsquigarrow j_k$ is also a path in G_Q by Lemma 3.6. Hence G_Q contains a cycle.

"$\Longleftarrow$" It is shown that if G_Q contains a cycle, then (3a) or (3b) are violated. It must be distinguished whether or not j_n is in the cycle.

(i) Assume that there is a cycle in G_Q containing node j_n. Let (j, j_n) be an edge in the cycle. Then $j \in \mathrm{Reach}_{G_Q}(j_n)$. Because

$$\mathrm{Reach}_{G_Q}(j_n) \overset{Lemma\ 3.6}{=} \mathrm{Reach}_{G_Q}(\{j_0, \ldots, j_n\}) = \mathrm{Reach}_{G'}(\{j_0, \ldots, j_n\}),$$

we have that $j \in \mathrm{Reach}_{G'}(\{j_0, \ldots, j_n\})$. On the other hand, because (j, j_n) is a nonzero element in $\hat{U}Q$, we have that $j \in \hat{\mathcal{U}}_{j_0}$. It follows that (3b) is violated.

(ii) Assume that there is a cycle in G_Q not containing node j_n. Let $0 \leq k \leq n-1$ be the smallest k such that node j_k is in the cycle.

If another node j_l is in the cycle, then we can assume w.l.o.g. that $k+1 \leq l \leq n-1$ is such that the intermediate nodes in the path $j_k \rightsquigarrow j_l$ do not contain any node of $\{j_0, \ldots, j_n\}$. By Lemma 1.1 $j_k \rightsquigarrow j_{l+1}$ is a path in G', violating (3a).

On the other hand, assume that there is no node of $\{j_0, \ldots, j_n\}$ in the cycle other than j_k. Let (j, j_k) be an edge in the cycle. Hence $j_k \rightsquigarrow j \rightsquigarrow j_k$ is a path in G_Q. By Lemma 1.1 these edges form a path $j_k \rightsquigarrow j \rightsquigarrow j_{k+1}$ in G', violating (3a).

□

When $\hat{U}$ is permuted triangular, then the permutation to triangular form is obtained as follows:

Lemma 3.8. *Let $\hat{U}Q$ have a zero-free diagonal and be upper triangular except for columns $j_0, \ldots, j_n$. Let G_Q be the directed graph of $\hat{U}Q$. When $\hat{U}Q$ is permuted triangular, then $(\hat{U}Q)(\boldsymbol{p}, \boldsymbol{p})$ is upper triangular, where $\boldsymbol{p}$ is obtained from the identity permutation $(1, \ldots, m)$ by removing the indices in $\mathrm{Reach}_{G_Q}(\{j_0, \ldots, j_n\})$ and appending them in topological order to the end.*

Proof. Analogously to the proof of Lemma 3.2. □

4 Implementation

We have written a software package called BASICLU which factorizes a matrix B into its LU factors and combines our permutation method with the Forrest-Tomlin update to maintain the factorization after column modifications to B. This section describes the data structures and routines implementing the update scheme. Quantities which occur in the code are notated in typewriter font.

The factored form of the matrix B is

$$B = LR^1 \cdots R^r U, \tag{4}$$

where L is the symmetric permutation of a unit lower triangular matrix, the R^k are row eta matrices of the form (1) and U is permuted triangular. Note that the columns of L are indexed so that column i is computed in the factorization step in which row i is pivot row. Accordingly the row indices of U are (unpermuted) row indices of B. We have chosen this form rather than using permuted indices because updating U by an unsymmetric permutation changes the row-column pairings defined by the pivot elements.

The permutation vectors $\boldsymbol{p}$, $\boldsymbol{q}$ which make $U(\boldsymbol{p}, \boldsymbol{q})$ upper triangular are modified by each update. Instead of storing the permutation vectors, we store row-column mappings `pmap`, `qmap`, so that $i =$ `pmap`(j) and $j =$ `qmap`(i) when element (i, j) of U is a pivot element. To solve a linear system with U and a sparse right-hand side, knowing `pmap` and `qmap` is sufficient to apply the method of Gilbert and Peierls [5]. Since maintaining `pmap`, `qmap` is technically simpler than maintaining $\boldsymbol{p}$, $\boldsymbol{q}$, we assume for the moment that only sparse

linear systems have to be solved. A method for handling a dense right-hand side is presented below.

