Science and Technology Facilities Council

Preprint
RAL-P-2020-003

A null–space approach for symmetric saddle point systems with a non zero (2,2) block

J Scott, M Tuma

July 2020

Submitted for publication in SIAM Journal on Scientific Computing

Enquiries concerning this report should be addressed to:

RAL Library
STFC Rutherford Appleton Laboratory
Harwell Oxford
Didcot
OX11 0QX

Tel: +44(0)1235 445384
Fax: +44(0)1235 446677
email: libraryral@stfc.ac.uk

Science and Technology Facilities Council reports are available online at:
https://epubs.stfc.ac.uk

ISSN 1361-4762

A NULL-SPACE APPROACH FOR SYMMETRIC SADDLE POINT SYSTEMS WITH A NON ZERO (2,2) BLOCK

JENNIFER SCOTT* AND MIROSLAV TŮMA†

Abstract. Null-space methods have long been used to solve large-scale symmetric saddle point systems of equations in which the $k \times k$ $(2,2)$ block is zero. This paper focuses on the case where the $(2,2)$ block is non zero. A novel null-space approach is proposed to transform the saddle point system into another symmetric saddle point system of the same order but with a zero $(2,2)$ block of order at most $2k$. Success of any null-space approach is dependent on the construction of a suitable null-space basis. The not uncommon case of the off-diagonal block being a wide matrix that has far fewer rows than columns and that may be dense is considered. A number of approaches are explored with the aim of balancing stability of the transformed system with sparsity. Linear least squares problems that contain a small number of dense rows arising from practical applications 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 lies 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, $B \in \mathbb{R}^{k\times n}$ $(n > k)$ has full rank and $C \in \mathbb{R}^{k\times k}$ is symmetric positive semidefinite. 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 frequently 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 also the excellent paper of Benzi, Golub and Liesen [6], which describes a wide range of real life problems that are dependent on the solution of saddle-point systems). Large-scale least squares problems are of particular interest to us and were one of the motivations behind our proposed null-space approach.

There has been considerable work over many years and in different research areas into null-space methods for saddle point problems. 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 for the matrix Z, whose columns form a basis for $\mathcal{N}(B)$, to have additional properties, such as bandedness, sparsity or orthogonality. Moreover, the classical null-space method is restricted to the case that the $(2,2)$ block is zero $(C = 0)$.

The first objective of this paper is to present a null-space approach for symmetric saddle point systems in which the $(2,2)$ block is non zero. The new approach will preserve symmetry and result in a transformed saddle point system of order $n+k$ with a zero $(2,2)$ of size $k+r$, where $r \leq k$ is the rank of B. The second objective is to propose a number of techniques for constructing null-space bases for B when k is small, exploring balancing sparsity in the transformed problem with stability through the use of threshold pivoting. Numerical examples taken from practical applications will be used to demonstrate the effectiveness of the approaches and illustrate their potential strengths and limitations.

* 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.

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 ith 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 ith 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 $b_1^T, \ldots, b_n^T$. $\mathcal{N}(B)$ 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 problems.

2.1. Null-space approach for $C = 0$. Saddle point systems with $C = 0$ frequently arise in practical applications and, in this case, a null-space method is one possible approach for solving (1.1) (see, for example, [6, Section 6]). Such methods are important in the field of numerical optimization [62], (where they are known as reduced Hessian methods). Here the underlying problem can be expressed as

$$\begin{aligned} \text{minimize} \quad & f(u) \\ \text{subject to} \quad & B\,u = g, \end{aligned} \tag{2.1}$$

with $u \in \mathbb{R}^n$ and $B \in \mathbb{R}^{k \times n}$ $(k < n)$. We assume that the constraint matrix B has full row rank. Computing u as a minimizer of the positive definite quadratic form that corresponds to this optimization problem leads to computing a particular solution $\hat{u}$ for the equilibrium equation (2.1), i.e., $\hat{u}$ satisfies

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

solving the saddle point system

$$\begin{pmatrix} H & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \bar{u} \\ v \end{pmatrix} = \begin{pmatrix} f - H\hat{u} \\ g \end{pmatrix}, \tag{2.2}$$

and then setting $u = \bar{u} + \hat{u}$. The null-space method for solving (2.2) assumes we have a matrix $Z \in \mathbb{R}^{n \times (n-k)}$ whose columns form a basis for $\mathcal{N}(B)$ i.e., $BZ = 0$. The second equation in (2.2) is equivalent to finding a vector $z \in \mathbb{R}^{n-k}$ such that $\bar{u} = Zz$. Substituting this into the first equation gives

$$\begin{aligned} & HZz + B^T v = f - H\hat{u} \\ \iff \quad & Z^T H Z z = Z^T (f - H\hat{u}) \end{aligned} \tag{2.3}$$

Therefore, by solving the reduced system (2.3), it is possible to straightforwardly recover $u = \hat{u} + Zz$; finally, v can be obtained by solving the symmetric positive definite system

$$BB^T v = B(f - Hu).$$

Thus solving the saddle point problem is reduced to that of solving two smaller systems of size $k \times k$ and $(n-k) \times (n-k)$ that have nice properties. In particular, if H is symmetric and positive definite, then $Z^T H Z$ stays symmetric and positive definite and efficient solvers can be used to solve (2.3). The algorithm to solve (2.2) is summarized as Algorithm 1.

