# A NULL-SPACE APPROACH FOR LARGE-SCALE SYMMETRIC SADDLE POINT SYSTEMS WITH A SMALL AND NON ZERO (2,2) BLOCK

JENNIFER SCOTT[*] AND MIROSLAV TŮMA[†]

**Abstract.** Null-space methods have long been used to solve large sparse $n \times n$ symmetric saddle point systems of equations in which the $(2,2)$ block is zero. This paper focuses on the case where the $(1,1)$ block is ill conditioned or rank deficient and the $k \times k$ $(2,2)$ block is non zero and small ($k \ll n$). Additionally, the $(2,1)$ block may be rank deficient. Such systems arise in a range of practical applications. A novel null-space approach is proposed that transforms the system matrix into a nicer symmetric saddle point matrix of order $n$ that has a non zero $(2,2)$ block of order at most $2k$ and, importantly, the $(1,1)$ block is symmetric positive definite. Success of any null-space approach depends on constructing a suitable null-space basis. We propose methods for wide matrices having far fewer rows than columns with the aim of balancing stability of the transformed saddle point matrix with preserving sparsity in the $(1,1)$ block. Linear least squares problems that contain a small number of dense rows are an important motivation and are used to illustrate our ideas and to explore their potential for solving large-scale systems.

**Key words.** sparse matrices, dense rows, null-space method, linear least squares problems, saddle point systems.

## 1. Introduction.

Our interest is in solving symmetric saddle point systems of equations of the form

$$\mathcal{A}\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} H & B^T \\ B & -C \end{pmatrix}\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}, \tag{1.1}$$

in which $H \in \mathbb{R}^{n\times n}$ is large, sparse and symmetric positive semidefinite (SPSD), $B \in \mathbb{R}^{k\times n}$ $(n > k)$ and $C \in \mathbb{R}^{k\times k}$ is SPSD. Our focus is on $k \ll n$ (the saddle point matrix may then be referred to as a "bordered" matrix) and the "wide" matrix $B$ may contain one or more dense rows. Such systems arise in a range of scientific applications, including the finite element approximation of PDEs in which a constraint is enforced on a subset of unknowns via a global scalar Lagrange multiplier, numerical continuation methods for large nonlinear systems of equations, electronic circuit simulation, constrained optimization, and linear least squares problems (see the excellent paper of Benzi, Golub and Liesen [7], which describes a wide range of real life problems that are dependent on the solution of saddle-point systems and see also [28, 35]). We are particularly concerned with large-scale least squares problems that have a few dense rows. They can be expressed in the form (1.1) and provide an important motivation for the null-space approach proposed in this paper. In this application, the $(1,1)$ block can be ill conditioned or rank deficient.

There has been considerable work over many years and in different research areas into null-space methods for saddle point problems with a zero $(2,2)$ block. The basic idea is to characterize the null space of the constraint (off-diagonal) blocks and use that characterization to reduce the system to two smaller linear systems of order $(n-k)\times(n-k)$ and $k \times k$ that have nice properties and are thus straightforward to solve. Consequently, applications of null-space methods are usually geared towards problems for which $n - k$ is small. An attractive feature of null-space methods that widens their applicability is that the $(1,1)$ block $H$ need not be invertible. However, a key problem is the need to construct a null-space basis $\mathcal{N}(B)$ for the matrix $B$. In general, this is challenging, particularly if it is desirable that the matrix whose columns form a basis for $\mathcal{N}(B)$ has additional properties, such as bandedness, sparsity or orthogonality. Moreover, the classical null-space method is restricted to $C = 0$.

The objectives of this paper are twofold. Firstly, we present a null-space based approach for symmetric saddle point systems in which the $(1,1)$ block is SPSD and the small $k \times k$ block $C$ is non zero. The new approach preserves symmetry and results in a transformed symmetric saddle point system of the same order with favourable numerical properties, namely a sparse symmetric positive definite (SPD) $(1,1)$ block

[*] STFC Rutherford Appleton Laboratory, Harwell Campus, Didcot, Oxfordshire, OX11 0QX, UK and School of Mathematical, Physical and Computational Sciences, University of Reading, Reading RG6 6AQ, UK. Correspondence to: jennifer.scott@stfc.ac.uk. Partially supported by EPSRC grant EP/M025179/1.

[†] Department of Numerical Mathematics, Faculty of Mathematics and Physics, Charles University, Czech Republic, (mirektuma@karlin.mff.cuni.cz.) Supported by the project 18-12719S of the Grant Agency of the Czech Republic.

and a non zero $(2,2)$ block of size $k+r$, where $r \leq k$ is the rank of $B$. Having an SPD $(1,1)$ block allows the use of block (possibly incomplete) factorization methods to be used to solve the system. Secondly, we propose techniques for constructing a null-space basis $\mathcal{N}(B)$ when $k$ is small, with an emphasis on balancing sparsity in the transformed problem with stability through the use of QR factorizations with threshold pivoting.

This paper is organised as follows. In Section 2, we first recall the standard null-space approach for saddle point systems with a zero $(2,2)$ block and the recent method of Howell [35] that allows for a non zero $C$ but sacrifices symmetry. We then propose a new symmetry preserving null-space approach for the case of a non zero $C$ and discuss how it can be used to solve large-scale linear least squares problems that have a small number of dense rows. Null basis construction is considered in Section 3. A brief review of methods for building null-space bases for sparse matrices is given before we focus on the case of interest to us: null-space bases for wide matrices that may include some dense rows. A number of approaches are proposed, with an emphasis on ensuring sparsity within the transformed saddle point system while also guaranteeing numerical stability through the use of pivoting. In Section 4, numerical examples from practical applications demonstrate the effectiveness of the approaches and illustrate their potential strengths and limitations. Finally, some concluding remarks and pointers to future research directions area given in Section 5.

We end this section by introducing notation that we will use throughout this paper. Let $x \in \mathbb{R}^n$ be a vector. We denote the $i$-th entry of $x$ by $(x)_i$ and $(x)_{j:l}$ denotes entries $j$ to $l$ of $x$ $(1 \leq j < l \leq n)$. $x^\perp$ denotes the $(n-1)$-dimensional space of all vectors $w \in \mathbb{R}^n$ such that $x^T w = 0$. $e_i$ denotes the $i$-th unit vector. Let $B \in \mathbb{R}^{k\times n}$ be a matrix. The $(i,j)$-th entry of $B$ is given by $(B)_{i,j}$. $(B)_{:,l}$ denotes column $l$ of $B$ and $(B)_{:,1:l}$ denotes a matrix comprising columns 1 to $l$ of $B$. $(B)_{1:l,1:l}$ denotes the leading submatrix of $B$ of order $l$. The rows of $B$ are $b_1^T, \dots, b_k^T$. $\mathcal{N}(B)$ and $\mathcal{R}(B)$ denote the null space and range space of $B$, respectively. $Z$ is used to denote a null-space basis matrix (that is, its columns form a null-space basis). $P$ (with or without a subscript and/or superscript) is used for a permutation matrix. $I_k$ denotes the $k \times k$ identity matrix and $0_{n,k}$ denotes the $n \times k$ null matrix.

## 2. Null-space approach for solving saddle point systems.

### 2.1. Null-space approach for zero $C$.

Consider the case that $B$ is of full rank and $C = 0_{k,k}$, that is,

$$\begin{pmatrix} H & B^T \\ B & 0_{k,k} \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}. \tag{2.1}$$

Such systems frequently arise in practical applications, particularly when solving constrained optimization problems (where they are usually referred to as reduced Hessian methods); a null-space approach is one possible method of solution (see, for example, [7, Section 6]). Assume we have a matrix $Z \in \mathbb{R}^{n\times(n-k)}$ whose columns form a basis for $\mathcal{N}(B)$ i.e., $BZ = 0_{k,n}$, and that we have a particular solution for the second equation in (2.1), that is, a vector $\hat{u}$ such that

$$B\hat{u} = g.$$

Then solving (2.1) is equivalent to solving

$$\begin{pmatrix} H & B^T \\ B & 0_{k,k} \end{pmatrix} \begin{pmatrix} \bar{u} \\ v \end{pmatrix} = \begin{pmatrix} f - H\hat{u} \\ 0_{k,1} \end{pmatrix},$$

where $\bar{u} = u - \hat{u}$. The second equation in this system is equivalent to finding a vector $z \in \mathbb{R}^{n-k}$ such that $\bar{u} = Zz$. Substituting this into the first equation gives

$$\begin{aligned} HZz + B^T v &= f - H\hat{u} \\ \iff \quad Z^T HZz &= Z^T(f - H\hat{u}) \end{aligned} \tag{2.2}$$

Therefore, by solving the reduced system (2.2), it is possible to straightforwardly recover $u = \hat{u} + Zz$. Finally, $v$ can be obtained by solving the reduced $k \times k$ symmetric positive definite system

$$BB^Tv = B(f - Hu).$$

The null-space method for solving (2.1) is summarized as Algorithm 1.

**Algorithm 1** Null-space method for solving (2.1) with $B$ of full rank

```
1: Construct $Z \in \mathbb{R}^{n\times(n-k)}$ such that its columns form a basis for the null-space of $B \in \mathbb{R}^{k\times n}$
2: Find $\hat{u} \in \mathbb{R}^n$ such that $B\hat{u} = g$.
3: Solve $Z^THZz = Z^T(f - H\hat{u})$.
4: Set $u = \hat{u} + Zz$.
5: Solve $BB^Tv = B(f - Hu)$ for $v \in \mathbb{R}^k$
```

A recent study of null-space factorizations for saddle point systems of the form (2.1) has been given by Rees and Scott [52] and the advantages and disadvantages of null-space methods are summarized in [43]. They can be very efficient for solving a series of problems with the same block $B$ but different $H$ because the null-space basis $\mathcal{N}(B)$ needs to be computed only once. A further key advantage is that $H^{-1}$ is not required. In fact, the method is applicable if $H$ is singular, provided $\mathcal{R}(H) \cap \mathcal{N}(B) = \{0\}$. Furthermore, the Schur complement is not needed. The null-space approach can also be useful when additional rows (dense or sparse) are added to the system matrix a posteriori [23]. However, a potential difficulty is the need to construct a null-space basis matrix $Z$ and, even when $Z$ can be computed efficiently, some columns of $Z$ may contain a large number of entries so that $Z^THZ$ is not sparse.

The null-space method can be generalised to unsymmetric saddle point systems with a zero $(2, 2)$ block in which $H$ is unsymmetric and the full rank $(1, 2)$ and $(2, 1)$ blocks are $B_1^T$ and $B_2$ with $B_1 \neq B_2$. In this case, $\hat{u}$ is required such that $B_2\hat{u} = g$ and matrices $Z_1$ and $Z_2 \in \mathbb{R}^{n\times(n-k)}$ whose columns form a basis for the null-spaces $\mathcal{N}(B_1)$ and $\mathcal{N}(B_2)$, respectively, must be constructed.