Algorithm 1 states the procedure for updating the factorization (4) without considering data structures. It implements the method from Section 3 to test if the spiked matrix is permuted triangular. Because before the update $U(\texttt{pmap}, \cdot)$ and $U(\cdot, \texttt{qmap})$ are symmetrically permuted triangular rather than U itself, the row-column mappings `pmap`, `qmap` must be taken into account in implementing (2) and (3).

The data structures for L and U must be designed to implement the update operations and to solve triangular systems efficiently. In particular, solving $L\boldsymbol{x} = \boldsymbol{b}$, $L^T\boldsymbol{x} = \boldsymbol{b}$, $U\boldsymbol{x} = \boldsymbol{b}$ and $U^T\boldsymbol{x} = \boldsymbol{b}$ for a sparse vector $\boldsymbol{b}$ requires rowwise and columnwise access to L and U. Because L is not modified by update operations, it can be stored in compressed column form and compressed row form. For U we decided to store $U(\texttt{pmap}, \cdot)$ rowwise and $U(\cdot, \texttt{qmap})$ columnwise. That means that rows in the rowwise storage are indexed by column indices of B and columns in the columnwise storage are indexed by row indices of B. This storage scheme allows to perform sparse triangular solves with U and U^T without redirection through `pmap` or `qmap`. To allow for fill-in, U must be stored with gaps between rows. The pivot elements are stored separately, one copy by row indices and one copy by column indices.

To solve the forward system $B\boldsymbol{x} = \boldsymbol{b}$, the inverse operations with L, $R^1, \ldots, R^r$ and $U(\cdot, \texttt{qmap})$ are carried out in the same work array without permutation. In the end the solution is permuted so that element i becomes element $\texttt{qmap}(i)$ of $\boldsymbol{x}$. The backward system $B^T\boldsymbol{x} = \boldsymbol{b}$ cannot be solved in a single work array. After the inverse operation with $U(\texttt{pmap}, \cdot)$ the solution must be permuted so that element j becomes element $\texttt{pmap}(j)$. The inverse operations with $R^r, \ldots, R^1$ and L are then carried out, giving the solution $\boldsymbol{x}$.

A factorization update is called "symmetric" when the diagonal element of the spike is nonzero. Lines 9–13 of Algorithm 1 implement condition (2) to test triangularity in the symmetric case. The implementation implicitly works on the matrix $U(\cdot, \texttt{qmap})$, which is symmetrically permuted triangular before inserting the spike into column i_0. It is exploited that $\boldsymbol{r}^T$ is computed before, whose nonzero pattern is $\text{Reach}(i_0) \setminus \{i_0\}$.

The triangularity test for the unsymmetric case is implemented in lines 16–32 of Algorithm 1. It works on the matrix $U(\texttt{pmap}, \cdot)$, which is symmetrically permuted triangular before inserting the spike into column j_0. The rowwise storage of this matrix allows to traverse the out-adjacency list of each node in its graph. To compute an augmenting path $(j_0, \ldots, j_n)$ a breadth first search is used because it gives a path of minimum length and thereby reduces the work in the subsequent loop. The reach in line 21 and 26 is determined by a depth first search. To operate on the graph G', the edges (j_k, j_{k+1}) are temporarily replaced by self-edges (j_k, j_k), which requires a search for index j_{k+1} in row j_k. When the spiked matrix is permuted triangular, then these elements become pivot elements and thus have to be replaced in the data structures for U.

We have ignored so far to maintain permutations $\boldsymbol{p}$, $\boldsymbol{q}$ which make $U(\boldsymbol{p}, \boldsymbol{q})$ upper triangular. They specify the order in which the columns respectively rows of U have to be processed in sequential triangular solves for $U\boldsymbol{x} = \boldsymbol{b}$ respectively $U^T\boldsymbol{x} = \boldsymbol{b}$. After a fresh factorization $\boldsymbol{p}$, $\boldsymbol{q}$ are stored as vectors of length m. As explained in Section 2 and Section 3, each update requires to move some indices to the end of $\boldsymbol{p}$, $\boldsymbol{q}$. Because this operation cannot be implemented efficiently, the implementation only appends these indices to the end, thereby generating duplicates and increasing the vector length.

In detail, the Forrest-Tomlin update appends i_0 and j_0 to the end of $\boldsymbol{p}$ and $\boldsymbol{q}$. The symmetric permutation update appends the indices in $\text{Reach}(i_0)$ in the graph of $U(\cdot, \texttt{qmap})$ in topological order to the end of $\boldsymbol{p}$. In Algorithm 1 they are readily available from the nonzero pattern of $\boldsymbol{r}^T$. The corresponding column indices (corresponding via `qmap`) are appended to the end of $\boldsymbol{q}$. Both the Forrest-Tomlin update and the update by symmetric permutation do not modify `pmap` or `qmap`.