A recent study of null-space factorizations for saddle point systems with $C = 0$ has been given by Rees and Scott [54] and the advantages and disadvantages of null-space methods are summarized in [46]. They can be very efficient for solving a series of problems with the same block B but different H because the null-space basis Z 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 [27]. Another motivation is when the constraints (2.1)

Algorithm 1 Dual variable method for solving (2.2)

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 $Z^T HZz = Z^T(f - H\hat{u})$.
4: Set $x = \hat{u} + Zz$.
5: Find $y \in \mathbb{R}^k$ such that $B^T y = f - Hu$.

need be satisfied accurately. Algorithms for the so-called *Linear Equality Problem* (LEP) are extremely important [20, 25, 62]. As already observed, a potential difficulty with the null-space method is the need to construct a null-space basis Z and, even when Z can be computed efficiently, some columns of Z many contain a large number of entries so that $Z^T HZ$ may not be sparse.

The null-space method can be generalised to unsymmetric saddle point systems in which H and/or C are unsymmetric and the $(1,2)$ 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.

2.2. Symmetry-preserving null-space approach for $C \neq 0$. As Golub et al. observe [6, Section 6], the null-space method cannot be applied to solve saddle point systems with $C \neq 0$. In a recent paper, Howell [37] proposes what he terms a one-sided application of the null-space method to solve non-singular generalised saddle point systems of the form

$$\mathcal{A}\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} H & B_1^T \\ B_2 & -C \end{pmatrix}\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}, \tag{2.4}$$

in which H and $C \neq 0$ are possibly unsymmetric and either $B_1 \in \mathbb{R}^{k\times n}$ and/or $B_2 \in \mathbb{R}^{k\times n}$ is dense. Howell introduces an auxillary variable w that is used to replace the $(n+k)\times(n+k)$ system (2.4) by a larger $(n+2k)\times(n+2k)$ system

$$\hat{\mathcal{A}}\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} H & B_1^T & 0_{n,k} \\ B_2 & -C & 0_{k,k} \\ 0_{k,n} & 0_{k,k} & I_k \end{pmatrix}\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} f \\ g \\ 0_{k,1} \end{pmatrix}. \tag{2.5}$$

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

$$\begin{pmatrix} \hat{H} & \hat{B}_1^T \\ \hat{B}_2 & 0_{k,k} \end{pmatrix}\begin{pmatrix} \hat{u} \\ w \end{pmatrix} = \begin{pmatrix} \hat{f} \\ g \end{pmatrix}, \tag{2.6}$$

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 (2.6) 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 (2.5); 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 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 an unsymmetric one.

Our aim is to develop a null-space approach for (1.1) with $C \neq 0$ that retains symmetry, has a dimension only $n+k$ and has favourable numerical properties. The following theorem presents the necessary transformation. Note that full row rank of B is not assumed.

THEOREM 2.1. *Consider the symmetric saddle point problem (1.1) of order $n+k$ with $rank(B) = r \leq k$, C positive definite and H positive definite on the null space of B. Its solution can be obtained by solving a transformed saddle point problem of order $n+k$ with a symmetric positive definite principal leading submatrix of order $n-r$.*

Proof. Since H is positive definite on the null space of B, invertibility of the system matrix follows (for example, Theorem 3.1 in [6]). Let $E = \begin{pmatrix} Z & Y \end{pmatrix} \in \mathbb{R}^{n\times n}$ with $Z \in \mathbb{R}^{n\times(n-r)}$ having full column rank and such that

$$BE = \begin{pmatrix} 0_{k,n-r} & B_Y \end{pmatrix}, \tag{2.7}$$

where $B_Y = BY \in \mathbb{R}^{k\times r}$ is of full column rank. Set $\mathcal{E}$ to be the $(n+k)\times(n+k)$ matrix

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

This matrix is symmetric and non singular and thus so is the transformed saddle point matrix

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

Rewriting this transformation as

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

where $\hat{H} = (E^T H E)_{1:n-r,1:n-r} = Z^T H Z$ is the positive definite leading principal submatrix of $E^T H E$ of order $n-r$, and

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

The transformed system then becomes

$$\tilde{\mathcal{A}} \begin{pmatrix} \tilde{u} \\ \tilde{v} \end{pmatrix} = \begin{pmatrix} \hat{H} & \hat{B}^T \\ \hat{B} & -\hat{C} \end{pmatrix} \begin{pmatrix} \tilde{u} \\ \tilde{v} \end{pmatrix} = \begin{pmatrix} \tilde{f} \\ \tilde{g} \end{pmatrix} = \mathcal{E}^T \begin{pmatrix} f \\ g \end{pmatrix}, \tag{2.9}$$

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 (2.9), 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

$$u = \begin{bmatrix} E\tilde{u} \\ (E\tilde{v})_{1:r} \end{bmatrix}, \qquad v = (\tilde{v})_{r+1:2k}.$$

□

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

A sparse symmetric indefinite solver can be used to solve the transformed system (2.9). Alternatively, a sparse direct solver can be used to compute a Cholesky factorization of the positive definite $(1,1)$ block i.e., $\hat{H} = L_1 L_1^T$. This leads to the following block factorization

$$\begin{pmatrix} \hat{H} & \hat{B}^T \\ \hat{B} & -\hat{C} \end{pmatrix} = \begin{pmatrix} L_1 & \\ L_2 & L_S \end{pmatrix} \begin{pmatrix} I & \\ & -D_S \end{pmatrix} \begin{pmatrix} L_1^T & L_2^T \\ & L_S^T \end{pmatrix}, \tag{2.10}$$

where

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

and

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

where L_S is unit lower triangular and D_S is block diagonal with 1×1 and 2×2 blocks.