**2.2. Symmetry-preserving null-space approach for non zero $C$.** As Golub et al. observe [7, Section 6], the null-space method cannot be applied to solve saddle point systems with $C \neq 0_{k,k}$. In a recent paper, Howell [35] proposes what he terms a one-sided application of the null-space method to solve non-singular generalised saddle point systems of the form

$$\mathcal{A}\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} H & B_1^T \\ B_2 & -C \end{pmatrix}\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}, \tag{2.3}$$

in which $H$ and $C \neq 0_{k,k}$ are possibly unsymmetric, $B_1 \in \mathbb{R}^{k\times n}$ and $B_2 \in \mathbb{R}^{k\times n}$ are of full row rank and either $B_1$ and/or $B_2$ is dense. Howell introduces an auxillary variable $w$ that is used to replace the $(n + k) \times (n + k)$ system (2.3) by a larger $(n + 2k) \times (n + 2k)$ system

$$\hat{\mathcal{A}}\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} H & B_1^T & 0_{n,k} \\ B_2 & -C & 0_{k,k} \\ 0_{k,n} & 0_{k,k} & I_k \end{pmatrix}\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} f \\ g \\ 0_{k,1} \end{pmatrix}. \tag{2.4}$$

Interchanging the second and third equations leads to another unsymmetric saddle point system

$$\begin{pmatrix} \hat{H} & \hat{B}_1^T \\ \hat{B}_2 & 0_{k,k} \end{pmatrix}\begin{pmatrix} \hat{u} \\ w \end{pmatrix} = \begin{pmatrix} \hat{f} \\ g \end{pmatrix}, \tag{2.5}$$

with

$$\hat{u} = \begin{pmatrix} u \\ w \end{pmatrix}, \qquad \hat{f} = \begin{pmatrix} f \\ 0_{k,1} \end{pmatrix}, \qquad \hat{H} = \begin{pmatrix} H & B_1^T \\ 0_{n,k} & 0_{k,k} \end{pmatrix}, \qquad \hat{B}_1^T = \begin{pmatrix} 0_{n,k} \\ I_k \end{pmatrix}, \qquad \hat{B}_2 = \begin{pmatrix} B_2 & -C \end{pmatrix}.$$

Because the coefficient matrix in (2.5) is non singular with a zero $(2,2)$ block, the null-space method can be applied. Howell provides further details and presents results for a number of practical applications that have a small number of dense rows and columns. However, a major problem with this approach is that it fails to preserve symmetry. If $B_1 = B_2$ and $H$ and $C$ are symmetric, the symmetry of the original problem is lost because of the swapping of the second and third equations in (2.4); it leads to the need to solve an $n \times n$ unsymmetric system of the form

$$\hat{Z}^T \hat{H} \hat{Z} \hat{v} = \hat{g},$$

where $\hat{Z}$ is a null-space basis matrix for $\hat{B}$. Howell does not comment on this aspect and the use of MATLAB backslash for solving linear systems within his numerical experiments obscures the fact that a symmetric problem has been traded for a larger unsymmetric one.

Our interest is in developing a null-space approach for (1.1) when $C \neq 0_{k,k}$ and $k$ is small that retains symmetry and leads to a transformed saddle point system of the same dimension $n+k$ that has favourable numerical properties, specifically, a positive definite $(1,1)$ block and a $(2,2)$ block of dimension at most $2k$. We consider the general case that both $H$ and $C$ are only SPSD and $B$ may not have full row rank. Lemma 2.1 gives sufficient conditions for $\mathcal{A}$ to be nonsingular. This extends results from [7]; see also [6].

Lemma 2.1. *Let*

$$\mathcal{A} = \begin{pmatrix} H & B^T \\ B & -C \end{pmatrix},$$

*with $H \in \mathbb{R}^{n\times n}$ and $C \in \mathbb{R}^{k\times k}$ SPSD matrices, and $B \in \mathbb{R}^{k\times n}$ is not necessarily of full row rank. If $\mathcal{N}(H) \cap \mathcal{N}(B) = \{0\}$ and $\mathcal{N}(C) \cap \mathcal{N}(B^T) = \{0\}$ then $\mathcal{A}$ is nonsingular.*

*Proof.* Consider a vector $w = \begin{pmatrix} u^T & v^T \end{pmatrix}$ such that $\mathcal{A}w = 0$. This means that $Hu + B^T v = 0$ and $Bu - Cv = 0$. Consequently, $w^T \mathcal{A} w = 0$ implies $u^T H u + v^T C v = 0$. Because $H$ and $C$ are SPSD, $u^T H u = 0$ and $v^T C v = 0$ and it follows that $Hu = 0$ and $Cv = 0$ [33]. Consequently, $B^T v = 0$ and $Bu = 0$. The null-space assumptions then imply that $w = 0$ and hence $\mathcal{A}$ is nonsingular. ◻

The following theorem presents our proposed transformation without $B$ needing to be of full row rank.

Theorem 2.2. *Consider the symmetric saddle point problem (1.1) of order $n+k$ with $rank(B) = r \leq k$, $H$ and $C$ SPSD. Assume furthermore that $\mathcal{N}(H) \cap \mathcal{N}(B) = \{0\}$ and $\mathcal{N}(C) \cap \mathcal{N}(B^T) = \{0\}$. The solution of (1.1) can be obtained by solving a transformed saddle point problem of order $n+k$ with a symmetric positive definite (SPD) principal leading submatrix of order $n-r$.*

*Proof.* Because the saddle-point problem satisfies the assumptions of Lemma 2.1, the system matrix $\mathcal{A}$ is nonsingular. Let

$$E = \begin{pmatrix} Z & Y \end{pmatrix} \in \mathbb{R}^{n\times n} \tag{2.6}$$

with $Z \in \mathbb{R}^{n\times(n-r)}$ having full column rank and such that

$$BE = \begin{pmatrix} 0_{k,n-r} & B_Y \end{pmatrix}, \qquad B_Y = BY \in \mathbb{R}^{k\times r}, \qquad rank(B_Y) = r. \tag{2.7}$$

Set $\mathcal{E}$ to be the $(n+k)\times(n+k)$ matrix

$$\mathcal{E} = \begin{bmatrix} E & 0_{n,k} \\ 0_{k,n} & I_k \end{bmatrix}.$$

Then the transformed saddle point matrix

$$\widetilde{\mathcal{A}} = \mathcal{E}^T \mathcal{A} \mathcal{E} \tag{2.8}$$

is symmetric and non singular. Rewriting this transformation as

$$\begin{aligned}\widetilde{\mathcal{A}} &= \begin{pmatrix} E^T & 0_{k,n} \\ 0_{n,k} & I_k \end{pmatrix} \begin{pmatrix} H & B^T \\ B & -C \end{pmatrix} \begin{pmatrix} E & 0_{n,k} \\ 0_{k,n} & I_k \end{pmatrix} \\ &= \begin{pmatrix} E^T HE & (BE)^T \\ BE & -C \end{pmatrix} \\ &= \begin{pmatrix} \widehat{H} & \widehat{B}^T \\ \widehat{B} & -\widehat{C} \end{pmatrix},\end{aligned}$$

where

$$\widehat{H} = (E^T HE)_{1:n-r,1:n-r} = Z^T HZ \tag{2.9}$$

is the SPD leading principal submatrix of $E^T HE$ of order $n-r$, and

$$\widehat{B} = \begin{pmatrix} (E^T HE)_{n-r+1:n,1:n-r} \\ 0_{k,n-r} \end{pmatrix}, \qquad \widehat{C} = \begin{pmatrix} -(E^T HE)_{n-r+1:n,n-r+1:n} & -B_Y^T \\ -B_Y & C \end{pmatrix}.$$

The transformed system then becomes

$$\widetilde{\mathcal{A}} \begin{pmatrix} \tilde{u} \\ \tilde{v} \end{pmatrix} = \begin{pmatrix} \widehat{H} & \widehat{B}^T \\ \widehat{B} & -\widehat{C} \end{pmatrix} \begin{pmatrix} \tilde{u} \\ \tilde{v} \end{pmatrix} = \begin{pmatrix} \tilde{f} \\ \tilde{g} \end{pmatrix} = \mathcal{E}^T \begin{pmatrix} f \\ g \end{pmatrix}, \tag{2.10}$$

where $\tilde{u} \in \mathbb{R}^{n-r}$ and $\tilde{v} \in \mathbb{R}^{r+k}$ and

$$\tilde{f} = (E^T f)_{1:n-r} \qquad \tilde{g} = \begin{pmatrix} (E^T f)_{n-r+1:n} \\ g \end{pmatrix}.$$

To solve (2.10), consider a conformal partitioning of the vectors $u$ and $f$. Once $\tilde{u}$ and $\tilde{v}$ have been computed, the solution of the original problem (1.1) is given by

$$u = \begin{bmatrix} E_{1:n-r,1:n-r}\tilde{u} \\ E_{n-r+1:n,n-r+1:n}\tilde{v}_{1:r} \end{bmatrix}, \qquad v = (\tilde{v})_{r+1:r+k}.$$

□

Remark 2.1. *The claim in Theorem 2.2 that the transformed saddle point matrix $\tilde{\mathcal{A}}$ given by (2.8) is symmetric and nonsingular cannot be strengthened to it being quasi definite, even when $B$ has full row rank. The problem is that the submatrix $(E^T HE)_{n-r+1:n,n-r+1:n}$ is then only symmetric.*

The $(1,1)$ block $\widehat{H}$ in the transformed system (2.10) is SPD. This allows block factorizations to be employed, which are useful not only for direct methods but also in the construction of preconditioners for use with iterative methods. For example, using the Cholesky factorization $\widehat{H} = L_1 L_1^T$ yields

$$\begin{pmatrix} \widehat{H} & \widehat{B}^T \\ \widehat{B} & -\widehat{C} \end{pmatrix} = \begin{pmatrix} L_1 & \\ L_2 & L_S \end{pmatrix} \begin{pmatrix} I & \\ & -D_S \end{pmatrix} \begin{pmatrix} L_1^T & L_2^T \\ & L_S^T \end{pmatrix}, \tag{2.11}$$

where

$$L_2 = \begin{bmatrix} \tilde{L_2} \\ 0_{1:k,1:n-r} \end{bmatrix} \in \mathbb{R}^{(r+k)\times(n-r)} \quad \text{with} \quad L_2 L_1^T = \widehat{B},$$

and

$$S = \widehat{C} + L_2 L_2^T = \begin{bmatrix} -Y^T HY + \tilde{L_2}\tilde{L_2}^T & -B_Y^T \\ -B_Y & C \end{bmatrix} = L_S D_S L_S^T,$$

where $L_S$ is unit lower triangular and $D_S$ is block diagonal with $1 \times 1$ and $2 \times 2$ blocks.

**2.3. An application: sparse-dense least squares problems.** Consider the linear least-squares (LS) problem

$$\min_x \|Ax - b\|_2,$$

where the system matrix $A \in \mathbb{R}^{m\times n}$ $(m \geq n)$ and the right-hand side vector $b \in \mathbb{R}^m$ are given. The solution $x$ satisfies the $n \times n$ *normal equations*

$$Cx = A^T b, \qquad C = A^T A,$$

where, provided $A$ has full column rank, the *normal matrix* $C$ is symmetric positive definite. Our interest lies in the case where $A$ is large and mainly sparse and includes a relatively small number of rows that are regarded as dense. These rows may be fully dense or have significantly more entries compared to the other rows of $A$ or may contain far fewer entries than $n$ but nevertheless lead to large amounts of fill in $C$. The presence of such rows has long been recognised as a fundamental difficulty in the solution of large LS problems; see, for example, [53] for a discussion and references to past work on this problem.