In the unsymmetric permutation update, `pmap` is updated by a cyclic permutation of

Algorithm 1 *LU* factorization update

Require: $B = LR^1 \cdots R^r U$, `pmap`, `qmap`, vector $\boldsymbol{a}$ to replace column j_0 of B

Prepare update

$i_0 = \texttt{pmap}(j_0)$
Let $\boldsymbol{w}^T$ be row i_0 of U without entry in position j_0.
Compute $\boldsymbol{r}^T = \boldsymbol{w}^T U^{-1}$.
Compute $\hat{\boldsymbol{a}} = (R^r)^{-1} \cdots (R^1)^{-1} L^{-1} \boldsymbol{a}$.
Replace column j_0 of U by $\hat{\boldsymbol{a}}$.

Test triangularity

`istriangular` = **true**
if $\hat{a}_{i_0} \neq 0$ **then** // spike has nonzero diagonal element
    `issymmetric` = **true**
    for all i for which $\hat{a}_i \neq 0$ **do**
        if $r_i \neq 0$ **then**
            `istriangular` = **false**
        end if
    end for
else // spike has zero diagonal element
    `issymmetric` = **false**
    Find augmenting path $(j_0, \ldots, j_n)$ in $U(\texttt{pmap}, \cdot)$.
    $j_{n+1} = j_0$
    Let G' be the directed graph of $U(\texttt{pmap}, \cdot)$ without the edges (j_k, j_{k+1}), $0 \leq k \leq n$.
    Assume that all nodes are unmarked.
    for $k = 0$ **to** $n - 1$ **do**
        Mark nodes in $\text{Reach}_{G'}(j_k)$.
        if node j_{k+1} is marked **then**
            `istriangular` = **false**
        end if
    end for
    Mark nodes in $\text{Reach}_{G'}(j_n)$.
    for all i for which $\hat{a}_i \neq 0$ **do**
        $j = \texttt{qmap}(i)$
        if $j \neq j_n$ **and** node j is marked **then**
            `istriangular` = **false**
        end if
    end for
end if

Apply update

if `istriangular` **then** // spiked matrix is permuted triangular
    if not `issymmetric` **then**
        $\texttt{pmap}(j_{k+1}) = \texttt{pmap}(j_k)$ for $0 \leq k \leq n$. // simultaneous assignment
        $\texttt{qmap}(\texttt{pmap}(j_k)) = j_k$ for $0 \leq k \leq n$.
    end if
else // Forrest-Tomlin update
    Compute pivot element $x_0 = \hat{a}_{i_0} - \boldsymbol{r}^T \hat{\boldsymbol{a}}$.
    Replace row i_0 of U by singleton row with nonzero x_0 in position j_0.
    Append row eta matrix R^{r+1} with off-diagonals in row i_0 given by $\boldsymbol{r}^T$.
end if

positions $(j_0, \ldots, j_n)$ and `qmap` is updated as its inverse permutation (lines 36–37 in Algorithm 1). Then the indices in $\mathrm{Reach}_{G^*}(\{j_0, \ldots, j_n\})$ are appended in topological order to the end of $\boldsymbol{q}$, where G^* is the directed graph of $U(\texttt{pmap}, \cdot)$. By implementing line 21 and 26 of Algorithm 1 through a depth first search, the indices in $\mathrm{Reach}_{G^*}(\{j_0, \ldots, j_n\})$ can be obtained in topological order as a by-product. The corresponding row indices (corresponding via `pmap`) are appended to the end of $\boldsymbol{p}$.

After a sequence of updates, the resulting permutation vectors $\boldsymbol{p}$, $\boldsymbol{q}$ have length $L > m$. In triangular solves for $U^T\boldsymbol{x} = \boldsymbol{b}$ and $U\boldsymbol{x} = \boldsymbol{b}$ they are traversed from the end. The solves are carried out in a work array of length m which initially holds the right-hand side. When an index is retrieved from the permutation vector for the first time, the computation for its solution component has been completed and the value is stored in an output array. Its position in the work array is set to zero. When an index occurs multiple times, then only at the first occurence its value in the work array is (possibly) nonzero. All later occurences will be skipped.

When the length of $\boldsymbol{p}$, $\boldsymbol{q}$ becomes too large, say $2m$, then garbage collection is done to remove duplicates.

