Strathprints Institutional Repository

Pestana, Jennifer and Wathen, Andrew (2014) The antitriangular factorisation of saddle point matrices. SIAM Journal on Matrix Analysis and Applications, 35. 339–353. ISSN 0895-4798 , http://dx.doi.org/10.1137/130934933

THE ANTITRIANGULAR FACTORISATION OF SADDLE POINT MATRICES

J. PESTANA AND A. J. WATHEN

Abstract. Mastronardi and Van Dooren [this journal, 34 (2013) pp. 173–196] recently introduced the block antitriangular ("Batman") decomposition for symmetric indefinite matrices. Here we show the simplification of this factorisation for saddle point matrices and demonstrate how it represents the common nullspace method. We show that rank-1 updates to the saddle point matrix can be easily incorporated into the factorisation and give bounds on the eigenvalues of matrices important in saddle point theory. We show the relation of this factorisation to constraint preconditioning and how it transforms but preserves the structure of block diagonal and block triangular preconditioners.

1. Introduction. The antitriangular factorisation proposed by Mastronardi and Van Dooren [17] converts a symmetric indefinite matrix $H \in \mathbb{R}^{p\times p}$ into a block antitriangular matrix M using orthogonal similarity transforms. The factorisation can be performed in a backward stable manner and linear systems with the block antitriangular matrix may be efficiently solved. Moreover, the orthogonal similarity transforms preserve eigenvalues and reveal the inertia of H. Thus, from M one can determine the triple (n_-, n_0, n_+) of H, where n_- is the number of negative eigenvalues, n_0 is the number of zero eigenvalues and n_+ is the number of positive eigenvalues.

The antitriangular factorisation takes the form

$$H = QMQ^T,\ Q^{-1} = Q^T,\ M = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & Y^T \\ 0 & 0 & X & Z^T \\ 0 & Y & Z & W \end{bmatrix} \begin{matrix} \}n_0 \\ \}n_1 \\ \}n_2 \\ \}n_1 \end{matrix}, \tag{1.1}$$

where $n_1 = \min(n_-, n_+)$, $n_2 = \max(n_-, n_+) - n_1$, $Z \in \mathbb{R}^{n_1\times n_2}$, $W \in \mathbb{R}^{n_1\times n_1}$ is symmetric, $X = \epsilon LL^T \in \mathbb{R}^{n_2\times n_2}$ is symmetric definite whenever $n_2 > 0$ and $Y \in \mathbb{R}^{n_1\times n_1}$ is nonsingular and antitriangular, so that entries above the main antidiagonal are zero. Additionally,

$$\epsilon = \begin{cases} 1 & \text{if } n_+ > n_- \\ -1 & \text{if } n_- > n_+. \end{cases}$$

The matrix M is strictly antitriangular whenever $n_2 = 0, 1$, i.e., whenever the number of positive and negative eigenvalues differs by at most one. However, the "bulge" X increases in dimension as H becomes closer to definite. In the extreme case that H is symmetric positive (or negative) definite $n_0 = n_1 = 0$, i.e., X is itself a $p \times p$ matrix. Accordingly, the antitriangular factorisation is perhaps best suited to matrices that have a significant number of both positive and negative eigenvalues. We emphasise, however, its generality for real symmetric matrices.

Saddle point matrices are symmetric and indefinite, so that the antitriangular factorisation can be applied. These matrices arise in numerous applications [2, Section 2] and have the form

$$\mathcal{A} = \begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{matrix} \}n \\ \}m \end{matrix}, \tag{1.2}$$

where $A \in \mathbb{R}^{n\times n}$ is symmetric (but not necessarily positive definite) and $B \in \mathbb{R}^{m\times n}$, $m \le n$. The matrix $\mathcal{A}$ is nonsingular with n positive eigenvalues and m negative eigenvalues when A is positive definite on the nullspace of B and $\text{rank}(B) = m$. We only consider this most common situation here.

The algorithm for computing an antitriangular factorisation proposed by Mastronardi and Van Dooren is designed to be applicable to *all* symmetric indefinite matrices. In this note we show that their algorithm simplifies when applied to saddle point matrices. An alternative based on a QR factorisation of B^T, that is like the approach applied by Mastronardi and Van Dooren to specific saddle point problems arising in constrained indefinite least squares [16], gives a different but related antitriangular form. Both algorithms are shown to be strongly backward stable but the optimal algorithm in terms of cost depends on the sizes of m and n.

Low-rank updates of A and B in $\mathcal{A}$, such as those used in quasi-Newton methods [8], interior point methods [1] or the augmented Lagrangian method [2, Section 3.5], can be efficiently incorporated into an antitriangular factorisation of $\mathcal{A}$. Additionally, bounds on the eigenvalues of A and the (negative) Schur complement $BA^{-1}B^T$ that depend only on the smaller blocks W, X and Y of the antitriangular matrix can be obtained.

We show that solving a saddle point system in antitriangular form is equivalent to applying the nullspace method [2, Section 6][21, Section 15.2]. In other words, the antitriangular factorisation allows the nullspace method to be represented not just as a procedure but also as a matrix decomposition, similarly to other well known methods for solving linear systems like Gaussian elimination.

If the matrix $\mathcal{A}$ is large, we may solve the saddle point system by an iterative method rather than a direct method like the antitriangular factorisation. When preconditioning is required block preconditioners, such as block diagonal, block triangular and constraint preconditioners, are popular choices for saddle point systems. We show that the same orthogonal transformation matrix that converts $\mathcal{A}$ into an antitriangular matrix can be applied to these preconditioners and that relevant structures are preserved.

The outline of our manuscript is as follows. The two algorithms are given in Section 2 where their complexities are also compared. Stability, extensions and low-rank updates are discussed in Section 3 while the connection to the nullspace method is outlined in Section 4. We state our eigenvalue bounds in Section 5 and discuss preconditioners in Section 6. Finally, Section 7 contains our conclusions.

Throughout,we use Matlab notation to denote submatrices. Thus $K(q:r, s:t)$ is the submatrix of K comprising the intersection of rows q to r with columns s to t. Also, $K(r:-1:q, s:t)$ (or $K(q:r, t:-1:s)$) represents the submatrix $K(q:r, s:t)$ with its rows (or columns) in reverse order. The nullspace and range of a matrix K are denoted by $\text{null}(K)$ and $\text{range}(K)$, respectively.