Assuming the rows of $A$ that are to be treated as dense have been permuted to the end and assuming a conformal partitioning of the vector $b$ (and omitting the row permutation matrix for simplicity), we have

$$A = \begin{pmatrix} A_s \\ A_d \end{pmatrix}, \ A_s \in \mathbb{R}^{m_s\times n}, \ A_d \in \mathbb{R}^{m_d\times n}, \ b = \begin{pmatrix} b_s \\ b_d \end{pmatrix}, \ b_s \in \mathbb{R}^{m_s}, \ b_d \in \mathbb{R}^{m_d},$$

where $m_s$ denotes the number of sparse rows of $A$, and $m_d$ is the number of dense rows, with $m = m_s + m_d$, $m_s \geq n > m_d \geq 1$ and $m_d \ll m_s$. The normal equations become

$$Cx = (C_s + A_d^T A_d)x = c, \qquad c = A_s^T b_s + A_d^T b_d, \tag{2.12}$$

where $C_s = A_s^T A_s$ is the *reduced normal matrix.* The solution of (2.12) can be obtained from the equivalent $(n + m_d) \times (n + m_d)$ system

$$\begin{pmatrix} C_s & A_d^T \\ A_d & -I_{m_d} \end{pmatrix} \begin{pmatrix} x \\ A_d x \end{pmatrix} = \begin{pmatrix} c \\ 0_{md,1} \end{pmatrix}. \tag{2.13}$$

This is of the form (1.1) with $k = m_d$, $H = C_s$, $B = A_d$ and $C = I_{m_d}$. If $A_s$ has full column rank, $C_s$ is SPD. But in practice $A_s$ may contain one or more null columns, and $C_s$ is then singular with a corresponding number of null rows and columns. Even if $C_s$ has no null columns, it can be singular or highly ill-conditioned. Recent studies have proposed ways of circumventing the problem of null columns in $A_s$. In [53, 54], it was shown how they can be dealt with either by perturbing diagonal entries of $C_s$ before applying a direct solver or by solving a number of related sparse LS problems and combining their solutions to give the solution of the original problem. Unfortunately, both methods incur overheads, with the former requiring the sparse direct solver is combined with an iterative solver and the latter needing the solution of a number of LS problems. These disadvantages led us to consider employing matrix stretching [55, 56]. The main aim of matrix stretching is to split each of the dense rows (that is, each row of $A_d$) into a number of sparser rows and to formulate a (larger) modified problem from which the solution to the original problem can be derived. This has the advantage that, provided $A$ is of full rank, the issue of null columns does not arise. However, when $A_s$ is very sparse, the dimensions of the stretched system can grow rapidly with $m_d$ so that the stretched LS problem can be considerably larger than the original problem. It may also be highly ill-conditioned. Thus we remain interested in developing alternative strategies and this was a key motivation behind the current work on null-space approaches.

The following result shows that, in the case of the full rank LS problem (2.13), the assumptions of Theorem 2.2 are satisfied, with $r \leq m_d$ the rank of $A_d$.

LEMMA 2.3. *Consider* $A = \begin{pmatrix} A_s \\ A_d \end{pmatrix}$, $A_s \in \mathbb{R}^{m_s\times n}$, $A_d \in \mathbb{R}^{m_d\times n}$. *If $A$ is of full rank, then* $C_s = A_s^T A_s$ *is positive definite on the null space of* $A_d$.

*Proof.* Let $v \in \mathcal{N}(A_d)$ and consider

$$v^T C_s v = (A_s v)^T A_s v \geq 0.$$

If $A_s v = 0$ then because $A_d v = 0$ it follows that $Av = 0$. Because $A$ is of full rank, this implies $v = 0$. Hence $v^T C_s v > 0$ and $C_s$ is positive definite on $\mathcal{N}(A_d)$. □

Thus, provided a basis for $\mathcal{N}(A_d)$ can be constructed, the proposed null-space approach offers an alternative way of solving sparse-dense LS problems, even in the case that $C_s$ is SPSD.

**3. Null-space basis construction.** Solving large-scale saddle point systems using null-space methods leads to the problem of finding null-space bases that preserve sparsity and lead to a stable transformed system. Before looking at null-space basis construction for wide matrices, we give a brief overview of the historical development of techniques for constructing null-space bases of sparse matrices.

**3.1. Construction of null-space bases of sparse matrices.** A null-space basis matrix $Z$ for $B \in \mathbb{R}^{k \times n}$ of rank $r > 0$ can sometimes be obtained directly by analyzing the problem [14] but more advanced methods are normally necessary. In structural analysis, early methods were based on factorizations of $B$ and focused on the sparsity of $Z$, with emphasis on $Z$ having a banded or skyline sparsity pattern. Computation of $Z$ based on an initial LU factorization of $B$ was proposed by Topçu [61] (see also [38, 58]). A specific strategy of this kind, called the *turnback* method [38, 61], attracted interest in the late 1970s/early 1980s (see also an early parallel implementation [11] and its recent use in practice in [18]). The turnback (backward looking) method finds null vectors by expressing $n - r$ chosen *start columns* of $B$ as linear combinations of a small set of previously numbered columns. The initial LU factorization serves to determine these start columns. Linear independence of the columns of $Z$ is guaranteed by not using the leftmost columns in these linear combinations from any set computed for the remaining start columns. An overview and modifications to the basic approach (such as replacing LU by a QR factorization) are described in [30]; see also [12] for further refinements. More recently, factorization approaches for the computation of null-space bases of sparse matrices are discussed in [26, 57].

Theoretical and algorithmic breakthroughs based on bipartite graph matchings and matroid theory came with the 1984 thesis of Pothen [50] and subsequent papers. The contributions covered not only new algorithmic approaches but also complexity concepts for the related problem of finding the *sparsest null-space basis* of $B$, that is, for finding $Z$ with the smallest possible number of non zero entries (see also [16]). It has been shown that such a basis can be constructed by a greedy algorithm that assembles the basis using a sequential choice of the sparsest vectors belonging to $\mathcal{N}(B)$. While this can provide extremely sparse $Z$, there are two important practical limitations. Firstly, finding the sparsest vectors of a null-space basis for a general constraint matrix is NP-hard. Secondly, sparsity of $Z$ may not be enough and its numerical properties must also be considered. Sophisticated proposals given in [17, 24, 50] were designed to compute sparse null-space bases of sparse matrices and they led to efficient algorithms for computing so-called *fundamental* and *triangular* null-space bases.

Origins of another line of research appear in the 1969 seminal paper of Henderson and Maunder [20] (see also [13]). Subsequently, a number of related algorithms for specific applications have been developed independently. These exploit the graph of $B$ and, in particular, cycles in the graph. The idea is most transparent if $B$ is the vertex-edge incident matrix of some underlying graph. In structural mechanics, cycles of the graph that describes the interconnection of separate substructures of a skeletal structure can be considered. Using the graph cycles, a set of independent null-space vectors that form columns of $Z$ can be computed [20, 45]. This construction was developed and discussed in the context of other approaches in [51]. In some partial differential equation applications the cycle approach is relevant because a simple constraint structure may be implied by discretization schemes [1, 2, 3, 4, 29]. Many publications discuss using cycles in the graph: pointers to relevant literature can be found in [39, 40]. There are heuristic ways of finding the cycle basis, a task that is straightforward if the underlying graph is planar [29]. More generally, construction may be based on a spanning tree of the graph [50]. Procedures to find sparse cycle bases in this way are given in [21] (see also [34, 59]). However, discretizations of three-dimensional grids may not allow sparse cycle bases to be found [4]. As with sparse factorization approaches, those based solely on local computations of $Z$ may suffer from ill-conditioning.

An interesting motivation is to construct $Z$ using structure, for example, by allowing a permutation to block angular form. Such an approach was introduced in [48]. Closely related proposals to compute $Z$ while explicitly exploiting substructuring, motivated in part by parallel computing, were given in [37, 49] (see also [36]). The block form can be obtained algebraically, as in the nested dissection ordering discussed in [59]; see also [48].

**3.2. Null-space basis for wide matrices.** In contrast to much of the previous work on constructing null-space bases, our focus is on wide matrices $B \in \mathbb{R}^{k\times n}$ with $k \ll n$. $B$ may be sparse but, because $k$ is small, we may want to treat $B$ as dense (although the null-space basis construction may still involve sparse operations). As it can happen in practical applications, we allow for $B$ being rank deficient ($rank(B) = r < k$). We want the nearly-square null-space basis matrix $Z \in \mathbb{R}^{n\times(n-r)}$ to be sparse but we are also concerned about conditioning. Our experiments consider the quality of $Z$ and, as we employ a sparse direct solver to factorize the $(1,1)$ block $Z^THZ$ given by (2.9) of the transformed system, we need $Z$ to have no dense rows and be such that $Z^THZ$ is sparse. Orthogonal transformations inevitably lead to dense $Z$ and so, although useful in other situations (see, for instance, [42]), are not appropriate here. Because $B$ is wide, each column of $Z$ can be computed independently and such that each has small support (the entries can be computed as the coefficients of linear combinations of at most $k$ other columns). Small support means that there is always some orthogonality but $Z$ may still be ill conditioned. Indeed, if $B$ is an adjacency matrix of a domain discretized by mixed hybrid finite elements, then the condition number of the null-space basis of the constraints that express the hybridization with small support can grow like $h^{-2}$, where $h$ is the discretization parameter [4]. Thus we propose employing a QR factorization that incorporates pivoting for stability and, because $B$ has only a small number of rows, this is not prohibitively expensive.

**3.2.1. Null-space bases with local support.** This section discusses constructing (a permutation of) $Z$ by finding linearly dependent sets of columns with close column indices. The coefficients of the linear combination of these columns that sum to zero are the values of the entries in $Z$. These bases are similar to those belonging to the category of triangular null-space bases [17], but employment of QR factorizations with pivoting leads to $Z$ having a more general sparsity pattern. Our focus is on the numerical qualities of the basis combined with limiting the number of entries through the employment of a threshold pivoting strategy (thus, as is common with sparse matrix factorizations, there is a trade off between stability and sparsity).

We begin with the simple case in which $B$ comprises a single fully dense row ($k = 1$) and consider two extreme cases for the structure of a sparse null-space basis $Z \in \mathbb{R}^{n\times(n-1)}$. First, $Z$ can be constructed to have non zeros only in its first row and on the diagonal of $Z_{2:n,1:n-1}$. The entries in row 1 are obtained by dividing entries 2 to $n$ of $B$ by the entry in its $(1,1)$ position. Column pivoting is needed to ensure that this entry is large. The resulting $Z$ is sparse but $Z^THZ$ is dense and so is not of interest here.

The second extreme case constructs $Z$ to be a permutation of a matrix with a banded structure; it is presented as Algorithm 2 and illustrated in Figure 3.1. When $B \in \mathbb{R}^{1\times n}$ we have $Q = I_1$, but we include the Q factor in the algorithm description because it will be needed in subsequent more general algorithms. The R factor is a single row with its first entry $\beta_1 \neq 0$. The computed $Y$ (recall (2.6)) is sparse, has full rank, is well-conditioned because of the column pivoting and belongs to the space of vectors $Z^\perp$ [22, 52]. In particular, $BY = 1$. We remark that a similar approach was recently proposed by Howell [35, Algorithm 3].