5 Results

The most apparent application of the presented LU update is the revised simplex method. For many large scale linear programming problems the simplex basis matrices are "nearly" permuted triangular. On these problems it can be expected that for a significant proportion of basis updates the LU factors can be updated by permutation. This section investigates the efficiency of our LU update on this class of matrices.

We use the same set of LP problems as in [6] since it represents a wide range of properties of practical problems. Each problem is solved with the dual simplex solver of CPLEX 12.7 and the pivot sequence is recorded. CPLEX is run without presolve, starting from the slack basis and using dual steepest edge pricing. The other parameters are defaults. Problem Linf_520c is removed from the test set because with these parameters it is not solved within 12 hours (it seems that printing out the pivot sequence significantly slows down CPLEX). Table 1 lists the problems with their dimensions and the number of simplex iterations. The column headed "Bump" is the average bump size of the basis matrices.

Our LU code is applied to the basis matrices dictated by the pivot sequence. For each basis matrix B the forward and backward systems

$$B\boldsymbol{x} = \boldsymbol{a}, \quad B^T\boldsymbol{y} = \boldsymbol{e}_p$$

are solved, where $\boldsymbol{a}$ is the vector that replaces column p of B. For each system the first triangular solve is done by the method of Gilbert and Peierls and the solution is stored for the update. The second triangular solve is done by the same method if its right-hand side has not more than $.05m$ nonzeros. Otherwise the system is solved by a sequential pass through the vector.

Our LU code chooses the refactorization frequency depending on the cost of the factorization and the cost of operations with the row eta matrices that are accumulated by updates. To get reproducable results, the costs are measured in terms of number of nonzeros and number of floating point operations rather than CPU time.

Table 2 lists the results obtained from the combined permutation/Forrest-Tomlin update and the pure Forrest-Tomlin update. For the latter, only the permutation test is skipped in our implementation. The column headed "Perm (Sym)" is the fraction of updates which could be done by permutation (symmetric permutation). Unsurprisingly there is a clear correlation between this quantity and a small bump size. The relation between symmetric and unsymmetric permutations varies heavily.

Name	Rows	Nz/Col	Iter	Bump
cre-b	9648	3.53	18559	137
dano3mip_lp	3202	5.74	26936	1116
dcp2	32388	26.53	24740	3331
dfl001	6071	2.91	21425	2319
fome12	24284	2.91	89316	9201
gen4	1537	24.92	935	243
ken-18	105127	2.32	202898	5
l30	2701	3.33	12326	2299
lp22	2958	4.88	18186	1381
maros-r7	3136	15.40	4863	765
mod2	34774	5.20	51378	2841
ns1688926	32768	103.22	7491	70
osa-60	10280	6.00	8752	4
pds-20	33874	2.18	36123	146
pds-40	66844	2.17	90999	536
pds-100	156243	2.15	241451	654
pilot87	2030	14.98	15659	991
qap12	3192	4.33	220281	2708
self	960	156.01	11015	790
sgpf5y6	246077	2.68	261456	275
stat96v1	5995	2.98	18138	4875
stat96v4	3173	7.88	66193	2764
stormg2-125	66185	2.66	119435	529
stormg2_1000	528185	2.65	986842	3646
stp3d	159488	3.23	116569	3302
truss	1000	3.16	19581	700
watson_1	201155	2.74	299832	1904
watson_2	352013	2.74	451528	2575
world	34506	5.02	55752	2467

Table 1: LP test problems.

The column headed "Ratio" is the total computation time of the combined update divided by the total computation time of the Forrest-Tomlin update. The update times of the combined approach are about a factor 2 higher than those of the Forrest-Tomlin update. However, since the total computation time is dominated by the solve and factorization times, overall the combined update is never significantly slower than the Forrest-Tomlin update. On those problems where many updates can be permuted, the combined approach reduces both the factorization time and the solve time significantly.

Note

During the final work on this report it came to our knowledge that the same idea has been recently investigated by Fukasawa and Poirrier [4]. The theory presented here and the software package BASICLU have been developed independently of their work.

References

[1] R. H. Bartels. A stabilization of the simplex method. *Numer. Math.*, 16:414–434, 1970/1971.

[2] I. S. Duff. On algorithms for obtaining a maximum transversal. *ACM Trans. Math. Softw.*, 7(3):315–330, September 1981.