2.3. Null-space approach for 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 and positive definite. Our interest lies in the case where A is large and mainly sparse and includes a relatively small number of rows that 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, [56] 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

$$Cx = (C_s + A_d^T A_d)x = c, \qquad c = A_s^T b_s + A_d^T b_d, \tag{2.11}$$

where $C_s = A_s^T A_s$ is the *reduced normal matrix*. The solution of (2.11) can be obtained from the equivalent $(n + m_d) \times (n + m_d)$ system

$$\begin{pmatrix} C_s & A_d^T \\ A_d & -I \end{pmatrix} \begin{pmatrix} x \\ A_d x \end{pmatrix} = \begin{pmatrix} c \\ 0 \end{pmatrix}. \tag{2.12}$$

This is of the form (1.1) with $k = m_d$, $H = C_s$, $B = A_d$ and $C = I$. Provided A_s has full column rank, C_s is symmetric positive definite. However, in practice the sparse row block 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 [56, 57], 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 to make the rows of A_d sparser [55, 58]. This has the advantage that, provided A is of full rank, the issue of null columns does not arise. However, the dimensions of the stretched system can grow rapidly with m_d (particularly when A_s is very sparse) so that, even for a small number of dense rows, 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 (2.12), the assumptions of Theorem 2.1 are satisfied, with $r \leq m_d$ the rank of A_d.

LEMMA 2.2. *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 $\mathcal{N}(A_d)$ can be constructed, the null-space approach offers an alternative way of solving sparse-dense LS problems, even in the case that C_s is positive semidefinite.

3. Null-space basis construction. Solving large-scale saddle point systems (2.2) 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. The null-space basis Z for $B \in \mathbb{R}^{k \times n}$ of rank r can be sometimes obtained directly by analyzing the problem [16] 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 [64] (see also [41, 60]). A specific strategy of this kind, called the *turnback* method [41, 64], attracted interest in the late 1970s/early 1980s (see also an early parallel implementation [11] and its recent use in practice in [21]). 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 [32]; see also [12] for further refinements. More recently, factorization approaches for the computation of null-space bases of sparse matrices are discussed in [30, 59].

Theoretical and algorithmic breakthroughs based on bipartite graph matchings and matroid theory came with the 1984 thesis of Pothen [52] 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 the null-space basis Z having the smallest number of non zeros

possible (see also [18]). 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 should also be considered. Consequently, sophisticated proposals given in [19, 28, 52] 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 [23] (see also [15]). 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 [23, 48]. This construction was developed and discussed in the context of other approaches in [53]. 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, 31]. Many publications discuss using cycles in the graph: pointers to relevant literature can be found in [42, 43]. There are heuristic ways of finding the cycle basis, a task that is straightforward if the underlying graph is planar [31]. More generally, construction may be based on a spanning tree of the graph [52]. Procedures to find sparse cycle bases in this way are given in [24] (see also [36, 61]). However, discretizations of three-dimensional grids may not allow sparse cycle bases to be found [4]. As with the 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 [50]. Closely related proposals to compute Z while explicitly exploiting substructuring, motivated in part by parallel computing, were given in [40, 51] (see also [39]). The block form can be obtained algebraically, as in the nested dissection ordering discussed in [61]; see also [50].

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. As it can happen in practical applications, we allow for B being rank deficient ($r < k$). We want the nearly-square $Z \in \mathbb{R}^{n\times n-r}$ to be sparse and because we are going to apply a direct solver to factorization $Z^T HZ$, we want Z to have no dense rows. Orthogonal transformations inevitably lead to dense Z and so, although sometimes useful in other situations (see, for instance, [45]), are not appropriate here. Because B is wide, columns of Z can be computed independently and such that they have a small support (they can be always expressed as linear combinations of at most other k columns). The 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 with small support can grow like h^{-2}, where h is the discretization parameter [4]. Thus we propose employing an orthogonal factorization that incorporates pivoting for stability and, because B has only a small number of rows, this is not prohibitively expensive.

3.2.1. Banded null-space basis. This section discusses obtaining a permuted banded null-space basis based on expressing columns as a linear combination of some of the previous columns. These bases are similar to those from the category of triangular null-space bases [19] but QR with pivoting implies a more general sparsity pattern for Z. Our focus is on the numerical qualities of the bases rather than the minimality of the linear combinations.

We begin with a simple example in which B comprises a single fully dense row ($k = 1$). Algorithm 2 presents a backward-looking method based on the QR decomposition of B with pivoting. In this simple case, $Q = I_1$. The factor R is a single column with its first entry $\beta_1 \neq 0$; all the diagonal entries of $\tilde{Z}$ depend on this pivot. The computed Y is the range space basis of $(BP)^T$ [26, 54] satisfying

Algorithm 2 Construct a matrix $E = (Z \;\; Y) \in \mathbb{R}^{n\times n}$ such that $Z \in \mathbb{R}^{n\times n-1}$ is a null basis $\mathcal{N}(B)$ for a dense matrix $B \in \mathbb{R}^{1\times n}$.