A dominant part of the saddle-point matrix (2.10) is its $(1,1)$ block $Z^THZ$. Its condition number $cond(Z^THZ)$ is bounded by

$$cond(Z^THZ) \leq cond(Z_0^THZ_0)\, cond(Z^TZ), \tag{3.1}$$

where $Z_0$ is an orthogonal null-space matrix (see Lemma 10 in [47]). Consequently, it is desirable for $cond(Z^TZ)$ to be small. Constructing $Z$ so that $P^TZ$ has a narrow band may not be sufficient to guarantee

**Algorithm 2** Let $B \in \mathbb{R}^{1\times n}$ be dense. Construct a full rank matrix $E = (Z \quad Y) \in \mathbb{R}^{n\times n}$ such that $Z \in \mathbb{R}^{n\times(n-1)}$ is a null basis matrix for $B$ and $Y$ is well-conditioned.

```
1: Factorize $BP = QR$, where $P \in \mathbb{R}^{n\times n}$ is a column permutation matrix such that $\beta_1 = (R)_{1,1} \neq 0$ is
   large and $Q = I_1$. Denote the entries of $R$ by $\big(\beta_1, \ldots, \beta_n\big)$.
2: Initialize $\tilde{Z} = 0_{n,n-1}$.
3: for $l = 1, ..., n-1$
4:     Set $(\tilde{Z})_{l+1,l} = 1$, $(\tilde{Z})_{l,l} = -\beta_{l+l}/\beta_l$.
5: end for
6: Set $E = \big(Z \quad Y\big)$ with $Z = P\tilde{Z}$, $Y = P\tilde{Y}$ for $\tilde{Y} = (1/\beta_1)\, e_1$.
```

$$B = \begin{pmatrix}1 & 2 & 3 & 10 & 4\end{pmatrix}, \qquad P = \begin{pmatrix}e_4 & e_2 & e_3 & e_1 & e_5\end{pmatrix}, \qquad BP = \begin{pmatrix}10 & 2 & 3 & 1 & 4\end{pmatrix},$$

$$\tilde{Z} = \begin{pmatrix} -0.2 & & & \\ 1 & -1.5 & 0 & 0 \\ 0 & 1 & -1/3 & 0 \\ 0 & 0 & 1 & -4 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \qquad \tilde{Y} = \begin{pmatrix}0.1\\0\\0\\0\\0\end{pmatrix}, \qquad E = \begin{pmatrix} 0 & 0 & 1 & -4 & 0 \\ 1 & -1.5 & 0 & 0 & 0 \\ 0 & 1 & -1/3 & 0 & 0 \\ -0.2 & 0 & 0 & 0 & 0.1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

FIG. 3.1. *Simple example with $n = 5$ and $r = 1$ showing $B \in \mathbb{R}^{1\times 5}$, $P \in \mathbb{R}^{5\times 5}$, the permuted matrix $BP$, $\tilde{Z}$ and $\tilde{Y}$ computed using Algorithm 2. E satisfies (2.7).*

this, as can be deduced from the following result for the idealized example of $B$ comprising a single row of all ones.

LEMMA 3.1. *Let $B \in \mathbb{R}^{1\times n}$ be a row vector of all ones and construct $Z$ using Algorithm 2. Then the condition number of $Z^T Z \in \mathbb{R}^{(n-1)\times(n-1)}$ is asymptotically of order $n^2$.*

*Proof.* The permutation $P$ is the identity and the computed $Z$ is bidiagonal with diagonal entries equal to $-1$ and off diagonal nonzeros equal to 1. $Z^T Z$ is a well-known Laplacian matrix with eigenvalues given by

$$2\cos(1 - \frac{j\pi}{n}),\ 1 \le j \le n-1.$$

Consequently, the largest eigenvalue goes asymptotically to 4 and the smallest one is

$$4\sin^2\frac{\pi}{2\,n} \approx \frac{\pi^2}{n^2}.$$

The result follows. □

Algorithm 3 considers general $B$ with $1 \le k < n$ rows and rank $r$ such that $0 < r \le k$. The basic idea is to express columns of $Z$ as linear combinations of previous columns with close indices. $r$ is computed using a QR factorization with column pivoting. The columns of $B$ corresponding to the first $r$ columns of the R factor are permuted to the front and the remaining $n - r$ columns are marked as dependent; they correspond to the columns of $Z$. They are computed independently while aiming to balance sparsity and numerical stability. As in other sparse numerical linear algebra algorithms that need to combine locality with ensuring stability, this is achieved using threshold column pivoting. The principle behind threshold pivoting is as follows. Assume a threshold parameter $\theta$ is given with $0 < \theta \le 1$. At a step of the QR factorization, assume that $\Delta$ is the maximum norm of all the columns eligible to be chosen as the next pivot column. While full column pivoting selects the pivot column to be an eligible column with norm $\Delta$, threshold pivoting chooses the pivot column to be the eligible column with norm at least $\theta\Delta$ and such that its index is closest to the index of the first (starting) column of the factorized matrix. In Algorithm 3, we perform QR factorizations repeatedly in the loop commencing at Line 3 for a set of $n - r$ starting columns. The initial permutation $P$ guarantees to find $r$ and determine the $n - r$ starting columns, that is, the columns that are expressed in the loop as linear combinations of previous columns in the order determined by $P$. The local permutation $P_L$ reverses the column order so that threshold column pivoting chooses columns closest to the starting column first. The coefficients are then obtained by performing QR factorizations (with threshold partial pivoting) of submatrices of $BP$. While Algorithm 3 is somewhat technical, it essentially combines two steps: it first determines $BP$ and then solves $n-r$ subproblems for the entries of the $n-r$ columns of $Z$. Permutations ensure the subproblems always find these entries. The matrix $E$ computed by Algorithm 3 satisfies (2.7).

**Algorithm 3** Given $B \in \mathbb{R}^{k\times n}$ with $rank(B) = r \leq k$ and a pivoting threshold $0 < \theta \leq 1$, construct a full rank matrix $E = (Z \;\; Y) \in \mathbb{R}^{n\times n}$ such that $Z \in \mathbb{R}^{n\times(n-r)}$ belongs to $\mathcal{N}(B)$.

```
1:  Factorize $\tilde{B} = BP = Q\begin{pmatrix} R_1 & R_2 \\ 0_{k-r,r} & 0_{k-r,n-r}\end{pmatrix}$ with threshold column pivoting.
        $P \in \mathbb{R}^{n\times n}$ is a permutation matrix , $Q \in \mathbb{R}^{k\times k}$ is an orthogonal matrix,
        $R_1 \in \mathbb{R}^{r\times r}$ is an upper triangular matrix of full rank.
2:  Initialize $\tilde{Z} = 0_{n,n-r}$.
3:  for $l = r+1, ..., n$
4:      if $(\tilde{B})_{:,l}$ is zero column then
5:          Set $(\tilde{Z})_{:,l-r} = e_l$.
6:      else
7:          Set $X_l = (\tilde{B})_{:,1:l-1}P_L$, $P_L \in \mathbb{R}^{(l-1)\times(l-1)}$, $P_L e_i = e_{l-i}$, $i = 1, \ldots, l-1$.
            (That is, $X_l$ is $(\tilde{B})_{:,1:l-1}$ with its columns in reverse order.)
8:          Compute $r$ steps of the QR factorization $X_l P_T = \hat{Q}\hat{R}$ with threshold column pivoting,
            where $\hat{R} \in \mathbb{R}^{r\times(l-1)}$ is upper triangular and of rank $r$.
9:          Set $\tilde{R} = (\hat{R})_{1:r,1:r}$.
10:         Compute $x = \tilde{R}^{-1}\hat{Q}^T(\tilde{B})_{:,l}$.
11:         Set $(\tilde{Z})_{l,l-r} = -1$.
12:         for $i = 1, ..., r$
13:             Set $(\tilde{Z})_{j,l-r} = (x)_i$ for $j$ satisfying $e_j^T(P_L P_T)_{:,i} \neq 0$.
                (Note that for each $i$ there is always one and only one $j$ that satisfies this.)
14:         end for
15:     end if
16: end for
17: Set $E = \begin{pmatrix} Z & Y \end{pmatrix}$ with $Z = P\tilde{Z}, Y = P\begin{pmatrix} I_r & 0_{r,n-r}\end{pmatrix}^T$.
```

Incorporating threshold column pivoting in the QR factorizations in Algorithm 3 implies that, provided the parameter $\theta$ is not too small, the matrix $BY$ should be reasonably well-conditioned. Moreover, it guarantees that any dependent column $(\tilde{B})_{:,l}$ of $\tilde{B}$ can be stably expressed as a linear combination of previous columns. Note that the coefficients of this linear combination are expressed using the inverse of $\tilde{R}$. Because each such column is used in the construction of just one column of $Z$, the columns of $Z$ must be linearly independent. Observe that a null column $(\tilde{B})_{:,l}$ implies a unit column in $Z$. Note also that Line 10 is well defined because the permutation $P$ guarantees not only that there is a submatrix of rank $r$ in each $X_l (l = r+1, \ldots, n)$ but also that $\hat{Q}^T(\tilde{B})_{:,l}$ cannot have non zeros in rows $r+1, \ldots, k$.

Figure 3.2 shows the basis matrix $Z$ constructed using Algorithm 3 for $B \in \mathbb{R}^{2\times 6}$ of full rank; $\theta$ is set to 0.1. With this threshold choice, it is not necessary to permute column 6 of $B$ (the one with largest norm) to the front and hence $P = I$ and $B = \tilde{B}$. The loop starting in Line 3 considers the last $n-r = 4$ columns of $\tilde{B}$ and expresses each as a linear combination of the closest previous columns, successively accessing the columns in reverse order; the initial permutation $P$ guarantees this is possible and each column provides a column of $Z = \tilde{Z}$. For example, $B_{:,4} = 2B_{:,3} - B_{:,2}$, giving entries $-1, 2, -1$ in rows 2, 3 and 4 of the second column of $Z$.

**3.2.2. Fundamental null-space basis.** We next discuss constructing the so-called fundamental null-space basis, which was first mentioned in an unpublished 1962 paper by Wolfe [62] (see [17]). Let

$$B = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 8 \\ 2 & 3 & 4 & 5 & 6 & 9 \end{pmatrix}, \quad \tilde{Z} = Z = \begin{pmatrix} -1 & & & \\ 2 & -1 & & \\ -1 & 2 & -1 & \\ & -1 & 2 & -3 \\ & & -1 & 4 \\ & & & -1 \end{pmatrix}, \quad Y = \begin{pmatrix} 1 & \\ & 1 \\ & \\ & \\ & \\ & \end{pmatrix}.$$

FIG. 3.2. *Simple example with $n = 6$ and $r = k = 2$. $Z$ and $Y$ are constructed using Algorithm 3; $E = (Z \ \ Y)$ satisfies (2.7).*

$B \in \mathbb{R}^{k\times n}$ have full row rank and assume the columns have been permuted so that $G = (B)_{1:k,1:k}$ is nonsingular. Then the *fundamental null-space basis* of $B = \begin{pmatrix} G & N \end{pmatrix}$ is defined to be

$$Z = \begin{pmatrix} -G^{-1}N \\ I_{n-k} \end{pmatrix}. \tag{3.2}$$

For $rank(B) = r < k$, let

$$BP = Q \begin{pmatrix} R_1 & R_2 \\ 0_{k-r,k-r} & 0_{r,n-r} \end{pmatrix} \quad \text{and} \quad P_1 B P_2 = \begin{pmatrix} L_1 & \\ L_2 & I_{k-r,k-r} \end{pmatrix} \begin{pmatrix} U_1 & U_2 \\ 0_{r,r} & 0_{r,n-r} \end{pmatrix}$$

be the pivoted QR and LU factorizations of $B$, respectively. The fundamental null-space basis of $B$ may be expressed as either

$$Z = P \begin{pmatrix} -R_1^{-1} R_2 \\ I_{n-r} \end{pmatrix} \tag{3.3}$$

or

$$Z = P_2 \begin{pmatrix} -U_1^{-1} U_2 \\ I_{n-r} \end{pmatrix}. \tag{3.4}$$

For the LU factorization, our experience is that complete pivoting should be used for stability.