2. An antitriangular factorisation of saddle point matrices. We are interested in applying orthogonal transformations to the saddle point matrix $\mathcal{A}$ in (1.2) to obtain the antitriangular matrix (1.1). Since $\mathcal{A}$ is nonsingular with n positive eigenvalues and m negative eigenvalues, in this case the antitriangular matrix has the specific form

$$\mathcal{M} = \begin{bmatrix} 0 & 0 & Y^T \\ 0 & X & Z^T \\ Y & Z & W \end{bmatrix} \begin{matrix} \}\, m \\ \}\, n-m \\ \}\, m \end{matrix} \quad ,$$

where $Y \in \mathbb{R}^{m \times m}$ is antitriangular, $X \in \mathbb{R}^{(n-m)\times(n-m)}$ is symmetric positive definite and $W \in \mathbb{R}^{m \times m}$ is symmetric. We note that linear systems with $\mathcal{M}$ can be solved with the obvious "antitriangular" substitution (finding the last variable from the first equation, the second-last variable from the second equation and so forth) twice, with a solve with the positive definite matrix X (using, for example, a Cholesky factorisation) in between.

2.1. The algorithm of Mastronardi and Van Dooren. Although it is possible to compute an antitriangular factorisation of (1.2) by the algorithm of Mastronardi and Van Dooren, the result is somewhat more involved than necessary since the algorithm simplifies if we first first permute $\mathcal{A}$ to

$$\widetilde{\mathcal{A}} = Q_1^T \mathcal{A} Q_1 = \begin{bmatrix} 0 & B \\ B^T & A \end{bmatrix} \begin{matrix} \}\, m \\ \}\, n \end{matrix}, \quad Q_1 = \begin{bmatrix} 0 & I_n \\ I_m & 0 \end{bmatrix}. \tag{2.1}$$

Given the permuted matrix (2.1), the algorithm of Mastronardi and Van Dooren proceeds outwards from the (1,1) entry of $\widetilde{\mathcal{A}}$, at each stage updating the antitriangular factorisation of $\widetilde{\mathcal{A}}(1:k, 1:k)$ to give a factorisation of $\widetilde{\mathcal{A}}(1:k+1, 1:k+1)$. Accordingly, the first m steps leave $\widetilde{\mathcal{A}} = Q_1^T \mathcal{A} Q_1$ unchanged and the inertia of $\widetilde{\mathcal{A}}(1:m, 1:m)$ is $(0, m, 0)$.

The next stage of the algorithm uses Householder matrices to convert B to an antitrapezoidal matrix

$$HB = V = \begin{bmatrix} V^{(1)} & V^{(2)} \end{bmatrix}, \tag{2.2}$$

where $H \in \mathbb{R}^{m\times m}$ is orthogonal, $V^{(1)} \in \mathbb{R}^{m\times m}$ is antitriangular and $V^{(2)} \in \mathbb{R}^{m\times(n-m)}$. Accordingly, after a further m steps, we obtain

$$Q_2^T Q_1^T \mathcal{A} Q_1 Q_2 = \begin{bmatrix} 0 & V \\ V^T & A \end{bmatrix}, \quad Q_2 = \begin{bmatrix} H^T & \\ & I_n \end{bmatrix},$$

and the inertia of the submatrix formed from the first $2m$ rows and columns of $Q_2^T Q_1^T \mathcal{A} Q_1 Q_2$ is $(m, 0, m)$.

If $n = m$ we are finished, although this situation is rare in practice. Otherwise, we must reduce V to antitriangular form by Givens rotations from the right. Thus,

$$VG = V[G^{(1)} \quad G^{(2)}] = [0 \quad Y^T], \tag{2.3}$$

where $G \in \mathbb{R}^{n\times n}$, $G^{(1)} \in \mathbb{R}^{n\times(n-m)}$, $Y \in \mathbb{R}^{m\times m}$. It follows from (2.2) that

$$B \begin{bmatrix} G^{(1)} & G^{(2)} \end{bmatrix} = \begin{bmatrix} 0 & (YH)^T \end{bmatrix}$$

and that $G^{(1)}$ and $G^{(2)}$ are bases for null(B) and range(B^T), respectively. Thus, applying

$$Q_3 = \begin{bmatrix} I_m & \\ & G \end{bmatrix}$$

gives the antitriangular form

$$\mathcal{M}_1 = \mathcal{Q}_1^T \mathcal{A} \mathcal{Q}_1 = \begin{bmatrix} 0 & 0 & Y_1^T \\ 0 & X_1 & Z_1^T \\ Y_1 & Z_1 & W_1 \end{bmatrix}, \quad \mathcal{Q}_1 = Q_1 Q_2 Q_3 = \begin{bmatrix} 0 & G^{(1)} & G^{(2)} \\ H^T & 0 & 0 \end{bmatrix},$$

where $Z_1 = (G^{(2)})^T A G^{(1)} \in \mathbb{R}^{m\times(n-m)}$, $W_1 = (G^{(2)})^T A G^{(2)} \in \mathbb{R}^{m\times m}$ is symmetric and $X_1 = (G^{(1)})^T A G^{(1)} \in \mathbb{R}^{(n-m)\times(n-m)}$ is symmetric positive definite, since A is

Algorithm 1: Antitriangular factorisation of a saddle point matrix by the algorithm of Mastronardi and Van Dooren.

Input: Saddle point matrix $\mathcal{A}$ from (1.2)
Output: Antitriangular matrix $\mathcal{M}_1$ and orthogonal matrix $\mathcal{Q}_1$ such that $\mathcal{A} = \mathcal{Q}_1\mathcal{M}_1\mathcal{Q}_1^T$
Permute the rows and columns of $\mathcal{A}$ as in (2.1)
Compute an upper trapezoidal factorisation of B as in (2.2) by Householder matrices
Compute the Givens rotations that transform V to antitriangular form $VG = [0 \quad Y_1^T]$ as in (2.3)
Set $G^{(1)} = G(1:n, 1:n-m)$ and $G^{(2)} = G(1:n, n-m+1:n)$
Set $X_1 = (G^{(1)})^T AG^{(1)}$, $Z_1 = (G^{(2)})^T AG^{(1)}$ and $W_1 = (G^{(2)})^T AG^{(2)}$
Set $\mathcal{M}_1 = \begin{bmatrix} 0 & 0 & Y_1^T \\ 0 & X_1 & Z_1^T \\ Y_1 & Z_1 & W_1 \end{bmatrix}$ and $\mathcal{Q}_1 = \begin{bmatrix} 0 & G^{(1)} & G^{(2)} \\ H^T & 0 & 0 \end{bmatrix}$

positive definite on the nullspace of B. An algorithm for this procedure is given in Algorithm 1.

Note that in this case we avoid the more complex case c in the Mastronardi and Van Dooren algorithm since, although A may not be definite, the positive definite part of A is automatically obtained in the process of antitriangularising B and B^T. In contrast, applying the algorithm to (1.2) would involve this third case.