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

$$BY = I_1.$$

Note that Z may not have a narrow band because of the permutation. Figure 3.1 illustrates Algorithm 2. A similar approach for B comprising a single dense row was recently proposed by Howell [37, Algorithm 3].

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

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

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

A dominant part of the saddle-point matrix (2.9) 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 [49]). Consequently, it is desirable for the condition number $cond(Z^TZ)$ to be small. The fact that constructing the basis matrix Z so that P^TZ has a narrow band may not be sufficient to guarantee this can be deduced from the following idealized example of B consisting of a single row vector of all ones.

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

Proof. P is the identity and the computed Z is bidiagonal with the diagonal entries equal to -1 and the remaining non zeros equal to 1. Z^TZ is a well-known Laplacian matrix with eigenvalues given by

$$2(1 - \frac{j\pi}{n+1}),\; 1 \leq j \leq n.$$

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

$$4\sin^2\frac{\pi}{2\,(n+1)} \approx \frac{\pi^2}{(n+1)^2}.$$

Hence the result. □

Howell [37] discusses the need to avoid division by small entries of B but his ad hoc strategy can still result in a highly ill-conditioned null-space basis: more sophisticated strategies are required. Algorithm 3 considers general $B \neq 0$ with $1 \leq k < n$ rows and rank $r \leq k$. The basic idea is to express columns of Z as

linear combinations of previous columns with close indices. The numerical rank of B is computed using a QR factorization with pivoting. The columns of B corresponding to the first r columns of R are permuted to the front. 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 the case of many sparse numerical linear algebra algorithms that need to combine locality with ensuring stability, this is achieved by using threshold column pivoting. This chooses pivot columns j that lie close to the r starting columns (that is, $j-r$ is small) and whose norm is at least θ times the maximum of the norms of the remaining columns, where $0 < \theta \leq 1$ is the pivoting threshold parameter.

Algorithm 3 Given a pivoting threshold $0 < \theta \leq 1$, construct a basis $Z \in \mathbb{R}^{n\times(n-r)}$ of $\mathcal{N}(B)$ for $B \in \mathbb{R}^{k\times n}$ with $rank(B) = r \leq k$.

1: Initialize $\tilde{Z} = 0_{n,n-r}$.
2: 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.
3: **for** $l = r+1, ..., n$
4: **if** $(\tilde{B})_{:,l}$ is zero column **then**
5: Set $(\tilde{Z})_{:,l-r} = e_{l-r}$
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 $Z = P\tilde{Z}, Y = P\begin{pmatrix} I_r & 0_{r,n-r} \end{pmatrix}^T$.

The matrix E computed by Algorithm 3 satisfies (2.7). Another possibility for Step 17 is to choose Y to be orthogonal by setting

$$Y = P\begin{pmatrix} R_1^{-1} & 0_{r,n-r} \end{pmatrix}^T,$$

so that the columns of Y are the first columns of the Q factor. Note that dense columns in Y do not necessarily imply as much fill-in in the saddle point matrix as results from dense rows in Z.

The pivoting in Step 2 of Algorithm 3 ensures Y is reasonably well-conditioned and guarantees that any dependent column $(\tilde{B})_{:,l}$ of $\tilde{B}$ can be expressed as a linear combination of previous columns. 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. P_L and P_T are local permutations. The first locally reverses the first $l-1$ columns of $\tilde{B}$ while the second safeguards the numerical stability of local expressions. Note also Step 10 is well defined because 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 a simple B with two rows and six columns; θ is set to 0.1. With this 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 Step 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.

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

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).*

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 [65] (see [19]). Let $B \in \mathbb{R}^{k\times n}$ 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

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

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

$$Z = P \begin{pmatrix} -R_1^{-1}R_2 \\ I_{n-r} \end{pmatrix} \tag{3.2}$$

or

$$Z = P_2 \begin{pmatrix} -U_1^{-1}U_2 \\ I_{n-r} \end{pmatrix}. \tag{3.3}$$

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

The biconjugation process of Hestenes [33] 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 [17] and some practical extensions were given by Benzi [5]; see also the direct projection method [7, 63]. Applying biconjugation approximately leads to so-called approximate inverse preconditioners [9]. We define a partial biconjugation factorization applied to $B \in \mathbb{R}^{k\times n}$, $rank(B) = r \le k$, to be a process that yields $V \in \mathbb{R}^{n\times n}$ and lower trapezoidal $L \in \mathbb{R}^{k\times n}$ satisfying

$$BV = L. \tag{3.4}$$

We term this the *right oblique conjugation*. Assuming r steps of an LU factorization of B can be performed without pivoting, Algorithm 4 computes a null-space basis matrix $Z \in \mathbb{R}^{k\times n}$ by right oblique conjugation. The key to this is that for $r = n$ the computed V is equal to U^{-1}, where U is the U factor of the

LDU factorization of B. We can see this as follows. V is upper triangular by construction because it is initialized to have unit vectors and then at the Step 4 of Algorithm 4, a linear combination that involves only previously computed vectors is subtracted from the current column of V. The uniqueness of the LDU factorization without pivoting in the square case together with (3.4) gives the result; see [8].