The biconjugation process of Hestenes [31] applied to a square matrix $A \in \mathbb{R}^{n\times n}$ yields a pair of biconjugate matrices $(\tilde{V}, \ \ V)$ such that $\tilde{V}AV$ is diagonal. Biconjugate factorizations were discussed in [15] and some practical extensions were given by Benzi [5]; see also the direct projection method [8, 60]. Applying biconjugation approximately leads to so-called approximate inverse preconditioners [10]. We define a partial biconjugation factorization applied to $B \in \mathbb{R}^{k\times n}$ with $rank(B) = r$, $0 < r \leq k \leq n$, to be a process that yields $V \in \mathbb{R}^{n\times n}$ and a lower trapezoidal $L \in \mathbb{R}^{k\times n}$ satisfying

$$BV = L. \tag{3.5}$$

We term this the *right oblique conjugation*. Assuming $r$ steps of an LDU factorization of $B$ can be performed without pivoting, Algorithm 4 computes a null-space basis matrix $Z \in \mathbb{R}^{(n-r)\times n}$ by right oblique conjugation. It initializes $V$ to be the identity. The columns of $V$ are then transformed so that, from Line 4, $b_i^T v_j^{(i)} = 0$. At the end of the algorithm, $BV$ is lower trapezoidal. This means, in particular, that the last $n-r$ columns form a null-space basis because they are linearly independent. To see the mutual relationship between oblique conjugation and LDU factorization, first consider the 'non-wide' square case ($r = k = n$). In this instance, $V = U^{-1}$, where $U$ is the U factor of the LDU factorization of $B$ without pivoting. This result follows from the fact that $V$ is unit upper triangular by construction, the uniqueness of the LDU factorization without pivoting and (3.5) is satisfied (see [9]). Now consider $r \leq k < n$. Because the construction commences with $V = I_n$, and all the updates at Line 4 are well defined, the computed $V$ is upper triangular and its last $n - r$ columns have the shape of (3.2), where $G$ represents the principal leading submatrix of $B$ of dimension $r$. In the algorithm outline, for simplicity, we omit pivoting. Even with column pivoting, an assumption is needed to guarantee the factorization exists. It is sufficient to

**Algorithm 4** Right oblique conjugation. Given $B \in \mathbb{R}^{k\times n}$ with $rank(B) = r \le k$ and rows $b_1^T, \ldots, b_k^T$, construct $V \in \mathbb{R}^{n\times n}$ such that $BV$ is lower trapezoidal and $Z \in \mathbb{R}^{(n-r)\times n}$ belongs to $\mathcal{N}(B)$.

```
1: Initialize $(v_1^{(0)}, \ldots, v_n^{(0)}) = I_n$
2: for $i = 1, \ldots, r$
3:     for $j = i+1, \ldots, n$
4:         Compute $v_j^{(i)} := v_j^{(i-1)} - \left(\frac{b_i^T v_j^{(i-1)}}{b_i^T v_i^{(i-1)}}\right) v_i^{(i-1)}$
5: end for
6: Set $V = \left(v_1^{(0)}, v_2^{(1)}, \ldots,\ v_r^{(r-1)},\ v_{r+1}^{(r)}, \ldots, v_n^{(r)}\right)$.
7: Set $Z = \left(v_{r+1}^{(r)}, \ldots, v_n^{(r)}\right)$.
```

assume linear independence of the first $r$ rows of $B$, which is weaker than requiring the first $r$ steps of the LDU factorization of $B$ can be performed without pivoting. We have the following straightforward result.

Lemma 3.2. *Let the first $r$ rows of $B \in \mathbb{R}^{k\times n}$ be linearly independent and let $rank(B) = r$. Apply right oblique conjugation to $B$. Then there exists a permutation $P \in \mathbb{R}^{n\times n}$ and a block partitioning of $BP = \begin{pmatrix} G & N \\ D & C \end{pmatrix}$ with $G \in \mathbb{R}^{r\times r}$ such that*

$$\begin{pmatrix} G & N \end{pmatrix} \begin{pmatrix} W & S \\ 0_{n-r,r} & I_{n-r} \end{pmatrix} = \begin{pmatrix} I_r & 0_{r,n-r} \end{pmatrix}, \tag{3.6}$$

*for some $W \in \mathbb{R}^{r\times r}$ and $S \in \mathbb{R}^{r\times(n-r)}$. In particular,*

$$Z = \begin{pmatrix} S \\ I_{n-r} \end{pmatrix}$$

*is the fundamental null-space basis corresponding to $\begin{pmatrix} G & N \end{pmatrix}$.*

*Proof.* Setting $P$ to be the permutation matrix corresponding to column pivoting within the LU factorization and using the uniqueness of the first $r$ steps of the LU factorization of $BP$, we have $W = G^{-1}$. Moreover, $GS + N = 0_{r,n-r}$, from which it follows that

$$\begin{pmatrix} S \\ I_{n-r} \end{pmatrix} \equiv \begin{pmatrix} -G^{-1}N \\ I_{n-r} \end{pmatrix}$$

is the fundamental null-space basis of the matrix $B$ with permuted columns. ∎

One reason why fundamental null-space bases using right oblique conjugation are of potential interest is the practical implementation of Algorithm 4 and relates to the fact that the one-sided factorization does not split the inverse of $G$ into two factors. In practice, pivoting needs to be incorporated to maximize the magnitude of the quantity $b_i^T v_i^{(i-1)}$ used in Line 4. While the rows of $B$ can be stored in static data structures, the columns of $V$ that are eligible to be pivot columns (those with indices $j = i+1, \ldots, n$) are updated and so need to be stored in a (single) dynamic data structure. By contrast, if the basis is constructed using a standard LU factorization, two sets of vectors (those that form the L and U factors) change dynamically and so must be stored using two dynamic data structures. This storage difference may be important when $k$ is large (which is not the case in this paper). Note that the one-sided factorization is a useful way to explain relations among closely related computational approaches [44]. In exact arithmetic, the diagonal entries $b_i^T v_i^{(i-1)}$ are inverses of the diagonal entries of the LDU factorization (see [5]). In finite precision arithmetic, there are other, and possibly more stable, ways of computing the pivots that we do not discuss here; theoretical properties of some biconjugation variants (with additional assumptions) are discussed in [41]. In the case $k \ll n$, complete pivoting is not prohibitively expensive and may be needed for stability.

A further attraction of right oblique conjugation is its flexibility. It was introduced as a way to obtain the fundamental null-space basis, but it can be modified to give other null-space bases, including banded ones. To see this, consider the inner loop of Algorithm 4. The oblique projection in Line 4 projects the vectors $v_j^{(i-1)}$ along $v_i^{(i-1)}$ onto the space $b_i^{\perp}$. The oblique projection can be rewritten using the projector

$$\left(I - \frac{v_i^{(i-1)} b_i^T}{b_i^T v_i^{(i-1)}}\right),$$

where the vectors $v_i^{(i-1)}, \ldots, v_n^{(i-1)}$ belong to the space $b_1^{\perp} \cap \ldots \cap b_{i-1}^{\perp}$ (see, for example, [8]). But $v_i^{(i-1)}, \ldots, v_n^{(i-1)}$ can be projected along any other set of linearly independent vectors that are not equal to the $v_i^{(i-1)}$. Algorithmically, the projection can be replaced by

$$v_j^{(i)} = \left(I - \frac{x_j b_i^T}{b_i^T x_j}\right) v_j^{(i-1)},\ j = i+1, \ldots, n,$$

where $x_2, \ldots, x_n$ are linearly independent. The real power to construct the null-space bases via oblique projections is apparent from the following result. It shows that Algorithm 2, which obtains the null-space basis $Z$ by permuting the upper bidiagonal matrix $\tilde{Z}$, can be cast in the form of projections. We state the result without proof because it can be verified by direct checking.

Lemma 3.3. *Algorithm 2 can be rephrased in terms of the general scheme of Algorithm 4 by using in Line 4 the oblique projections*

$$z_j^{(i)} = \left(I - \frac{x_j b_i^T}{b_i^T x_j}\right) z_j^{(i-1)},\ j = i+1, \ldots, n, \tag{3.7}$$

*with* $x_j = e_{j-1}$, $j = 2, \ldots, n$.

**3.2.3. Composite null-space basis for wide dense matrices.** There are situations in which the null-space basis can be constructed from partial null-space bases in several steps, instead of the single pass of Algorithm 3. This may simplify the computation, allow it to be more parallel, or to be useful in specific cases, for example, when $B$ has a particular block structure. The following is a straightforward extension of Theorem 6.4.1 of [25] for rank deficient blocks and non-orthonormal bases (a slightly less general version was described in [35]).

Theorem 3.4. *Consider the null-space basis $Z_F \in \mathbb{R}^{n\times(n-r_1)}$ of the matrix $F \in \mathbb{R}^{k_1\times n}$ of rank $r_1$ and the null-space basis $Z_G \in \mathbb{R}^{(n-r_1)\times(n-r_1-r_2)}$ of the matrix $GZ_F \in \mathbb{R}^{k_2\times(n-r_1)}$ of rank $r_2$, where $G \in \mathbb{R}^{k_2\times n}$. Then the columns of the matrix $Z_F Z_G \in \mathbb{R}^{n\times(n-r_1-r_2)}$ form a basis of $\mathcal{N}(F)\cap\mathcal{N}(G)$.*

This result allows Algorithms 2 and 3 to be generalised by adding rows (or blocks of rows) sequentially to $B$. An important application for such a construction is when a sequence of problems is generated by successively modifying $B$ through the addition of further rows. A special case of a procedure of this type was proposed by Howell [35]. To demonstrate the mechanism, we introduce Algorithm 5 that applies Algorithm 3 repeatedly to $\tau \geq 1$ row blocks of $B$ and the null-space basis construction exploits Theorem 3.4. For simplicity, we assume that the rows of $B$ are dense and $rank(B) = k$ (so that all the row blocks are also of full rank). In practice, $B$ may contain zeros that fill in as the algorithm proceeds.

An important practical application of Algorithm 5 is $B$ having a non-trivial block angular form as this can be exploited by to reduce the work. Assume that $B$ (of full row rank) can be ordered to the block angular form

$$B = \begin{pmatrix} \bar{B}_1 & & & & \bar{D}_1 \\ & \bar{B}_2 & & & \bar{D}_2 \\ & & \ddots & & \vdots \\ & & & \bar{B}_s & \bar{D}_s \end{pmatrix}, \tag{3.8}$$

where $\bar{B}_i \in \mathbb{R}^{k_i\times n_i}$ and $\bar{D}_i \in \mathbb{R}^{k_i\times n_d}$. If $\mathcal{N}(B)$ can be composed from the null-space bases of the $\bar{B}_i$ extended by zeros outside their domains, then the null-space basis matrix $Z$ is of block angular form; see also the framework of substructuring in [49].

**Algorithm 5** Given $B \in \mathbb{R}^{k\times n}$ with $rank(B) = k$ and $\tau$ row blocks, construct $Z \in \mathbb{R}^{(n-k)\times n}$ belonging to $\mathcal{N}(B)$.

```
1: Split $B$ into $\tau$ row blocks given by
       $$B = \left(B_1^T, \ldots, B_\tau^T\right)^T,$$
   with $B_i \in \mathbb{R}^{k_i\times n}, i = 1, \ldots, \tau$, with $\sum_1^\tau k_i = k$.
2: Initialise $\lambda_0 = 0$ and $Z = I$.
3: for $j = 1, ..., \tau$
4:     Set $\lambda_j = \lambda_{j-1} + k_j$.
5:     Construct the null space basis $Z_j \in \mathbb{R}^{(n-\lambda_{j-1})\times(n-\lambda_j)}$ of $B_j Z$ using Algorithm 3.
6:     Set $Z = ZZ_j$, $Z \in \mathbb{R}^{n\times(n-\lambda_j)}$.
7: end for
```

In our experiments, we consider the special case in which each row block contains a single row but $B$ is not assumed to have full row rank. The approach is given in Algorithm 6 and illustrated in Figure 3.3. It constructs the null-space basis $Z$ as a product of bases $Z_i$ corresponding to the rows of $B$. Let $(\tilde{B})_{1:i,:}$ be the $i \times n$ matrix with rows $\tilde{b}_j^T$, $j = 1, \ldots, i$. Then at Line 4, $l_i = l - 1$ if $rank((\tilde{B})_{1:i,:}) > rank((\tilde{B})_{1:i-1,:})$ and $l_i = l$ otherwise.