2.2. An alternative. By reordering the operations, we obtain an alternative to Algorithm 1 which instead involves only permutations and a QR factorisation of B^T,

$$B^T = \underbrace{\begin{bmatrix} U^{(1)} & U^{(2)} \end{bmatrix}}_{U} \begin{bmatrix} R \\ 0 \end{bmatrix}, \tag{2.4}$$

where $U \in \mathbb{R}^{n\times n}$ and $R \in \mathbb{R}^{m\times m}$. Note that now the columns of $U^{(1)} \in \mathbb{R}^{n\times m}$ form an orthonormal basis for range(B^T) while the columns of $U^{(2)}$ form an orthonormal basis for null(B).

As in the previous algorithm we start from (2.1) but now apply

$$Q_4 = \begin{bmatrix} I_m & 0 \\ 0 & U \end{bmatrix}$$

to obtain

$$Q_4^T Q_1^T \mathcal{A} Q_1 Q_4 = \begin{array}{c} \begin{bmatrix} 0 & R^T & 0 \\ R & \widehat{A}_{11} & \widehat{A}_{12} \\ 0 & \widehat{A}_{12}^T & \widehat{A}_{22} \end{bmatrix} \end{array} \begin{array}{l} \}\, m \\ \}\, m \\ \}\, n-m \end{array}, \tag{2.5}$$

where $\widehat{A}_{ij} = (U^{(i)})^T AU^{(j)}$, $i, j = 1, 2$. Note that $\widehat{A}_{22} = (U^{(2)})^T AU^{(2)}$ is positive definite, analogously to the Mastronardi and Van Dooren algorithm.

Then all that remains is to permute the last $n-m$ rows and columns so that R is transformed to an antitriangular matrix that sits in the last m rows. This is achieved by applying

$$
(2.6) \qquad Q_5 = \begin{bmatrix} I_m & 0 \\ 0 & \widehat{S} \end{bmatrix}, \quad \widehat{S} = \begin{bmatrix} 0 & S_m \\ I_{n-m} & 0 \end{bmatrix}, \quad S_m = \begin{bmatrix} & & 1 \\ & \iddots & \\ 1 & & \end{bmatrix}.
$$

The matrix S_m is the $m \times m$ reverse identity, which satisfies $S_m^{-1} = S_m^T = S_m$.

Combining these steps gives

$$
(2.7) \quad \mathcal{M}_2 = \mathcal{Q}_2^T \mathcal{A} \mathcal{Q}_2 = \begin{bmatrix} 0 & 0 & Y_2^T \\ 0 & X_2 & Z_2^T \\ Y_2 & Z_2 & W_2 \end{bmatrix}, \quad \mathcal{Q}_2 = Q_1 Q_4 Q_5 = \begin{bmatrix} 0 & U^{(2)} & U^{(1)} S_m \\ I_m & 0 & 0 \end{bmatrix},
$$

where $Y_2 = S_m R \in \mathbb{R}^{m \times m}$, $Z_2 = S_m \widehat{A}_{12}$, $W_2 = S_m \widehat{A}_{11} S_m$ is symmetric and $X_2 = \widehat{A}_{22} \in \mathbb{R}^{(n-m)\times(n-m)}$ is symmetric positive definite. We summarise this method in Algorithm 2.

Algorithm 2: Antitriangular factorisation of a saddle point matrix using the QR factorisation.

Input: Saddle point matrix $\mathcal{A}$ from (1.2)

Output: Antitriangular matrix $\mathcal{M}_2$ and orthogonal matrix $\mathcal{Q}_2$ such that $\mathcal{A} = \mathcal{Q}_2 \mathcal{M}_2 \mathcal{Q}_2$

Permute the rows and columns of $\mathcal{A}$ as in (2.1)

Compute the QR factorisation $B^T = UR$

Set $U^{(1)} = U(1:n, 1:m)$ and $U^{(2)} = U(1:n, m+1:n)$

Compute $\widehat{A}_{11} = (U^{(1)})^T A U^{(1)}$, $\widehat{A}_{12} = (U^{(1)})^T A U^{(2)}$ and $\widehat{A}_{22} = (U^{(2)})^T A U^{(2)}$

Set $Y_2 = R(m:-1:1, 1:m)$, $X_2 = \widehat{A}_{22}$, $Z_2 = \widehat{A}_{12}(m:-1:1, 1:n-m)$ and $W_2 = A_{11}(m:-1:1, m:-1:1)$

Set $\mathcal{M}_2 = \begin{bmatrix} 0 & 0 & Y_2^T \\ 0 & X_2 & Z_2^T \\ Y_2 & Z_2 & W_2 \end{bmatrix}$ and $\mathcal{Q}_2 = \begin{bmatrix} 0 & U^{(2)} & U^{(1)}(1:n, m:-1:1) \\ I_m & 0 & 0 \end{bmatrix}$

2.3. Complexity of the antitriangular algorithms. Both Algorithms 1 and 2 start from the permuted matrix $Q_1^T \mathcal{A} Q_1$ in (2.1) and convert B and B^T to antitriangular form. Algorithm 1 achieves this by a two-sided orthogonal transformation $HBG = [0 \quad Y_1^T]$, where H gives the intermediate antitrapezoidal form (2.2). Algorithm 2 instead uses the one-sided transformation $BU\widehat{S} = [0 \quad Y_2^T]$. Thus, the differences between the algorithms are due to the choice of one-sided or two-sided transformations and the optimal choice in terms of floating point operations depends on the ratio of n to m as we now show.

The Mastronardi and Van Dooren algorithm first uses Householder transforms to convert B to Y and this requires $2m^2(n - m/3)$ flops [6, Section 5.2.1]. If $n > m$ we must apply $n - m$ sequences of m Givens rotations to convert V to the correct antitriangular form. Each sequence annihilates an antidiagonal of V and alters m columns of $Q_2^T Q_1^T \mathcal{A} Q_1 Q_2$. The total number of flops required to apply these Givens rotations from the right is

$$
6 \sum_{j=1}^{n-m} \sum_{i=1}^{m} (n + m - i + 1) \approx 6n^2 m - 3nm^2 - 3m^3.
$$

Algorithm	Dimensions	Flops
Algorithm 1	$n = m$	$2m^2 n - \frac{2}{3}m^3$
	$m < n < 2m$	$3n^3 - 3mn^2 + 8m^2 n - \frac{20}{3}m^3$
	$n = 2m$	$6mn^2 + 2m^2 n - \frac{20}{3}m^3$
	$n > 2m$	$12n^2 m - 10m^2 n + \frac{16}{3}m^3$
Algorithm 2	$n \geq m$	$8mn^2 - 2m^2 n - \frac{2}{3}m^3$

TABLE 2.1
Complexity of Algorithms 1 and 2.