Algorithm 4 Right oblique conjugation. Given $B \in \mathbb{R}^{k\times n}$ with $rank(B) = r$ and rows $b_1^T, \ldots, b_n^T$, compute $V \in \mathbb{R}^{n\times n}$ such that BV is lower trapezoidal and the null-space basis matrix $Z \in \mathbb{R}^{(n-r)\times n}$.

1: Initialize $(v_1^{(0)}, \ldots, v_n^{(0)}) = I$
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 $Z = \left(v_{r+1}^{(r)}, \ldots, v_n^{(r)}\right)$.

Some early results [10, 14] show, for example, that if B is a banded square matrix with a fully dense band and it can be factorized without permutations then V is fully dense (and Z is empty). The following straightforward result provides some insight into why the right oblique conjugation is of interest here.

LEMMA 3.2. *Apply right oblique conjugation to $B \in \mathbb{R}^{k\times n}$ with $rank(B) = r$. Then there exist a permutation $P \in \mathbb{R}^{n\times n}$ and a block partitioning of $B = \begin{pmatrix} G & N \\ D & C \end{pmatrix}$ with $G \in \mathbb{R}^{r\times r}$ such that*

$$\begin{pmatrix} G & N \\ D & C \end{pmatrix} P \begin{pmatrix} Y & S \\ 0 & I \end{pmatrix} = \begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix} \tag{3.5}$$

In particular,

$$Z = P \begin{pmatrix} S \\ I \end{pmatrix}$$

is a 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 $Y = G^{-1}$. Moreover, $GS + N = 0$, from which it follows that

$$\begin{pmatrix} S \\ I \end{pmatrix} \equiv \begin{pmatrix} -G^{-1}N \\ I \end{pmatrix}$$

is the fundamental null-space basis. □

The first reason why fundamental null-space bases using right oblique conjugation are potentially of 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. Pivoting needs to be incorporated to maximize the magnitude of the quantity $b_i^T v_i^{(i-1)}$ used in Step 4. While the rows of B can be stored in static data structures, the columns of V eligible for pivoting (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 [47]. 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 [44]. In the case $k \ll n$, complete pivoting is not prohibitively expensive and may be needed for stability.

The second reason for considering 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 Step 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, [7]). 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 Step 4 the oblique projections*

$$z_j^{(i)} = \left(I - \frac{x_j b_i^T}{b_i^T x_j}\right) z_j^{(i-1)},\ j = i+1, \ldots, n, \tag{3.6}$$

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 variation of Theorem 6.4.1 of [29] (a slightly less general version was described in [37]).

THEOREM 3.4. *Consider the null-space basis* $Z_F \in \mathbb{R}^{n \times (n-r_1)}$ *of the matrix* $F \in \mathbb{R}^{k_1 \times n}$ *and the null-space basis* $Z_G \in \mathbb{R}^{(n-r_1)\times(n-r_1-r_2)}$ *of the matrix* $G \in \mathbb{R}^{k_2 \times n}$ *and* $GZ_F \in \mathbb{R}^{k_2 \times (n-r_1)}$. *Then the columns of* $Z_F Z_G$ *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 constraints. A special case of a procedure of this type was proposed by Howell [37]. To demonstrate the mechanism, we assume the rows of B are dense and $rank(B) = k$. Algorithm 5 generalises Algorithm 2 to $k \geq 1$. Its formulation directly uses the matrices E_l that contain both null-space and range-space bases vectors. The full row rank condition for B is assumed to simplify the exposition of the algorithm.

In the special case of no pivoting, $B_k \in \mathbb{R}^{k \times k}$ is lower triangular with non zero diagonal entries. Although we regard the rows of B as dense, in practice they may contain many zeros, which may fill in as Algorithm 5 proceeds. If B has a non-trivial block angular form, Algorithm 5 can take advantage of this to reduce the work needed. Three combinatorial approaches to find null-space bases for such matrices are given in [50]; here we build on our approach described above. Assume that the row blocks of B with full

Algorithm 5 Construct a null basis $\mathcal{N}(B)$ for $B \in \mathbb{R}^{k\times n}$, $rank(B) = k$, $k \geq 1$.

1: Set $J = \{1, \ldots, n\}$. Split B into τ row blocks given by

$$B = \left(B_{1,J}^T, \ldots, B_{\tau,J}^T\right)^T$$

such that $B_{i,J} \in \mathbb{R}^{k_i\times n}, i = 1, \ldots, \tau$, $\sum_i^\tau k_i = n$.

2: Apply Algorithm 2 to $B_{1,J}$ to construct $E_1 = \begin{pmatrix} Z_1 & Y_1 \end{pmatrix} \in \mathbb{R}^{n\times n}$ with the permutation matrix P_1.

3: Remove k_i column indices that correspond to Y_1 in the QR factorization with the column permutation P_1 from J. Those kept in J are termed eligible.