**Algorithm 6** Given $B \in \mathbb{R}^{k\times n}$ with rows $b_1^T, \ldots, b_k^T$, construct $Z$ belonging to $\mathcal{N}(B)$ as a product of null-space bases corresponding to the rows of $B$.

```
1: Initialize $Z = I_n \in \mathbb{R}^{n\times n}$ and set $l = n$
2: for $i = 1, ..., k$
3:     Compute $\tilde{b}_i^T = b_i^T Z \in \mathbb{R}^{1\times l}$
4:     Construct a null-space basis $Z_i \in \mathbb{R}^{l\times l_i}$ of $\tilde{b}_i^T$ using Algorithm 2.
5:     Set $l = l_i$
6:     Compute $Z = ZZ_i \in \mathbb{R}^{n\times l}$.
7: end for
```

$$B = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 8 \\ 2 & 3 & 4 & 5 & 6 & 9 \\ 3 & 4 & 5 & 6 & 7 & 8 \end{pmatrix}.$$

$$Z_1 = \begin{pmatrix} 2 & & & & \\ -1 & 3/2 & & & \\ & -1 & 4/3 & & \\ & & -1 & 5/4 & \\ & & & -1 & 8/5 \\ & & & & -1 \end{pmatrix}, \quad Z_2 = \begin{pmatrix} 1/2 & & & \\ -1 & 2/3 & & \\ & -1 & 3/4 & \\ & & -1 & 12/5 \\ & & & -1 \end{pmatrix}, \quad Z_3 = \begin{pmatrix} -1 & & \\ & -1 & \\ & & -1 \end{pmatrix}.$$

$$Z = Z_1 Z_2 Z_3 = \begin{pmatrix} -1 & & \\ 2 & -1 & \\ -1 & 2 & -1 \\ & -1 & 2 \\ & & -1 \\ & & \end{pmatrix}.$$

FIG. 3.3. *Simple* $3 \times 6$ *example illustrating Algorithm 6.*

**4. Numerical experiments.** In this section, we present numerical results to illustrate the potential of the proposed approaches for computing null space bases for solving symmetric saddle point problems with a small non zero $(2,2)$ block. Unless stated otherwise, our test examples are from the SuiteSparse Matrix Collection [19]. They are least squares problems that were used in our earlier study [53] and so we use the formulation (2.13). The row block $A_d$ is identified using the variant of the approach of Meszaros [46] described in [56]. Having computed a null-space basis, we employ the sparse direct solver `HSL_MA87` [32] from the HSL mathematical software library[1] to compute the Cholesky factorization within the block factorization (2.11). All reported norms are Euclidean norms.

**4.1. Results for problem `aircraft`.** We first report detailed findings for problem `aircraft`. It has dimensions $m = 7517$ and $n = 3754$ with $nnz(A) = 20,267$ (here and elsewhere, for any matrix $H$, $nnz(H)$ denotes the number of non zero entries, with the number set to those in the lower triangular part when $H$ is symmetric). $A_d \in \mathbb{R}^{m_d \times n}$ comprises $m_d = 17$ rows; the sparse row block $A_s$ has 4 null columns and thus the normal matrix $C_s = A_s^T A_s$ is rank deficient.

Figure 4.1 demonstrates the effect of varying the threshold parameter $\theta \in [0.001, 1]$ in Algorithm 3 on the sparsity of the null-space basis matrix $Z$ and on the transformed normal matrix $Z^T C_s Z$. It confirms the significant advantage of using a small $\theta$. The sparsity patterns of $Z$ and $Z^T C_s Z$ for $\theta = 0.15$, 0.5 and 1 are given in Figures 4.2 and 4.3. The effects of varying $\theta$ on the orthogonality of the null-space basis

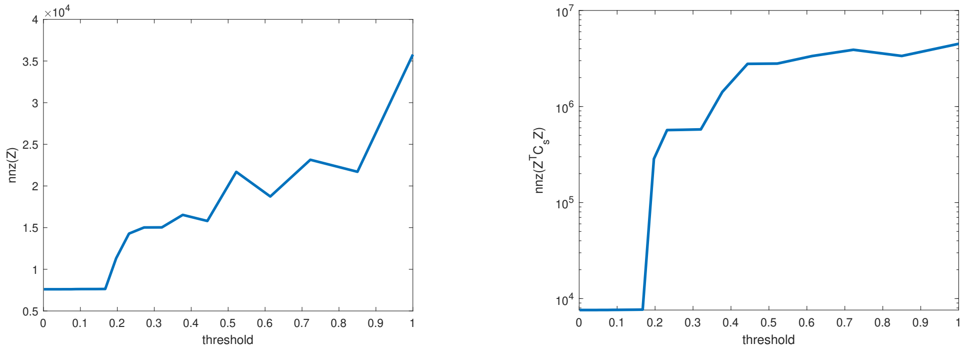

FIG. 4.1. *Dependence of the number of entries in $Z$ (left) and in the triangular part of $Z^T C_s Z$ (right) on the threshold parameter $\theta$ for the problem* `aircraft`. *$Z$ is computed using Algorithm 3.*

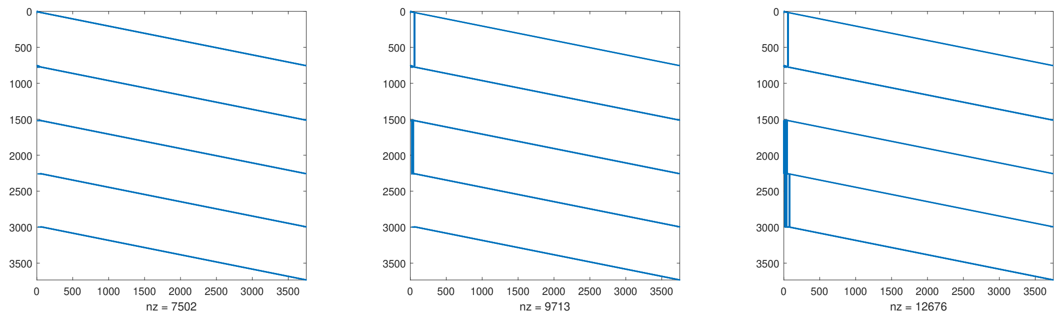

FIG. 4.2. *Sparsity pattern of $Z$ for $\theta = 0.15$ (left), $\theta = 0.5$ (centre), and $\theta = 1$ (right) for the problem* `aircraft`. *$Z$ is computed using Algorithm 3.*

(here and elsewhere this is measured as the norm of $BZ$) and on the condition number $cond(Z^T C_s Z)$ (computed using the MATLAB 1-norm condition number estimator `condest`) are illustrated in Figure 4.4. We see that, except for very small $\theta$, $\|BZ\|$ is small and there is little variation in $cond(Z^T C_s Z)$.

We also tested the fundamental null-space basis approaches discussed in Section 3.2.2. The standard approach based on the pivoted QR factorization (3.3) finds a well-conditioned submatrix and $nnz(Z^T C_s Z) \approx 7 \times 10^6$ for a range of values of $\theta$. For right oblique conjugation (Algorithm 4), $cond(Z^T C_s Z) = 2.8 \times 10^9$ (independently of $\theta$) and $nnz(Z^T C_s Z) \approx 1.4 \times 10^6$. The greater density

[1]Available at `http://www.hsl.rl.ac.uk`

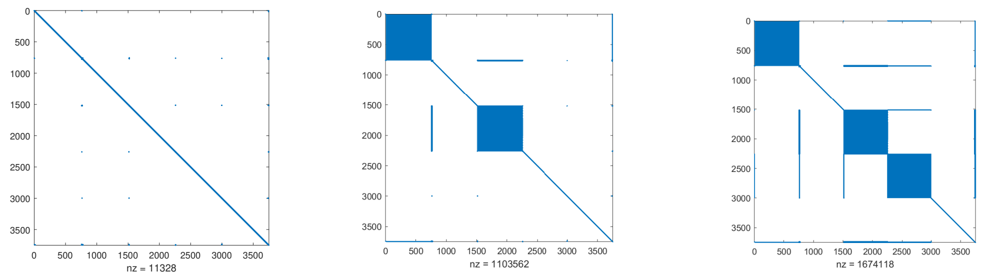

FIG. 4.3. *Sparsity pattern of $Z^TC_sZ$ for $\theta = 0.15$ (left), $\theta = 0.5$ (centre), and $\theta = 1$ (right) for the problem* `aircraft`*. The number of entries in $Z^TC_sZ$ is 11,328, 1,103,582 and 1,674,118, respectively. Z is computed using Algorithm 3.*

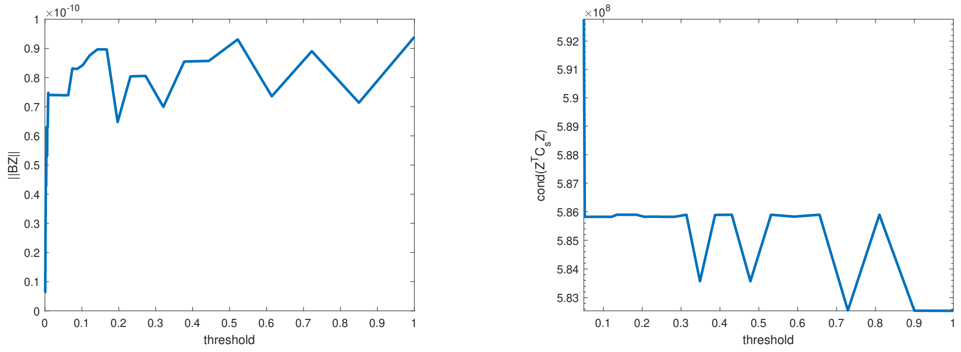

FIG. 4.4. *Dependence of orthogonality of the null-space basis (left) and on $cond(Z^TC_sZ)$ (right) on the threshold parameter $\theta$ for the problem* `aircraft`*. Z is computed using Algorithm 3.*

for the fundamental null-space approach is potentially an important disadvantage if it is used as here in combination with a sparse direct solver. However, we anticipate that the approach may be more attractive if $Z^TC_sZ$ is applied implicitly, such as in the employment of an iterative solver; we plan to investigate this further in the future.

In Figure 4.5, we plot $cond(Z^TZ)$ and the ratio $||A^Tres||/||res||$ for Algorithms 3, 4 and 6 for the threshold parameter $\theta \in [0.001, 1]$ (here $res = b - Ax$ is the least squares residual). We see that Algorithm 4 leads to the smallest condition numbers but the variation in ratio is much smaller.