Although, by exploiting symmetry, we can update V^T and $A(m+1:n, 1:m)$ without additional computations, we must still apply Givens rotations to the rows of $A(m+1:n, m+1;n)$. The number of operations depends on the size of $n-m$ compared with the size of m. As the first antidiagonal of V is annihilated, we apply $3(n-m)$ operations to the rows of $A(m+1:n, m+1;n)$, at the second $6(n-m)$, at the third $12(n-m)$ and so on until either we reach the last row of the matrix or we have applied m sequences of Givens rotations. This requires

$$3(n-m) + 6(n-m)\sum_{j=1}^{r-1} i, \quad r = \min\{n-m, m\}$$

flops or, to leading order, $3(n-m)^3$ when $n < 2m$ and $3(n-m)m^2$ otherwise. If $n > 2m$ we must apply additional Givens rotations to make V and V^T antitriangular at a cost of $6m(n-m)(n-2m)$ flops. The total flop counts for these different cases are given in Table 2.1.

The cost of Algorithm 2, which involves only the QR factorisation of B^T and the formation of $U^T AU$ can also be determined. The QR decomposition of B^T requires $2m^2(n-m/3)$ flops if Householder transformations are used. Then $U^T A$ can be computed in approximately $2mn(2n-m)$ flops [6, Section 5.1.6] and similarly for $(U^T A)U$. Thus, the total cost of computing the antitriangular factorisation by Algorithm 2 is approximately $8mn^2 - 2m^2(n+m/3)$ flops.

From this comparison it is clear that the optimal algorithm depends on the size of m, the number of constraints, relative to the number of primal variables n. If m is almost as large as n Algorithm 2 is favourable while Algorithm 1 is better when m is small relative to n.

Unless otherwise stated, we concentrate on the QR-variant (Algorithm 2) in the remainder of this manuscript for ease of exposition, but the same analysis could easily be applied to the antitriangular matrix from Algorithm 1. We additionally drop subscripts on the matrices $\mathcal{M}$ $\mathcal{Q}$, W, X, Y and Z.

3. Properties of the antitriangular decomposition.

In this section we discuss properties of the antitriangular decomposition of saddle point matrices, including stability, extensions and low-rank modifications.

3.1. Stability.

Algorithms 1 and 2 are not only backward stable (provided the QR decomposition in Algorithm 2 is computed in a backward stable manner) but are strongly stable, in the sense of Sun [25], i.e., the computed matrices $\mathcal{Q}$ and $\mathcal{M}$ satisfy

$$\begin{bmatrix} A + \Delta A & B^T + \Delta B^T \\ B + \Delta B & 0 \end{bmatrix} = \mathcal{Q}\mathcal{M}\mathcal{Q}^T,$$

where $\|\Delta A\|_2/\|A\|_2 = O(\epsilon_m)$, $\|\Delta B\|_2/\|B\|_2 = O(\epsilon_m)$, $\|\Delta B^T\|_2/\|B^T\|_2 = O(\epsilon_m)$ and ϵ_m is machine precision. To prove this we first note that the antitriangular decomposition comprises two parts: the antitriangular factorisation of B and B^T, and the multiplication of A by orthogonal matrices. The factorisation of B and B^T by either Algorithm 1 or 2 is backward stable and the computed matrices $\bar{G}$ and $\bar{U}$ satisfy $\|\bar{G}-G\|_2 = O(\epsilon_m)$ and $\|\bar{U}-U\|_2 = O(\epsilon_m)$, for some orthogonal G and U; this is proved in a similar way to the QR factorisation results in the book by Higham [9, Chapter 19]. Additionally, multiplication by an orthogonal matrix is backward stable [9, Section 3.5]. Consequently, neither algorithm should have problems with breakdown but if B is numerically rank deficient then Y will be as well, as expected.

3.2. Extensions. Although we consider only real matrices, since this is the most prevalent case in practice, the extension of Algorithms 1 and 2 to complex Hermitian matrices is trivial if we apply unitary matrices instead of orthogonal ones.

We can also find a factorisation of non-Hermitian matrices, such as block complex symmetric matrices, i.e., matrices of the form (1.2) but with $A \in \mathbb{C}^{n\times n}$, $A = A^T$ and $B \in \mathbb{C}^{m\times n}$. Such matrices arise in, for example, electrical networks [2, page 5][11][15]. In this setting $\mathcal{Q}$ is complex, but the complex symmetry is preserved, i.e., the resulting antitriangular matrix is complex symmetric. It no longer makes sense to discuss inertia, since the eigenvalues of $\mathcal{A}$ may be complex. Moreover, these eigenvalues are not preserved by $\mathcal{M}$, since $\mathcal{Q}^T \neq \mathcal{Q}^{-1}$. However, solving systems with this matrix are straightforward and the process is equivalent to a nullspace method, such as that employed by Mahawar and Sarin [15].

3.3. Updating the antitriangular factorisation. Mastronardi and Van Dooren showed that the antitriangular factorisation can be efficiently updated when a rank-1 modification is applied. These updates can be somewhat involved when applied to a general symmetric matrix but the procedure simplifies for saddle point matrices. We discuss some relevant modifications here.

If A is ill-conditioned or singular it may be desirable to apply the augmented Lagrangian approach in which we replace (4.1) by [2, Section 3.5]

$$\mathcal{A}_{AL}x = \begin{bmatrix} A + B^TEB & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f + B^TEg \\ g \end{bmatrix}, \tag{3.1}$$

where $E \in \mathbb{R}^{m\times m}$ is symmetric positive definite. Updating the antitriangular factorisation in this case is straightforward, since B^TEB is orthogonal to null(B). Thus, given the antitriangular factorisation (2.7) of $\mathcal{A} = \mathcal{Q}\mathcal{M}\mathcal{Q}^T$, the antitriangular factorisation of $\mathcal{A}_{AL}$ is $\mathcal{A}_{AL} = \mathcal{Q}\mathcal{M}_{AL}\mathcal{Q}^T$, where

$$\mathcal{M}_{AL} = \begin{bmatrix} 0 & 0 & Y^T \\ 0 & X & Z^T \\ Y & Z & W + YEY^T \end{bmatrix}.$$

The idea can be extended to more general symmetric positive semidefinite updates $F \in \mathbb{R}^{n\times n}$ to A. If

$$\mathcal{A}_F = \begin{bmatrix} A + F & B^T \\ B & 0 \end{bmatrix}$$

then the antitriangular factorisation of $\mathcal{A}_F$ is $\mathcal{A}_F = \mathcal{Q}\mathcal{M}_F\mathcal{Q}^T$, where

$$\mathcal{M}_F = \begin{bmatrix} 0 & 0 & Y^T \\ 0 & X_F & Z_F^T \\ Y & Z_F & W_F \end{bmatrix},$$

with $X_F = X+(U^{(2)})^T F U^{(2)}$, $Z_F = Z+S_m(U^{(1)})^T F U^{(2)}$ and $W_F = W+S_m(U^{(1)})^T F U^{(1)} S_m$. If F is low-rank then the updates to W, X and Z can be cheaply computed.