4: **for** $j = 2, ..., \tau$

5: Let $(BE)_{j,J} \in \mathbb{R}^{n-j+1}$ be the $|J|$ columns of the block row j that correspond to eligible indices of

$$BE_1 \prod_{l=2}^{j-1} \begin{pmatrix} E_l & \\ & I_{\lambda_l} \end{pmatrix}, \text{ where } \lambda_l = \sum_{i=1}^{l-1} n_l.$$

6: Construct $E_j = \begin{pmatrix} Z_j & Y_j \end{pmatrix} \in \mathbb{R}^{(n-\lambda_j+1)\times(n-\lambda_j+1)}$ by applying Algorithm 2 with the permutation matrix $P_j \in \mathbb{R}^{(n-\lambda_j+1)\times(n-\lambda_j+1)}$.

7: Update J by removing the column indices that correspond to Y_j in the QR factorization with the column permutation P_j from J.

8: **end for**

9: Set $E = \prod_{l=2}^{\tau} \begin{pmatrix} E_l & \\ & I_{\lambda_l} \end{pmatrix}$.

row rank can be ordered into the block angular form

$$B = \begin{pmatrix} \bar{B}_1 & & & \bar{D}_1 \\ & \bar{B}_2 & & \bar{D}_2 \\ & & \ddots & \vdots \\ & & \bar{B}_s & \bar{D}_s \end{pmatrix}, \tag{3.7}$$

where $\bar{B}_i \in \mathbb{R}^{k_i\times n_i}$ and $\bar{D}_i \in \mathbb{R}^{k_i\times n_d}$. Furthermore, assume that the null-space basis contains all $\sum_{i=1}^s (n_i - k_i)$ null space vectors of all $\bar{B}_i$ extended by zeros outside their domains. Then the matrix Z of null-space vectors is also of block angular form.

In our experiments, we use a special case of Algorithm 5 in which $\tau = k$. That is, each row block contains a single row. The approach is given in Algorithm 6; B is not assumed to have full row rank. It constructs the null-space basis of $B \in \mathbb{R}^{k\times n}$ as a product of bases 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 Step 4, $l_i = l - 1$ if $rank((\tilde{B})_{1:i,:}) > rank((\tilde{B})_{1:i-1,:})$ and $l_i = l$ otherwise.

Algorithm 6 Construct a null basis Z of $B \in \mathbb{R}^{k\times n}$ as a product of bases corresponding to rows of B.

1: Initialize $Z \in \mathbb{R}^{n\times n} = I$ 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 $\tilde{b}_i^T$.

5: Set $l = l_i$

6: Compute $Z = ZZ_i \in \mathbb{R}^{n\times l}$.

7: **end for**

To illustrate Algorithm 6, consider the 3×6 matrix

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

The matrices Z_i, $i = 1, 2, 3$, computed using Algorithm 6 are

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

Combined they give the 5×3 null-space basis matrix

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

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 non zero $(2, 2)$ block. Having computed a null space basis, in all our experiments, we use the sparse direct solver `HSL_MA97` [34, 35] from the HSL mathematical software library [38] to compute the required solution.

4.1. Results for problem `aircraft`. We first report detailed findings for problem `aircraft` from the SuiteSparse Matrix Collection [22] and then present results for other sparse-dense least squares problems used in our earlier study [56]. The example `aircraft` 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). In the least squares saddle point formulation (2.12), the row block $B = A_d \in \mathbb{R}^{m_d \times n}$ comprises the $m_d = 17$ rows that are identified as dense; 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 θ in Algorithm 3 on the sparsity of Z and the transformed normal matrix $Z^T C_s Z$. It confirms the significant advantage of using a small θ. The sparsity pattern of $Z^T C_s Z$ for $\theta = 1$ and 0.15 are given in Figure 4.2. We also tested the fundamental null-space basis approaches discussed in Section 3.2.2. The standard approach based on the pivoted QR factorization (3.2) finds a well-conditioned submatrix. In this case, for a range of values of the threshold parameter, $nnz(Z^T C_s Z) \approx 7 \times 10^6$, while for right conjugation (Algorithm 4), $nnz(Z^T C_s Z) \approx 1.4 \times 10^6$. The greater density for the fundamental null-space approach is an important disadvantage if it is used as here in combination with a direct solver. However, we anticipate that it may be more attractive if $Z^T C_s Z$ is applied implicitly, such as in the employment of an iterative solver; we plan to investigate this further in the future.

The effects of varying θ in Algorithm 3 on the orthogonality of the null-space basis 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 4.3. We see that using small θ does not adversely affect the orthogonality of the computed Z (measured as the norm of BZ) and, except for very small θ, there is little variation in $cond(Z^T C_s Z)$. For right conjugation, $cond(Z^T C_s Z) = 2.8 \times 10^9$ (independently of θ). The right-hand plot in Figure 4.4 shows the ratio $||A^T r||_2 / ||r||_2$ for Algorithms 3 and 6, where $r = b - Ax$ is the LS residual). We see that both approaches achieve similar results.