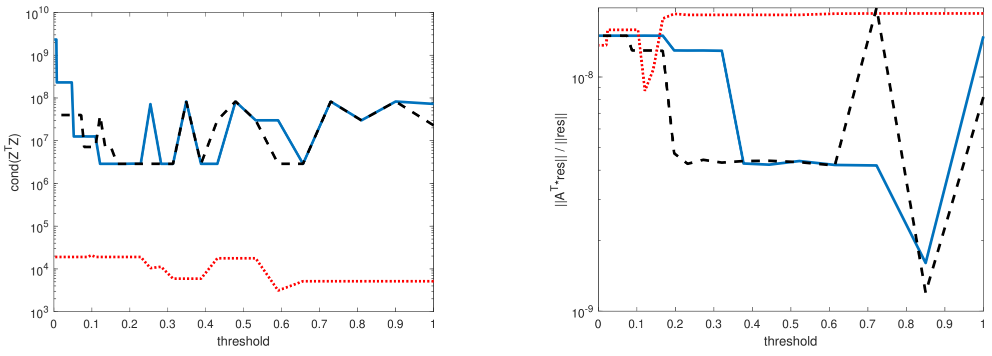

FIG. 4.5. *Dependence of the condition number of $Z^TZ$ (left) and the ratio $||A^Tres||/||res||$ (right) on the threshold parameter for the problem* `aircraft`*. Results for Algorithm 3 are represented by the blue solid line, Algorithm 4 by the red dotted line, and Algorithm 6 by the black dashed line.*

Finally, Figure 4.6 looks at varying the number of dense rows and plots the ratio $||A^Tres||/||res||$ for Algorithms 3 and 4 and the standard pivoted QR approach. The problem has 17 dense rows; here we report on $m_d$ dense rows with $1 \le m_d \le 17$ (we discard the remaining $17 - m_d$ rows and keep the sparse row block $A_s$ unchanged). We see that each of the approaches gives similar results, illustrating that pivoted QR can be replaced by one of the less expensive approaches.

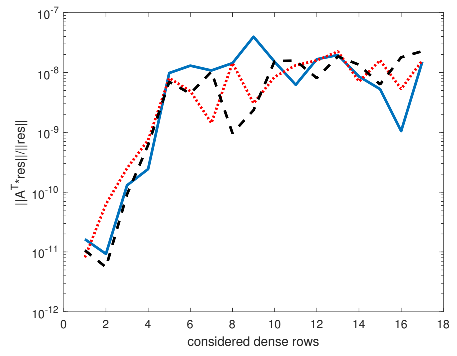

Fig. 4.6. *The effect on the ratio $||A^T res||/||res||$ of varying the number of dense rows problem* `aircraft`*. Results for Algorithm 3 are represented by the blue solid line, Algorithm 4 by the red dotted line, and pivoted QR by the black dashed line. The threshold parameter is $\theta =$.*

**4.2. Experiments on other matrices.** Table 4.1 presents results for other least squares test examples. In these experiments, the threshold parameter is $\theta = 0.25$. The emphasis here is on comparing the proposed approaches for computing a null-space basis for wide $B = A_d \in \mathbb{R}^{m_d \times n}$. In particular, we compare the standard pivoted QR computation of the fundamental null-space basis given by (3.3) with Algorithm 3 and the right conjugation approach of Algorithm 4. The condition number estimate $cond(Z^T C_s Z)$ of the transformed normal matrix is again computed using `condest`. The results demonstrate that, as expected, the different approaches lead to null-space bases with complementary properties, with no single approach being uniformly advantageous. In particular, we see that for the

Table 4.1
*Results for other least squares examples with the fixed threshold parameter $\theta = 0.25$. Here $m_d$ is the number of rows in the dense row block $A_d$, nnz and cond denote the number of entries and the condition number estimate for $Z^T C_s Z$, respectively. ‡ indicates insufficient memory for* `condest`*; † indicates insufficient memory to construct $Z^T C_s Z$.*

| | | | | Pivoted QR | | Algorithm 3 | | Algorithm 4 | |
|---|---|---|---|---|---|---|---|---|---|
| Identifier | $m$ | $n$ | $m_d$ | *nnz* | *cond* | *nnz* | *cond* | *nnz* | *cond* |
| `deter3` | 21,777 | 7,647 | 15 | $2.9 \times 10^7$ | $1.2 \times 10^3$ | $2.6 \times 10^4$ | $8.5 \times 10^3$ | $1.3 \times 10^5$ | $2.1 \times 10^3$ |
| `deter8` | 10,905 | 3,831 | 15 | $7.3 \times 10^6$ | $5.3 \times 10^2$ | $1.3 \times 10^4$ | $8.5 \times 10^3$ | $3.8 \times 10^4$ | $1.0 \times 10^3$ |
| `lp_agg` | 615 | 488 | 20 | $1.1 \times 10^5$ | $1.6 \times 10^5$ | $3.6 \times 10^4$ | $3.3 \times 10^9$ | $2.8 \times 10^4$ | $1.3 \times 10^5$ |
| `PDE1` | 270,595 | 271,792 | 1 | † | ‡ | $2.2 \times 10^6$ | $4.2 \times 10^2$ | $1.6 \times 10^{11}$ | ‡ |
| `sc205-2r` | 62,423 | 35,213 | 8 | $1.1 \times 10^7$ | ‡ | $1.3 \times 10^5$ | $8.4 \times 10^1$ | $1.1 \times 10^7$ | $6.8 \times 10^3$ |
| `sctap1-2b` | 33,858 | 15,390 | 34 | $1.1 \times 10^8$ | ‡ | $7.5 \times 10^6$ | $3.5 \times 10^4$ | $3.1 \times 10^6$ | $4.6 \times 10^6$ |

chosen threshold parameter, the approach of Algorithm 3 results in sparse transformed matrices but it can lead to a large condition number (as illustrated by problem `lp_agg`). Furthermore, comparing the pivoted QR and Algorithm 4, which both construct fundamental bases, we see that the condition number estimate is similar but the latter produces a sparser transformed normal matrix.

**4.3. Results for problem `hues_mod`.** The experiments presented so far targeted the saddle-point formulation of sparse-dense least squares problems. The next example, `hues_mod`, comes from a convex quadratic programming problem and is taken from the CUTEst test set [27]. While the $(1,1)$ Hessian block of size $n = 10000$ is well-conditioned, the $k = 2$ rows that form the off-diagonal constraint block $B$ are dense with entries that differ by 6 orders of magnitude. This is potentially challenging when constructing a sparse null space basis matrix $Z$. The $(2,2)$ block is $C = 10^{-6} * I_2$. In Figure 4.7, we report results for Algorithm 3 for different values of the threshold pivoting parameter $\theta$. We see that, even for small thresholds ($\theta \approx 0.1$), the orthogonality of $Z$ is very good and both the number of entries $nnz(Z^T HZ)$ in the transformed $(1,1)$ block and its factor increase steadily with $\theta$. Note that, for this problem, there is little fill in the factor of $Z^T HZ$.

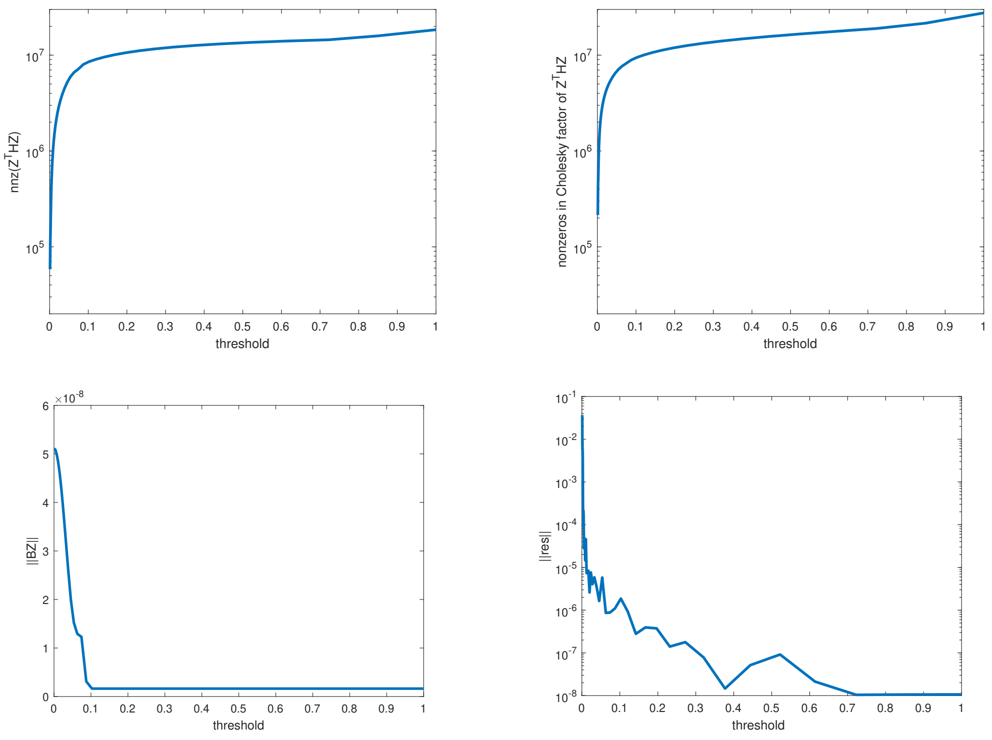

FIG. 4.7. *Dependence of the number of entries in the transformed* $(1,1)$ *block (top left) and in its Cholesky factor (top right), the orthogonality of* $Z$ *measured as* $||BZ||$ *(bottom left), and the residual norm (bottom right) on the threshold parameter for the problem* `hues_mod`*.* $Z$ *is computed using Algorithm 3.*

In Figure 4.8, we illustrate how the solution time is influenced by the threshold parameter. This is the total solution time (in seconds).[2] We see that using a large threshold is expensive.

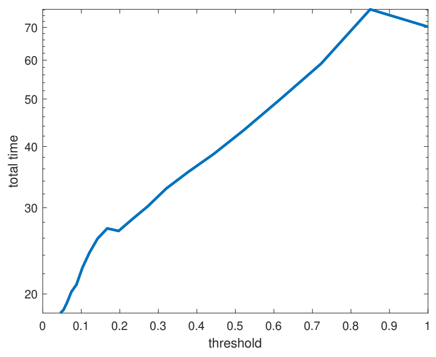

FIG. 4.8. *Dependence of the solution time in seconds on the threshold parameter for the problem* `hues_mod`*.* $Z$ *is computed using Algorithm 3.*

**5. Concluding remarks and future directions.** In this paper, we have proposed a new null-space approach for solving general symmetric saddle point systems with a small and non zero $(2,2)$ block and a $(2,1)$ block that may be rank deficient. An important motivation was solving large-scale linear least squares problems in which the system matrix has a small number of rows that are considered to be dense. Because the success of null-space approaches depends on being able to construct appropriate null-space bases, we have looked at how this can be done stably for our applications. In particular, our emphasis has been on null-space bases for $k \times n$ matrices that are wide ($k \ll n$) and possibly dense.

[2]Timings on a uniprocessor Intel Core(TM) i5-4990, 3.30GHz, 12GB memory.

The standard QR-based fundamental null-space basis computation leads to a transformed matrix that is relatively dense but has the advantage of being generally well conditioned. If a sparse direct solver is employed, the blocks of the transformed matrix must be constructed explicitly and the factors will further fill in. In this case, the QR approach is not ideal; indeed, memory requirements limit the size of systems that can be tackled. Null-space bases computed using right conjugation are also well-conditioned and offer the possibility of sparser transformed matrices.

Fundamental null-space bases are potentially attractive for iterative solvers if the basis can be efficiently applied implicitly and provided an effective preconditioner is available. In the future, we plan to develop preconditioners for use with an iterative solver for the solution of large-scale saddle-point systems with a small non zero $(2,2)$ block via our proposed null-space transformation. Possible lines of research are the left inverses proposed by Nash and Sofer [47] and the factorization behind the conjugation process. Preconditioning of the transformed system based on constructing $Z$ using Algorithm 3 with a small threshold parameter may be more straightforward but possible ill-conditioning must be taken into account.