When a sequence of saddle point matrices are solved, as in the quasi-Newton method, or in interior point methods, it may be necessary to update B and B^T as well as A. If the updates have special structure the antitriangular factorisation can be updated by low-rank approximations as in Griewank, Walther and Korzec [8]. In the generic case, however, we require a low-rank update of the antitriangular factorisations of B and B^T, which can be obtained by extending the rank-one update procedure described in Mastronardi and Van Dooren [17] or by using an updated QR-factorisation. Since both approaches are similar, we describe the QR approach here.

We consider the updated matrix

$$\mathcal{A}_{UP} = \mathcal{A} + uv^T + vu^T = \begin{bmatrix} A & (B+u_1v_1^T)^T \\ (B+u_1v_1^T) & 0 \end{bmatrix},$$

where $u = [0 \quad u_1]^T$, $v = [v_1 \quad 0]^T$, $u_1 \in \mathbb{R}^{m\times 1}$ and $v_1 \in \mathbb{R}^{n\times 1}$. If $B^T = U\widehat{R}$, $\widehat{R} = [R^T \quad 0]^T$, is the QR decomposition of B^T then a QR decomposition of $B^T + v_1u_1^T$ is [6, Section 12.5.1]

$$B^T + v_1u_1^T = U_{UP}\widehat{R}_{UP} = \begin{bmatrix} U_{UP}^{(1)} & U_{UP}^{(2)} \end{bmatrix} \begin{bmatrix} R_{UP} \\ 0 \end{bmatrix},$$

where $U_{UP} = UJ \in \mathbb{R}^{n\times n}$, $U_{UP}^{(1)} \in \mathbb{R}^{n\times m}$ and J is orthogonal. Since

$$Q_5\mathcal{Q}^T(\mathcal{A} + uv^T + vu^T)\mathcal{Q}Q_5^T = \begin{bmatrix} 0 & (\widehat{R} + U^Tv_1u_1^T)^T \\ \widehat{R} + U^Tv_1u_1^T & U^TAU \end{bmatrix}$$

then with

$$Q_6 = \begin{bmatrix} I_m & 0 \\ 0 & J \end{bmatrix}$$

we have that

$$Q_6^TQ_5\mathcal{Q}^T(\mathcal{A} + uv^T + vu^T)\mathcal{Q}Q_5^TQ_6 = \begin{bmatrix} 0 & R_{UP}^T & 0 \\ R_{UP} & (\widehat{A}_{UP})_{11} & (\widehat{A}_{UP})_{12} \\ 0 & (\widehat{A}_{UP})_{21} & (\widehat{A}_{UP})_{22} \end{bmatrix},$$

where $(\widehat{A}_{UP})_{ij} = (U_{UP}^{(i)})^T A (U_{UP}^{(j)})$, $j = 1, 2$.

Finally, applying Q_5 as in Algorithm 2 gives the antitriangular form:

$$\mathcal{Q}_{UP}^T\mathcal{Q}^T(\mathcal{A} + uv^T + vu^T)\mathcal{Q}\mathcal{Q}_{UP} = \begin{bmatrix} 0 & 0 & Y_{UP}^T \\ 0 & X_{UP} & Z_{UP}^T \\ Y_{UP} & Z_{UP} & W_{UP} \end{bmatrix},$$

where $\mathcal{Q}_{UP} = Q_5^TQ_6Q_5$, $W_{UP} = S_m(\widehat{A}_{UP})_{11}S_m$, $X_{UP} = (\widehat{A}_{UP})_{22}$ and $Z_{UP} = S_m(\widehat{A}_{UP})_{12}$.

The computational cost associated with the update arises from the application of J, a composition of Givens rotations, to $\mathcal{M}$ and $u_1v_1^T$.

4. Comparison with the nullspace method. The range space method for solving

$$\mathcal{A}x = \begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix} \tag{4.1}$$

is applicable when A is invertible and is related to a block LDL^T decomposition [2, Section 5] since

$$\mathcal{A} = \begin{bmatrix} I_n & 0 \\ BA^{-1} & I_m \end{bmatrix} \begin{bmatrix} A & 0 \\ 0 & -BA^{-1}B^T \end{bmatrix} \begin{bmatrix} I_n & A^{-1}B^T \\ 0 & I_m \end{bmatrix}.$$

The matrix factorisation representation of the nullspace method is the antitriangular factorisation, as we now show.

Given a basis for the nullspace[1] of B, such as $U^{(2)}$, and a particular solution $\widehat{u}$ of $Bu = g$ the nullspace method proceeds as follows [2, Section 6][21, Section 15.2]:

1. Solve $(U^{(2)})^T A U^{(2)} v = (U^{(2)})^T (f - A\widehat{u})$;
2. Set $u_* = U^{(2)} v + \widehat{u}$;
3. Solve $BB^T p_* = B(f - Au_*)$,

then (u_*, p_*) solves (4.1).

On the other hand, applying the antitriangularisation (2.7) to (4.1) gives

$$(\mathcal{Q}^T \mathcal{A} \mathcal{Q}) y = \mathcal{M} y = (\mathcal{Q}^T b), \quad y = \mathcal{Q}^T x$$

or

$$\begin{bmatrix} 0 & 0 & Y^T \\ 0 & X & Z^T \\ Y & Z & W \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} g \\ (U^{(2)})^T f \\ S(U^{(1)})^T f \end{bmatrix}.$$

To recover u and p we must solve

$$Y^T y_3 = g \tag{4.2a}$$

$$X y_2 + Z^T y_3 = (U^{(2)})^T f \tag{4.2b}$$

$$Y y_1 + Z y_2 + W y_3 = S(U^{(1)})^T f, \tag{4.2c}$$

using the antitriangular substitution described in Section 2. This is equivalent to applying the inverse of $\mathcal{M}$, which has upper block antitriangular structure, that is,

$$\mathcal{M}^{-1} = \begin{array}{c} \begin{bmatrix} Y^{-1}(ZX^{-1}Z^T - W)Y^{-T} & -Y^{-1}ZX^{-1} & Y^{-1} \\ -X^{-1}Z^T Y^{-T} & X^{-1} & 0 \\ Y^{-T} & 0 & 0 \end{bmatrix} \end{array} \begin{array}{l} \}\, m \\ \}\, n-m \\ \}\, m \end{array}. \tag{4.3}$$

Having obtained y, we recover u and p from

$$\begin{bmatrix} u \\ p \end{bmatrix} = \mathcal{Q} y = \begin{bmatrix} U^{(2)} y_2 + U^{(1)} S_m y_3 \\ y_1 \end{bmatrix}. \tag{4.4}$$

We now show that solving (4.2a)–(4.2c) is equivalent to applying the nullspace method. From (4.2a), since $Y = S_m R$ and $B^T = U^{(1)} R$,

$$R^T (U^{(1)})^T U^{(1)} S_m y_3 = B(U^{(1)} S_m y_3) = g,$$

[1] Note that any basis of the nullspace of B in general would be sufficient but we use $U^{(2)}$ here as this common choice corresponds to the antitriangular factorisation.

so that $\widehat{u} = U^{(1)} S_m y_3$ is a particular solution of $Bu = g$.