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

[4] R. Fukasawa and L. Poirrier. Permutations in the factorization of simplex bases. http://www.optimization-online.org/DB_HTML/2016/12/5770.html. submitted Dec 13, 2016; accessed May 8, 2017.

[5] J. R. Gilbert and T. Peierls. Sparse partial pivoting in time proportional to arithmetic operations. *SIAM J. Sci. Statist. Comput.*, 9(5):862–874, 1988.

[6] Q. Huangfu and J. A. J. Hall. Novel update techniques for the revised simplex method. *Comput. Optim. Appl.*, 60(3):587–608, 2015.

[7] J. K. Reid. A sparsity-exploiting variant of the Bartels-Golub decomposition for linear programming bases. *Math. Programming*, 24(1):55–69, 1982.

Name	Perm (Sym)	Combined				Forrest-Tomlin				Ratio
		n_f	t_f	t_s	t_u	n_f	t_f	t_s	t_u	
cre-b	0.94 (0.04)	34	0.04	0.21	0.03	49	0.05	0.31	0.02	0.74
dano3mip_lp	0.28 (0.01)	223	0.48	3.03	0.15	223	0.48	3.06	0.08	1.02
dcp2	0.41 (0.34)	39	0.33	1.57	0.06	41	0.34	1.66	0.05	0.96
dfl001	0.53 (0.16)	103	0.32	3.54	0.09	108	0.32	3.62	0.05	0.99
fome12	0.52 (0.15)	237	3.18	19.60	0.49	246	3.16	20.46	0.25	0.97
gen4	0.02 (0.01)	7	0.13	0.40	0.05	7	0.14	0.41	0.02	1.05
ken-18	0.99 (0.72)	14	0.26	0.84	0.24	143	2.42	2.25	0.08	0.28
l30	0.04 (0.03)	102	0.51	1.83	0.13	102	0.52	1.83	0.04	1.03
lp22	0.17 (0.06)	143	0.79	2.59	0.21	144	0.81	2.53	0.07	1.06
maros-r7	0.38 (0.29)	31	0.10	0.37	0.03	36	0.11	0.38	0.02	0.98
mod2	0.67 (0.28)	118	0.88	11.30	0.19	123	0.85	12.12	0.09	0.95
ns1688926	0.51 (0.48)	18	0.29	6.49	0.21	18	0.29	6.43	1.33	0.87
osa-60	0.96 (0.02)	16	0.01	0.15	0.01	23	0.02	0.31	0.04	0.49
pds-20	0.97 (0.42)	21	0.08	0.45	0.06	76	0.24	0.56	0.02	0.73
pds-40	0.96 (0.42)	50	0.49	3.11	0.23	152	1.24	4.05	0.07	0.71
pds-100	0.97 (0.41)	90	2.32	8.72	0.91	337	7.31	10.80	0.20	0.65
pilot87	0.13 (0.10)	84	0.89	2.10	0.24	84	0.88	2.12	0.22	1.01
qap12	0.05 (0.00)	1170	29.70	60.53	5.41	1176	30.39	62.30	1.86	1.01
self	0.02 (0.01)	136	13.68	4.40	0.94	138	14.08	4.96	1.04	0.95
sgpf5y6	0.99 (0.17)	38	1.57	2.41	0.40	182	6.69	6.92	0.15	0.32
stat96v1	0.14 (0.13)	99	0.63	4.68	0.13	100	0.62	4.60	0.06	1.03
stat96v4	0.19 (0.18)	333	2.26	9.23	0.36	335	2.27	9.23	0.26	1.01
stormg2-125	0.94 (0.26)	45	0.44	0.42	0.08	103	0.98	0.86	0.05	0.50
stormg2_1000	0.95 (0.22)	127	14.78	18.34	1.01	308	33.39	31.61	0.70	0.52
stp3d	0.84 (0.31)	120	3.26	45.10	0.74	129	3.26	53.10	0.43	0.86
truss	0.29 (0.10)	181	0.11	0.47	0.04	186	0.12	0.46	0.03	1.02
watson_1	0.93 (0.53)	114	4.28	2.62	0.22	156	5.67	4.67	0.18	0.68
watson_2	0.94 (0.58)	129	8.71	4.36	0.32	176	11.37	8.04	0.26	0.68
world	0.70 (0.28)	127	0.86	10.23	0.17	134	0.86	11.77	0.10	0.89

Table 2: Comparison of LU update methods in terms of number of factorizations (n_f), factorization time (t_f), solve time (t_s) and update time (t_u) in seconds.