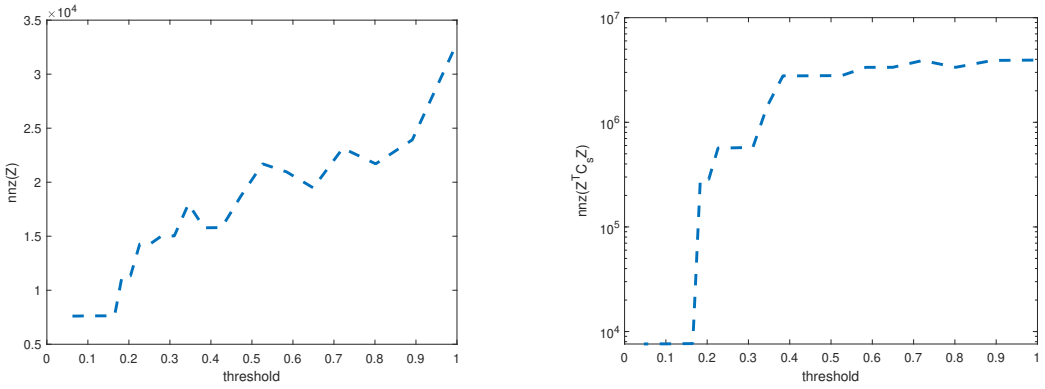

FIG. 4.1. *Dependence of the number of entries in Z (left) and in the triangular part of $Z^T C_s Z$ (right) on the threshold parameter θ for the problem* `aircraft`. *Z is computed using Algorithm 3.*

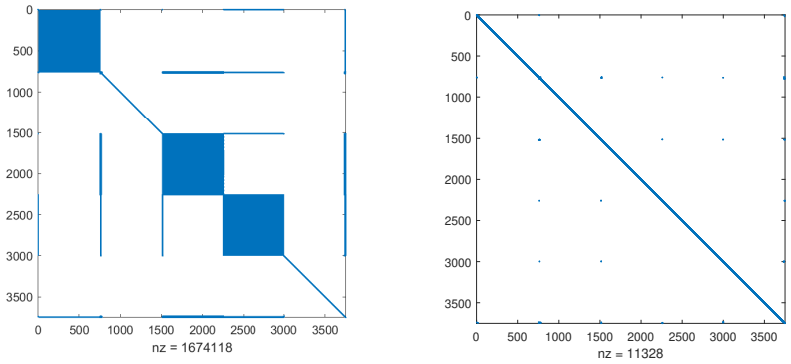

FIG. 4.2. *Pattern of $Z^T C_s Z$ for $\theta = 1$ (left) and $\theta = 0.15$ (right) for the problem* `aircraft`. *Z is computed using Algorithm 3.*

4.2. Experiments on other matrices. Table 4.1 presents results for other test examples in which A has a small number (m_d) of dense rows. 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 Algorithm 3, the standard QR-based computation of the fundamental null-space basis with column pivoting based on (3.2) and the right conjugation approach of Algorithm 4. The condition number estimate $cond(Z^T C_s Z)$ of the transformed matrix $Z^T C_s Z$ is again computed using `condest`. The reported 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,

TABLE 4.1

Results for other examples with the fixed threshold parameter $\theta = 0.25$. Here 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 $Z^T C_s Z$.*

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

we see that for the chosen threshold parameter, Algorithm 3 results in the sparsest transformed matrix.

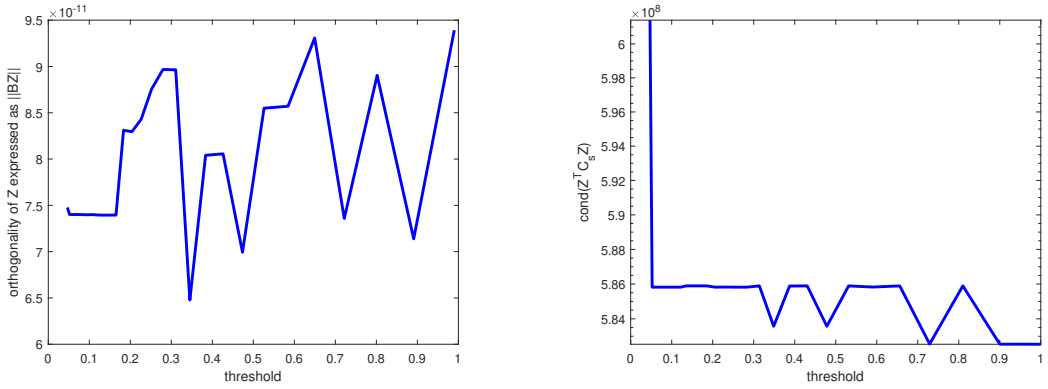

FIG. 4.3. *Dependence of orthogonality of the null-space basis (left) and on cond$(Z^T C_s Z)$ (right) on the threshold parameter θ for the problem* `aircraft`. *Z is computed using Algorithm 3.*