Since $X = \widehat{A}_{22}$ and $Z = S\widehat{A}_{12}$, where $\widehat{A}_{ij} = (U^{(i)})^T A U^{(j)}$, $i, j = 1, 2$ as before, we have from (4.2b) that

$$(U^{(2)})^T A U^{(2)} y_2 = (U^{(2)})^T (f - A\widehat{u}).$$

Substituting for $\widehat{u}$ and $W = S\widehat{A}_{11}S$ in (4.2c) then gives that

$$\begin{aligned} Rp &= (U^{(1)})^T [f - A(U^{(2)} y_2 + \widehat{u})] \\ R^T R p &= (U^{(1)} R)^T [f - A(U^{(2)} y_2 + \widehat{u})] \\ BB^T p &= B(f - A u_*), \end{aligned}$$

where $u_* = U^{(2)} y_2 + \widehat{u}$.

Thus, solving a system with the antitriangular factorisation is equivalent to applying the nullspace method with the QR nullspace basis and with $\widehat{u} = U^{(1)} S_m y_3$. From (4.4) we then have that $u = u^* = U^{(2)} y_2 + \widehat{u}$ and $p = y_1$.

Note that no antitriangular solves are required in the nullspace method, even though we are solving a linear system with a block antitriangular matrix. This is because the permutation matrix S_m that transforms the upper triangular matrix R to antitriangular form occurs as $S_m^2 = I$ in (4.2a) and can be eliminated from (4.2c).

We have seen that the antitriangular factorisation allows us to view the nullspace method as a factorisation rather than as a procedure, similarly to other direct solvers such as Gaussian elimination, which can be written as the product of structured matrices. This idea could of course be generalised to other factorisations with different representations of the nullspace.

5. Eigenvalue bounds. Of interest when solving saddle point systems are the eigenvalues of A and the (negative) Schur complement $BA^{-1}B^T$ when it exists, i.e., when A is invertible. Since the Schur complement involves the inverse of the $n \times n$ matrix A, its eigenvalues can be particularly difficult to approximate. Here we give bounds for the eigenvalues of both matrices that depend only on the eigenvalues of X and W and the singular values of Y.

Since

$$\widehat{M} = \begin{bmatrix} X & Z^T \\ Z & W \end{bmatrix} = \begin{bmatrix} (U^{(2)})^T \\ S_m (U^{(1)})^T \end{bmatrix} A \begin{bmatrix} U^{(2)} U^{(1)} S_m \end{bmatrix}$$

for Algorithm 2[2] the eigenvalues of A are identical to those of $\widehat{M}$. This means that Cauchy's interlacing theorem can be used to bound the eigenvalues of A.

LEMMA 1. *Let $\lambda_1(A) \le \lambda_2(A) \le \cdots \le \lambda_n(A)$ be the eigenvalues of A, $0 < \lambda_1(X) \le \lambda_2(X) \le \cdots \le \lambda_{n-m}(X)$ be the eigenvalues of X and $\lambda_1(W) \le \lambda_2(W) \le \cdots \le \lambda_m(W)$ be the eigenvalues of W. Then,*

$$\begin{aligned} \lambda_k(A) \le \lambda_k(X) \le \lambda_{k+m}(A),\ k = 1, \dots, n-m, \\ \lambda_k(A) \le \lambda_k(W) \le \lambda_{k+n-m}(A),\ k = 1, \dots, m. \end{aligned}$$

Proof. The results follow from the similarity of $\widehat{M}$ and A and by applying the interlacing theorem [10, Theorem 4.3.15] to $\widehat{M}$ using X or W. □

[2]An analogous transform holds for Algorithm 1.

Also of interest when A is positive definite are the eigenvalues of $BA^{-1}B^T$. To bound these we first prove the following lemma.

LEMMA 2. *Assume that A is positive definite. Let $0 < \lambda_1(\widetilde{W}) \leq \lambda_2(\widetilde{W}) \leq \cdots \leq \lambda_m(\widetilde{W})$ be the eigenvalues of $\widetilde{W} = S_m(U^{(1)})^T A^{-1} U^{(1)} S_m$ and $0 < \lambda_1(W^{-1}) \leq \lambda_2(W^{-1}) \leq \cdots \leq \lambda_m(W^{-1})$ be the eigenvalues of W^{-1}. Then,*

$$\lambda_k(W^{-1}) \leq \lambda_k(\widetilde{W}), \qquad k = 1, \ldots, m.$$

Proof. First note that, for any $x \neq 0$, the Cauchy-Schwarz inequality gives that $(x^T x)^2 = (x^T A^{\frac{1}{2}} A^{-\frac{1}{2}} x)^2 \leq (x^T A x)(x^T A^{-1} x)$ or

$$\frac{x^T x}{x^T A x} \leq \frac{x^T A^{-1} x}{x^T x}.$$

Using this and the orthogonality of S_m and $U^{(1)}$, the Courant-Fischer theorem [10, page 180] gives

$$\begin{aligned}
\lambda_k(W^{-1}) &= \min_{\dim(\mathcal{S})=k} \max_{\substack{x\in\mathcal{S} \\ x\neq 0}} \frac{x^T S_m((U^{(1)})^T A U^{(1)})^{-1} S_m x}{x^T S_m^T S_m x} \\
&= \min_{\dim(\mathcal{S})=k} \max_{\substack{y\in\mathcal{S} \\ y\neq 0}} \frac{y^T (U^{(1)})^T U^{(1)} y}{y^T (U^{(1)})^T A U^{(1)} y} \\
&\leq \min_{\dim(\mathcal{S})=k} \max_{\substack{y\in\mathcal{S} \\ y\neq 0}} \frac{y^T (U^{(1)})^T A^{-1} U^{(1)} y}{y^T (U^{(1)})^T U^{(1)} y} \\
&= \min_{\dim(\mathcal{S})=k} \max_{\substack{z\in\mathcal{S} \\ z\neq 0}} \frac{z^T S_m (U^{(1)})^T A^{-1} U^{(1)} S_m z}{z^T z} = \lambda_k(\widetilde{W}).
\end{aligned}$$

□

Lemma (2) can be used to bound the eigenvalues of $BA^{-1}B^T$ as follows.

