Science & Technology Facilities Council

Technical Report
RAL-TR-2014-016

The null-space method and its relationship with matrix factorizations for sparse saddle point systems

T Rees, J Scott

November 2014

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 446403
email: libraryral@stfc.ac.uk

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

ISSN 1358-6254

THE NULL-SPACE METHOD AND ITS RELATIONSHIP WITH MATRIX FACTORIZATIONS FOR SPARSE SADDLE POINT SYSTEMS

TYRONE REES*, JENNIFER SCOTT*

Abstract. The null-space method for solving saddle point systems of equations has long been used to transform an indefinite system into a symmetric positive definite one of smaller dimension. A number of independent works in the literature have identified the equivalence of the null-space method and matrix factorizations. In this report, we review these findings, highlight links between them, and bring them into a unified framework. We also investigate the suitability of using null-space based factorizations to derive sparse direct methods, and present numerical results for both practical and academic problems. Finally, we explore some properties of an incomplete version of one of these factorizations as a preconditioner and provide eigenvalue bounds.

Key words. Null-space methods, saddle point systems, direct methods, preconditioning

1. Introduction. A saddle point system is an indefinite linear system of equations of the form

$$\underbrace{\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix}}_{\mathcal{A}} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix}, \tag{1.1}$$

where we will assume that $B \in \mathbb{R}^{m\times n}$ $(n > m)$ has full rank and $A \in \mathbb{R}^{n\times n}$ is symmetric positive definite on the null space of B. We are particularly interested in the case where A and B are large, sparse matrices and in much of our discussion we will focus on A symmetric and positive semi-definite.

One approach for solving (1.1) is the null-space method [3, Section 6]. Suppose we are given a matrix $Z \in \mathbb{R}^{n\times(n-m)}$ whose columns form a basis for the null-space of B, i.e., $BZ = 0$. Suppose additionally that we have a particular solution for the second equation, i.e., a vector $\widehat{\mathbf{x}}$ such that

$$B\widehat{\mathbf{x}} = \mathbf{g}.$$

Then solving (1.1) is equivalent to solving

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \bar{\mathbf{x}} \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{f} - A\widehat{\mathbf{x}} \\ \mathbf{0} \end{bmatrix},$$

where $\mathbf{x} = \widehat{\mathbf{x}} + \bar{\mathbf{x}}$. The second equation in this system is equivalent to finding a vector $\mathbf{z} \in \mathbb{R}^{(n-m)}$ such that $\bar{\mathbf{x}} = Z\mathbf{z}$. Substituting this into the first equation we have

$$\begin{aligned} & AZ\mathbf{z} + B^T\mathbf{y} = \mathbf{f} - A\widehat{\mathbf{x}} \\ \iff \quad & Z^TAZ\mathbf{z} = Z^T(\mathbf{f} - A\widehat{\mathbf{x}}) \end{aligned} \tag{1.2}$$

Therefore, by solving the reduced system (1.2), we can straight-forwardly recover $\mathbf{x} = \hat{\mathbf{x}} + Z\mathbf{z}$. We can then obtain $\mathbf{y}$ by solving the overdetermined system $A\mathbf{x} + B^T\mathbf{y} = \mathbf{f}$. This is the null-space method, which we summarise as Algorithm 1.

Null-space methods have been used in the fields of optimization (where they are known as reduced Hessian methods), structural mechanics (where they are known

*Rutherford Appleton Laboratory, Chilton, Didcot, UK, {tyrone.rees,jennifer.scott}@stfc.ac.uk

Algorithm 1 Null-space method for solving (1.1)

Choose Z so that its columns form a basis for the null space of B.
Find $\widehat{\mathbf{x}}$ such that $B\widehat{\mathbf{x}} = \mathbf{g}$.
Solve $Z^T AZ\mathbf{z} = Z^T(\mathbf{g} - A\widehat{\mathbf{x}})$
Set $\mathbf{x} = Z\mathbf{z} + \widehat{\mathbf{x}}$
Find $\mathbf{y}$ such that $B^T\mathbf{y} = \mathbf{f} - A\mathbf{x}$

as the 'force' method), fluid mechanics (where they are known as the 'dual variable' method) and electrical engineering (where they are known as 'loop analysis'). For a detailed overview and references, see Benzi, Golub and Liesen [3, Chapter 6]. Other methods for solving (1.1) include full space methods [7, 16, 32, 46] and reduced space Schur-complement based methods [3, Chapter 5]. However, null-space methods remain popular, particularly in the large-scale optimization literature [4, 5, 6, 31, 33, 40]. Such methods have been used in cases where the problem is perceived to be too large for a sparse indefinite solver to be effective, as the dimension of the problem to be factorized is reduced from $(n+m)\times(n+m)$ to $(n-m)\times(n-m)$. Thus null-space methods are particularly attractive when $n - m$ is small. If A is symmetric and positive semi-definite, then $Z^T AZ$ is symmetric positive definite and efficient solvers can be used to solve the reduced system (1.2). Moreover, an important feature of the saddle point systems that arise in solving a quadratic program using the active-set method is that the successive iterations only differ in that B has one row added or deleted. The null-space method is able to use this feature advantageously to reduce the work involved by updating the factors.