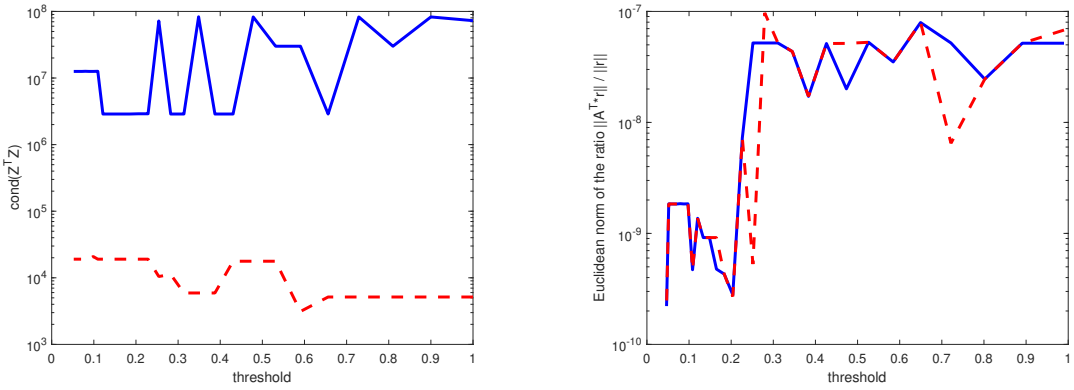

FIG. 4.4. *Dependence of the condition number of $Z^T Z$ (left) and the ratio $||A^T r||_2/||r||_2$ (right) on the threshold parameter for the problem* `aircraft`. *Here Algorithm 3 is the solid line; the dashed line is Algorithm 4 (left) and Algorithm 6 (right).*

Indeed, in this case, for each test example, Z is banded with a narrow band. However, Algorithm 3 can lead to a large condition number (for example, problem `lp_agg`).

5. Concluding remarks and future directions. In this paper, we have proposed a new null-space approach for general symmetric saddle point systems. An important motivation was solving least squares problems in which the system matrix has a small number of rows that are considered to be dense. Since success of the null-space approach is dependent on being able to construct an appropriate null-space basis, we have also looked at how this can be done stably for our applications of interest. In particular, our emphasis has been on $n \times k$ matrices that are wide ($k \ll n$) and possibly dense.

The standard QR-based fundamental null-space basis computation leads to a relatively dense transformed matrix but, as we have seen, it has the advantage that the resulting transformed matrix is generally well conditioned. For a direct solver, the transformed matrix must be constructed explicitly and its factors will further fill in. In this case, the QR approach is not ideal; indeed, the memory requirements limit the size of systems that a direct solver can tackle. 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 non zero $(2,2)$ block via our proposed null-space approach. Possible lines of research are the left inverses proposed by Nash and Sofer [49] and the factorization behind the right conjugation process. Preconditioning of the transformed system based on a banded Z from 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 achieve tighter satisfaction of the linear constraints, that is, using the LS notation of Section 2.3, to ensure the residual of the constraints $r_d = b_d - A_d x$ is small. For Algorithm 3 applied to the test example `aircraft` we found $||r_d||_\infty / ||r||_\infty \approx 2.2 \times 10^{-2}$, with little variation for different values of the threshold parameter θ. Thus the constraints are not very tightly satisfied; this is a consequence of employing a direct solver to solve the transformed saddle-point system (2.9). In the future, we plan to explore how to use the null-space approach presented here in combination with other techniques to reduce $||r_d||_\infty$. Two possible ideas will be explored: the standard formulation of the null space approach [13] applied to a regularized problem and the development of selective iterative refinement.

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

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

[8] 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.

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

[10] M. Benzi and M. Tůma. Orderings for factorized sparse approximate inverse preconditioners. *SIAM J. on Scientific Computing*, 21(5):1851–1868, 2000.

[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] Å. Björck. *Numerical Methods for Least Squares Problems*. SIAM, Philadelphia, 1996.

[14] R. Bridson and W.-P. Tang. Ordering, anisotropy, and factored sparse approximate inverses. *SIAM J. on Scientific Computing*, 21(3):867–882, 1999.

[15] 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.

[16] 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.

[17] 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.

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

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

[20] R. W. Cottle and M. N. Thapa. *Linear and nonlinear optimization*, volume 253 of *International Series in Operations Research & Management Science*. Springer, New York, 2017.

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

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

[23] 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.

[24] 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.

[25] M. Fathi and H. Bevrani. *Optimization in Electrical Engineering.* Springer, 2019.

[26] 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.

[27] 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.

[28] 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.

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

[30] 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.

[31] 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.

[32] 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.

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

[34] J. D. Hogg and J. A. Scott. HSL_MA97: a bit-compatible multifrontal code for sparse symmetric systems. Technical Report RAL-TR-2011-024, Rutherford Appleton Laboratory, 2011.

[35] J. D. Hogg and J. A. Scott. New parallel sparse direct solvers for multicore archiectures. *Algorithms*, 6:702–725, 2013.

[36] 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.

[37] 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.

[38] HSL. A collection of Fortran codes for large-scale scientific computation, 2016. `http://www.hsl.rl.ac.uk`.

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

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

[41] 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.

[42] A. Kaveh. *Computational structural analysis and finite element methods.* Springer, 2014.

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

[44] 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.

[45] 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.

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

[47] Jing Li and Olof B. Widlund. FETI-DP, BDDC, and block Cholesky methods. *Internat. J. Numer. Methods Engrg.*, 66(2):250–271, 2006.

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

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

[50] 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.

[51] 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.

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

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

[54] 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.

[55] J. Scott and M. Tůma. Strengths and limitations of stretching for least-squares problems with some dense rows. Technical Report RAL-P-2019-001, RAL, 2019.

[56] 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.

[57] 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.

[58] 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.

[59] 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.

[60] 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.

[61] 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.

[62] G. Strang. A framework for equilibrium equations. *SIAM Review*, 30(2):283–297, 1988.

[63] 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.

[64] 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.

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