COROLLARY 3. *Let $0 < \lambda_1(W^{-1}) \leq \lambda_2(W^{-1}) \leq \cdots \leq \lambda_m(W^{-1})$ be the eigenvalues of W^{-1} and $0 < \lambda_1(BA^{-1}B^T) \leq \lambda_2(BA^{-1}B^T) \leq \cdots \leq \lambda_m(BA^{-1}B^T)$ be the eigenvalues of $BA^{-1}B^T$. Then,*

$$\theta_k \lambda_k(W^{-1}) \leq \lambda_k(BA^{-1}B^T), \qquad k = 1, \ldots, m,$$

where $\sigma_m(Y)^2 \leq \theta_k \leq \sigma_1(Y)^2$.

Proof. From the QR decomposition (2.4) of B^T we have that $BA^{-1}B^T = Y^T \widetilde{W} Y$ and so $\lambda_k(BA^{-1}B^T) = \lambda_k(Y^T \widetilde{W} Y)$. By Ostrowski's theorem [10, Theorem 4.5.9], it follows that $\lambda_k(Y^T \widetilde{W} Y) = \theta_k \lambda_k(\widetilde{W})$. Combining this with the inequality in Lemma 2 gives the result. □

Thus, the antitriangular factorisation gives lower bounds on all the eigenvalues of the Schur complement $BA^{-1}B^T$. In particular, it bounds from below the smallest eigenvalue, which can be useful when bounding the eigenvalues of $\mathcal{A}$ or when approximating inf-sup constants [23]. Note that since Y is antitriangular its singular values are relatively easy to compute.

6. The antitriangular factorisation and preconditioning. When the saddle point system (4.1) is too large to be solved by a direct method an iterative method such as a Krylov subspace method is usually applied. Unfortunately, however, these iterative methods typically converge slowly when applied to saddle point problems unless preconditioners are used. Many preconditioners for saddle point matrices have been proposed [2, Section 10][3], but we focus here on block preconditioners and show how they can be factored by the antitriangular factorisation in Section 2. We first discuss block diagonal and block triangular preconditioners and then describe constraint preconditioners, showing that in this latter case the same orthogonal transformation converts $\mathcal{A}$ and $\mathcal{P}$ to antitriangular form. We assume throughout that $\mathcal{A}$ in (4.1) is factorised in antitriangular form (2.7), i.e., that $\mathcal{A} = \mathcal{Q}\mathcal{M}\mathcal{Q}^T$.

We briefly mention the block diagonal matrix

$$\mathcal{P}_D = \begin{bmatrix} T & 0 \\ 0 & V \end{bmatrix},$$

where $T \in \mathbb{R}^{n\times n}$ and $V \in \mathbb{R}^{m\times m}$ are symmetric. Often T is chosen to approximate A and V to approximate the Schur complement $BA^{-1}B^T$. Indeed, if $T = A$ and $V = BA^{-1}B^T$ then $\mathcal{P}_D^{-1}\mathcal{A}$ has three eigenvalues, 1 and $(1 \pm \sqrt{5})/2$ [13, 18].

Applying Q in a similarity transform gives

$$\mathcal{Q}^T\mathcal{P}_D\mathcal{Q} = \begin{bmatrix} V & 0 & 0 \\ 0 & \widehat{T}_{22} & \widehat{T}_{12}^T S \\ 0 & S\widehat{T}_{12} & S\widehat{T}_{11}S \end{bmatrix},$$

where $\widehat{T}_{ij} = (U^{(i)})^T T U^{(j)}$, $i, j = 1, 2$. Thus, the transformed preconditioner is also block diagonal, with an $m \times m$ block followed by an $n \times n$ block. Note that since $\mathcal{P}_D$ is positive definite, $\mathcal{Q}^T\mathcal{P}_D\mathcal{Q}$ can not have significant block antidiagonal structure.

Similarly, the block lower triangular preconditioner

$$\mathcal{P}_T = \begin{bmatrix} T & 0 \\ B & V \end{bmatrix}$$

gives

$$\mathcal{Q}^T\mathcal{P}_T\mathcal{Q} = \begin{bmatrix} V & 0 & Y^T \\ 0 & \widehat{T}_{22} & \widehat{T}_{12}^T S \\ 0 & S\widehat{T}_{12} & S\widehat{T}_{11}S \end{bmatrix}.$$

The corresponding upper triangular preconditioner has an analogous form.

Constraint preconditioners [12, 14, 19, 20]

$$\mathcal{P}_C = \begin{bmatrix} T & B^T \\ B & 0 \end{bmatrix}, \tag{6.1}$$

on the other hand, preserve the constraints of $\mathcal{A}$ exactly but replace A by a symmetric approximation T. Precisely because the constraints are preserved,

$$\mathcal{M}_C = \mathcal{Q}^T\mathcal{P}_C\mathcal{Q} = \begin{bmatrix} 0 & 0 & Y^T \\ 0 & \widehat{T}_{22} & \widehat{T}_{12}S \\ Y & S\widehat{T}_{12} & S\widehat{T}_{11}S \end{bmatrix}, \tag{6.2}$$

where $\widehat{T}_{ij} = U_i^T T U_j$, $i,j = 1,2$, is an antitriangular matrix when $\widehat{T}_{22}$ is positive definite.

It is known that $\mathcal{P}_C^{-1}\mathcal{A}$ has at least $2m$ unit eigenvalues, with the remainder being the eigenvalues λ of $(U^{(2)})^T A U^{(2)} v = \lambda (U^{(2)})^T T U^{(2)} v$ [12, 14]. (Note that we could use any basis for the nullspace of B in place of $U^{(2)}$.) Since A is positive definite on the nullspace of B, any non-unit eigenvalues are real, although negative eigenvalues will occur when $(U^{(2)})^T T U^{(2)}$ is not positive definite. These facts are also easily discerned from the antitriangular forms. Since $\mathcal{P}_C^{-1}\mathcal{A} = \mathcal{Q}\mathcal{M}_C^{-1}\mathcal{M}\mathcal{Q}^T$, which is similar to $\mathcal{M}_C^{-1}\mathcal{M}$, explicitly obtaining $\mathcal{M}_C^{-1}$ as in (4.3) and multiplying by $\mathcal{M}$ gives a block upper triangular matrix from which the eigenvalues are immediately obvious. Indeed, Keller *et al.* [12] use a flipped version of (6.2) to investigate the eigenvalues of $\mathcal{P}_C^{-1}\mathcal{A}$.

The matrix $\mathcal{Q}^T \mathcal{P}_C \mathcal{Q}$ may be applied using the procedure outlined in Section 4, and it makes clear the equivalence between constraint preconditioners and the nullspace method that has previously been observed [7, 14, 22]. Conversely, any matrix $\mathcal{N}$ in antitriangular form with $Y = SR$ defines a constraint preconditioner

$$\mathcal{P}_C = \mathcal{Q}\mathcal{N}Q^T = \begin{bmatrix} T & B \\ B & 0 \end{bmatrix}$$

for $\mathcal{A}$, with

$$T = U^{(2)} X (U^{(2)})^T + U^{(1)} S Z (U^{(2)})^T + U^{(2)} Z^T S (U^{(1)})^T + U^{(1)} S W S (U^{(1)})^T.$$

We note that alternative factorisations of constraint preconditioners, some of which rely on a basis for the nullspace of B, have also been proposed. In particular, the Schilders' factorisation [4, 5, 24] re-orders the variables so that $B = [B_1\ B_2]$, with $B_1 \in \mathbb{R}^{m\times m}$ nonsingular. In this case

$$N = \begin{bmatrix} -B_1^{-1}B_2 \\ I \end{bmatrix}$$

is a basis for the nullspace, and is important to the factorisation.