A further goal will be to satisfy linear constraints with a small residual, that is, using the least squares notation of Section 2.3, if we require $A_d x = b_d$ then we need to ensure $res_d = b_d - A_d x$ is small. Applying Algorithm 3 to the test example `aircraft` gives $||res_d||_\infty / ||res||_\infty \approx 2.2 \times 10^{-2}$, with little variation for different values of the threshold parameter $\theta$. Thus the constraints are not tightly satisfied. In the future, we will explore how we can use the null-space approach presented here in combination with other techniques to reduce $||res_d||_\infty$.

## REFERENCES

[1] R. Amit, C. A. Hall, and T. A. Porsching. An application of network theory to the solution of implicit Navier-Stokes difference equations. *J. of Computational Physics*, 40(1):183–201, 1981.

[2] M. Arioli and G. Manzini. Null space algorithm and spanning trees in solving Darcy's equation. *BIT Numerical Mathematics*, 43(suppl.):839–848, 2003.

[3] M. Arioli and G. Manzini. A network programming approach in solving Darcy's equations by mixed finite-element methods. *Electronic Transactions on Numerical Analysis*, 22:41–70, 2006.

[4] M. Arioli, J. Maryška, M. Rozložník, and M. Tůma. Dual variable methods for mixed-hybrid finite element approximation of the potential fluid flow problem in porous media. *Electronic Transactions on Numerical Analysis*, 22:17–40, 2006.

[5] M. Benzi. *A Direct Row-Projection Method For Sparse Linear Systems*. PhD thesis, Department of Mathematics, 1993.

[6] M. Benzi and G. H. Golub. A preconditioner for generalized saddle point problems. *SIAM J. on Matrix Analysis and Applications*, 26(1):20–41, 2004.

[7] M. Benzi, G.H. Golub, and J. Liesen. Numerical solution of saddle point problems. *Acta Numerica*, 14:1–137, 2005.

[8] M. Benzi and C. D. Meyer. A direct projection method for sparse linear systems. *SIAM J. on Scientific Computing*, 16(5):1159–1176, 1995.

[9] M. Benzi, C. D. Meyer, and M. Tůma. A sparse approximate inverse preconditioner for the conjugate gradient method. *SIAM J. on Scientific Computing*, 17(5):1135–1149, 1996.

[10] M. Benzi and M. Tůma. A sparse approximate inverse preconditioner for nonsymmetric linear systems. *SIAM J. on Scientific Computing*, 19(3):968–994, 1998.

[11] M. Berry and R. Plemmons. Computing a banded basis of the null space on the Denelcor HEP multiprocessor. *Contemporary Mathematics*, 47:7–23, 1985.

[12] M. W. Berry, M. T. Heath, I. Kaneko, M. Lawo, R. J. Plemmons, and R. C. Ward. An algorithm to compute a sparse basis of the null space. *Numerische Mathematik*, 47(4):483–504, 1985.

[13] A. C. Cassell, J. C. de C. Henderson, and A. Kaveh. Cycle basis for flexibility analysis of structures. *International J. of Numerical Methods in Engineering*, 8:521–528, 01 1974.

[14] E. Chow, T. Manteuffel, C. Tong, and B. Wallin. Algebraic elimination of slide surface constraints in implicit structural analysis. *International J. of Numerical Methods in Engineering*, 57:1129–1144, 2003.

[15] M. T. Chu, R. E. Funderlic, and G. H. Golub. A rank-one reduction formula and its applications to matrix factorizations. *SIAM Review*, 37:512–530, 1995.

[16] T. F. Coleman and A. Pothen. The null space problem. I. Complexity. *SIAM J. on Algebraic and Discrete Methods*, 7(4):527–537, 1986.

[17] T. F. Coleman and A. Pothen. The null space problem. II. Algorithms. *SIAM J. on Algebraic and Discrete Methods*, 8(4):544–563, 1987.

[18] T. Dang, K. Ling, and J. Maciejowski. Banded null basis and ADMM for embedded MPC. *IFAC-PapersOnLine*, 50:13170–13175, 2017.

[19] T. A. Davis and Y. Hu. The University of Florida sparse matrix collection. *ACM Transactions on Mathematical Software*, 38(1):1–28, 2011.

[20] J. C. de C. Henderson and E. A. W. Maunder. A problem in applied topology: On the selection of cycles for the flexibility analysis of skeletal structures. *J. of the Institute of Mathematics and its Applications*, 5:254–269, 1969.

[21] N. Deo, G. M. Prabhu, and M. S. Krishnamoorthy. Algorithms for generating fundamental cycles in a graph. *ACM Transactions on Mathematical Software*, 8(1):26–42, 1982.

[22] R. Fletcher and T. Johnson. On the stability of null-space methods for KKT systems. *SIAM J. on Matrix Analysis and Applications*, 18(4):938–958, 1997.

[23] A. George and M. T. Heath. Solution of sparse linear least squares problems using Givens rotations. *Linear Algebra and its Applications*, 34:69–83, 1980.

[24] J. R. Gilbert and M. T. Heath. Computing a sparse basis for the null space. *SIAM J. on Algebraic and Discrete Methods*, 8(3):446–459, 1987.

[25] G. H. Golub and C. F. Van Loan. *Matrix Computations. 4th edition.* The Johns Hopkins University Press, Baltimore and London, 1996.

[26] C. Gotsman and S. Toledo. On the computation of null spaces of sparse rectangular matrices. *SIAM J. on Matrix Analysis and Applications*, 30(2):445–463, 2008.

[27] N. I. M. Gould, D. Orban, and Ph. L. Toint. CUTEst: a constrained and unconstrained testing environment with safe threads for mathematical optimization. *Computational Optimization and Applications*, 60:545–557, 2015.

[28] W. Govaerts. Solution of bordered singular systems in numerical continuation and bifurcation. In *Proceedings of the Fifth International Congress on Computational and Applied Mathematics (Leuven, 1992)*, volume 50, pages 339–347, 1994.

[29] C. A. Hall. Numerical solution of Navier-Stokes problems by the dual variable method. *SIAM J. on Algebraic and Discrete Methods*, 6(2):220–236, 1985.

[30] M. T. Heath, R. J. Plemmons, and R. C. Ward. Sparse orthogonal schemes for structural optimization using the force method. *SIAM J. on Scientific and Statistical Computing*, 5(3):514–532, 1984.

[31] M. R. Hestenes. Inversion of matrices by biorthogonalization and related results. *Journal of the Society for Industrial and Applied Mathematics*, 6:51–90, 1958.

[32] J. D. Hogg, J. K. Reid, and J. A. Scott. Design of a multicore sparse Cholesky factorization using DAGs. *SIAM J. on Scientific Computing*, 32:3627–3649, 2010.

[33] R. A. Horn and C. R. Johnson. *Matrix analysis.* Cambridge University Press, 1985.

[34] J. D. Horton. A polynomial-time algorithm to find the shortest cycle basis of a graph. *SIAM J. on Computing*, 16(2):358–366, 1987.

[35] J. S. Howell. Prestructuring sparse matrices with dense rows and columns via null space methods. *Numerical Linear Algebra with Applications*, 25:1–30, 2018. DOI:10.1002/nla.2133.

[36] D. James. Implicit nullspace iterative methods for constrained least squares problems. *SIAM J. on Matrix Analysis and Applications*, 13(3):962–978, 1992.

[37] D. James and R. J. Plemmons. An iterative substructuring algorithm for equilibrium equations. *Numerische Mathematik*, 57(6-7):625–633, 1990.

[38] L. Kaneko, M. Lawo, and G. Thierauf. On computational procedures for the force method. *International J. of Numerical Methods in Engineering*, 18(10):1469–1495, 1982.

[39] A. Kaveh. *Computational Structural Analysis and Finite Element Methods.* Springer, 2014.

[40] A. Kaveh. Graph transformations for efficient structural analysis. *Acta Mechanica*, 229(2):659–675, 2018.

[41] J. Kopal, M. Rozložník, A. Smoktunowicz, and M. Tůma. Rounding error analysis of orthogonalization with a non-standard inner product. *BIT Numerical Mathematics*, 52:1035–1058, 2012.

[42] S. Le Borne. Block computation and representation of a sparse nullspace basis of a rectangular matrix. *Linear Algebra and its Applications*, 428(11-12):2455–2467, 2008.

[43] S. Le Borne. Preconditioned nullspace method for the two-dimensional Oseen problem. *SIAM J. on Scientific Computing*, 31(4):2494–2509, 2009.

[44] J. Li and O. B. Widlund. FETI-DP, BDDC, and block Cholesky methods. *International J. of Numerical Methods in Engineering*, 66(2):250–271, 2006.

[45] E. A. W. Maunder. *Topological and linear analysis of skeletal structures.* PhD thesis, Imperial College, London, 1971.

[46] C. Meszaros. Detecting dense columns in interior point methods for linear programs. *Computational Optimization and Applications*, 36:309–320, 2007.

[47] S. G. Nash and A. Sofer. Preconditioning reduced matrices. *SIAM J. on Matrix Analysis and Applications*, 17(1):47–68, 1996.

[48] A. Pinar, E. Chow, and A. Pothen. Combinatorial algorithms for computing column space bases that have sparse inverses. *Electronic Transactions on Numerical Analysis*, 22:122–145, 2006.

[49] R. J. Plemmons and R. E. White. Substructuring methods for computing the nullspace of equilibrium matrices. *SIAM J. on Matrix Analysis and Applications*, 11(1):1–22, 1990.

[50] A. Pothen. *Sparse Null Bases and Marriage Theorems.* PhD thesis, Cornell University, Ithaca, NY, USA, 1984. AAI8415425.

[51] A. Pothen. Sparse null basis computations in structural optimization. *Numerische Mathematik*, 55(5):501–519, 1989.

[52] T. Rees and J. A. Scott. A comparative study of null-space factorizations for sparse saddle point systems. *Numerical Linear Algebra with Applications*, 25:e2103, 2018. DOI: 10.1002/nla.2103.

[53] J. A. Scott and M. Tůma. Solving mixed sparse-dense linear least-squares problems by preconditioned iterative methods. *SIAM J. on Scientific Computing*, 39(6):A2422–A2437, 2017.

[54] J. A. Scott and M. Tůma. A Schur complement approach to preconditioning sparse least-squares problems with some dense rows. *Numerical Algorithms*, 79:1147–1168, 2018. DOI: 10.1007/s11075-018-0478-2.

[55] J. A. Scott and M. Tůma. Sparse stretching for solving sparse-dense linear least-squares problems. *SIAM J. on Scientific Computing*, 41(3):A1604–1625, 2019.

[56] J. A. Scott and M. Tůma. Strengths and limitations of stretching for least-squares problems with some dense rows. *ACM Transactions on Mathematical Software*, 2020. To appear.

[57] G. Shklarski and S. Toledo. Computing the null space of finite element problems. *Computer Methods in Applied Mechanics and Engineering*, 198(37-40):3084–3095, 2009.

[58] E. Soyer and A. Topçu. Sparse self-stress matrices for the finite element force method. *International Journal for Numerical Methods in Engineering*, 50:2175 – 2194, 03 2001.

[59] J. M. Stern and S. A. Vavasis. Nested dissection for sparse nullspace bases. *SIAM J. on Matrix Analysis and Applications*, 14(3):766–775, 1993.

[60] M. Tůma. Implicit Gauss algorithm for solving the sparse unsymmetric sets of linear equations. Technical Report CSGS 1/85, Department of Mathematics, Statistics and Informatics, University of Bergamo, 1992.

[61] A Topçu. *A contribution to the systematic analysis of finite element structures using the force method.* PhD thesis, University of Essen, Federal Republic of Germany, 1979.

[62] P. Wolfe. Methods of nonlinear programming. In *Nonlinear Programming (NATO Summer School, Menton, 1964)*, pages 97–131. North-Holland, Amsterdam, 1967.