For a true, inertia-revealing antitriangular factorisation T must be positive definite on the nullspace of B, so that $\widehat{T}_{22}$ is symmetric positive definite. If we are only interested in preconditioning (and not in the inertia of $\mathcal{P}_C$) we only require that $\widehat{T}_{22}$ is invertible.

In summary, we see that applying the same orthogonal similarity transform that makes $\mathcal{A}$ antitriangular to $\mathcal{P}_D$, $\mathcal{P}_T$ and $\mathcal{P}_C$ results in preconditioners with specific structures. The antitriangular form of $\mathcal{P}_C$ makes the equivalence between constraint preconditioners and the nullspace method clear, reveals the eigenvalues of $\mathcal{P}_C^{-1}\mathcal{A}$ and may provide other insights into the properties of constraint preconditioners.

7. Conclusions. We have considerably simplified the antitriangular factorisation for symmetric indefinite matrices of Mastronardi and Van Dooren in the specific and common case of saddle point matrices. This leads to the observation that this factorisation is equivalent to the well known nullspace method. We have shown that the factorisation is strongly stable and that low-rank updates to A and B can be efficiently incorporated into an existing antitriangular factorisation. The blocks X, Z and Y can be used to obtain bounds on the eigenvalues of A and the Schur complement. Additionally, we have considered the form of this antitriangular factorisation for popular constraint preconditioning, block diagonal and block triangular preconditioning, showing how specific structures are preserved.

Acknowledgements The authors would like to thank Gil Strang for inspiring this work and for valuable suggestions. We are also grateful to David Titley-Peloquin and the editor and referees for their helpful comments and advice that greatly improved the manuscript. This publication was based on work supported in part by Award No KUK-C1-013-04, made by King Abdullah University of Science and Technology (KAUST).

REFERENCES

[1] S. Bellavia, V. De Simone, D. di Serafino, and B. Morini, *Updating constraint preconditioners for KKT systems in quadratic programming via low-rank corrections*, ArXiv e-prints, (2013).

[2] M. Benzi, G. H. Golub, and J. Liesen, *Numerical solution of saddle point problems*, Acta Numer., 14 (2005), pp. 1–137.

[3] Michele Benzi and Andrew J. Wathen, *Some preconditioning techniques for saddle point problems*, in Model Order Reduction: Theory, Research Aspects and Applications, W. H. A. Schilders, H. A. van der Vorst, and J. Rommes, eds., vol. 13 of Mathematics in Industry, Springer-Verlag Berlin Heidelberg, 2008, pp. 195–211.

[4] H. S. Dollar, N. I. M. Gould, W. H. A. Schilders, and A. J. Wathen, *Implicit-factorization preconditioning and iterative solvers for regularized saddle-point systems*, SIAM J. Matrix Anal. Appl., 28 (2006), pp. 170–189.

[5] H. S. Dollar and A. J. Wathen, *Approximate factorization constraint preconditioners for saddle-point problems*, SIAM J. Sci. Comput., 27 (2006), pp. 1555–1572.

[6] G. H. Golub and C. F. van Loan, *Matrix Computations*, The John Hopkins University Press, Maryland, USA, third ed., 1996.

[7] N. I. M. Gould, M. E. Hribar, and J. Nocedal, *On the solution of equality constrained quadratic programming problems arising in optimization*, SIAM J. Sci. Comput., 23 (2001), pp. 1376–1395.

[8] A. Griewank, A. Walther, and M. Korzec, *Maintaining factorized KKT systems subject to rank-one updates of Hessians and Jacobians*, Optimization Methods and Software, 22 (2007), pp. 279–295.

[9] N. J. Higham, *Accuracy and Sttability of Numerical Agorithms*, SIAM, Philadelphia, PA, second ed., 2002.

[10] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, Cambridge, 1990.

[11] V. E. Howle and S. A. Vavasis, *An iterative method for solving complex-symmetric systems arising in electrical power modeling*, SIAM J. Matrix Anal. Appl., 26 (2005), pp. 1150–1178.

[12] C. Keller, N. I. M. Gould, and A. J. Wathen, *Constraint preconditioning for indefinite linear systems*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1300–1317.

[13] YU. A. Kuznetsov, *Efficient iterative solvers for elliptic finite element problems on nonmatching grids*, Russ. J. Numer. Anal. Math. Modelling, 10 (1995), pp. 187–211.

[14] L. Lukšan and J. Vlček, *Indefinitely preconditioned inexact Newton method for large sparse equality constrained non-linear programming problems*, Numer. Linear Algebr. Appl., 5 (1998), pp. 219–247.

[15] H. Mahawar and V. Sarin, *Parallel iterative methods for dense lienar systems in inductance extraction*, Parallel Computing, 29 (2003), pp. 1219–1235.

[16] N. Mastronardi and P. Van Dooren, *An algorithm for solving the indefinite least squares problem with equality constraints*, BIT, To appear (2013).

[17] ———, *The antitriangular factorization of symmetric matrices*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 173–196.

[18] M. F. Murphy, G. H. Golub, and A. J. Wathen, *A note on preconditioning for indefinite linear systems*, SIAM J. Sci. Comput., 21 (2000), pp. 1969–1972.

[19] B. A. Murtagh and M. A. Saunders, *Large-scale linearly constrained optimization*, Math. Program., 14 (1978), pp. 41–72.

[20] ———, *A projected Lagrangian algorithm and its implementation for sparse nonlinear constraints*, Math. Program. Stud., 16 (1982), pp. 84–117.

[21] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, New York, NY, 1999.

[22] I Perugia, V Simoncini, and M Arioli, *Linear algebra methods in a mixed approximation of magnetostatic problems*, SIAM J. Sci. Comput., 21 (1999), pp. 1085–1101.

[23] J. Pestana and A. J. Wathen, *Natural preconditioners for saddle point problems*, Tech. Report 1754, Mathematical Institute, University of Oxford, 2013.
[24] W. H. A. Schilders, *A preconditioning technique for indefinite systems arising in electronic circuit simulation*. Talk at the one-day meeting on preconditioning methods for indefinite linear systems, December 9, 2002.
[25] J.-G. Sun, *Structured backward errors for KKT systems*, Linear Algebra Appl., 288 (1999), pp. 75–88.