Another way to solve large-scale linear systems is to use an iterative method, with Krylov subspace methods being particularly popular. To be effective such methods generally need to be applied in conjunction with an appropriate preconditioner. The class of projected Krylov methods [23, 24], i.e. algorithms that solve (1.1) by implicitly projecting onto the null space, are a modern version of the null-space method; these are mathematically equivalent to applying a Krylov subspace method preconditioned with a constraint preconditioner [32] to equation (1.2).

There has been a sizable body of work—some historical, but much recent—that has revisited the null-space method, directly or indirectly, and put it into the framework of a matrix factorization, e.g., [2, 12, 14, 37, 39, 44, 48, 34, 20, 22, 32, **?**]. The main contribution of this report is to bring these factorizations together in a unified framework, highlight the relationships between them (some of which do not appear to be well known), and to compare their merits. The rest of this report is laid out as follows. In Section 2, we discuss the matrix factorizations that result from different choices of the null space basis. In Section 3, we explore the suitability of these factorizations as an alternative to a standard sparse direct indefinite LDL^T solver. We test a number of different forms of the factorization on a range of problems, both academic and practical, and report numerical results to illustrate the stability and sparsity of the computed factors. In Section 4, we present two novel preconditioners—together with an eigenanalysis—which are developed by considering the null-space method as a factorization. We conclude with some final remarks in Section 5.

2. Null-space methods as a factorization. Consider again the system (1.1). The primal variable $\mathbf{x}$ can be expressed in the form

$$\mathbf{x} = Z\mathbf{x}_N + Y\mathbf{x}_R, \tag{2.1}$$

where $Y \in \mathbb{R}^{n\times m}$ is chosen so that $\begin{bmatrix} Z & Y \end{bmatrix}$ spans $\mathbb{R}^n$. Thus

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} Y & Z & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} \mathbf{x}_R \\ \mathbf{x}_N \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix}$$

and hence

$$\begin{bmatrix} Y^T & 0 \\ Z^T & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} Y & Z & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} \mathbf{x}_R \\ \mathbf{x}_N \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} Y^T & 0 \\ Z^T & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix},$$

and so

$$\begin{bmatrix} Y^TAY & Y^TAZ & Y^TB^T \\ Z^TAY & Z^TAZ & 0 \\ BY & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}_R \\ \mathbf{x}_N \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} Y^T & 0 \\ Z^T & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix}. \tag{2.2}$$

It is clear that this is a matrix representation of Algorithm 1. First, $\widehat{\mathbf{x}} = Y\mathbf{x}_R$ is found by solving the linear system $BY\mathbf{x}_R = \mathbf{g}$. Then, the component $\mathbf{x}_N$ of $\mathbf{x}$ in the null space of B is found by solving the linear system

$$Z^TAZ\mathbf{x}_N = Z^T(\mathbf{f} - AY\mathbf{x}_R) = Z^T(\mathbf{f} - A\widehat{\mathbf{x}}).$$

Finally, $\mathbf{y}$ is recovered by solving

$$Y^TB^T\mathbf{y} = Y^T\mathbf{f} - Y^TAY\mathbf{x}_R - Y^TAZ\mathbf{x}_N = Y^T(\mathbf{f} - A\mathbf{x}).$$

Note that the matrix $\begin{bmatrix} Y^T & 0 \\ Z^T & 0 \\ 0 & I \end{bmatrix}$ is square and non-singular, and so using (2.1) and (2.2), we can rewrite (1.1) as

$$\begin{bmatrix} Y^T & 0 \\ Z^T & 0 \\ 0 & I \end{bmatrix}^{-1} \begin{bmatrix} Y^TAY & Y^TAZ & Y^TB^T \\ Z^TAY & Z^TAZ & 0 \\ BY & 0 & 0 \end{bmatrix} \begin{bmatrix} Y & Z & 0 \\ 0 & 0 & I \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix}.$$

Thus the factorization

$$\mathcal{A} = \begin{bmatrix} \begin{bmatrix} Y^T \\ Z^T \end{bmatrix}^{-1} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} Y^TAY & Y^TAZ & Y^TB^T \\ Z^TAY & Z^TAZ & 0 \\ BY & 0 & 0 \end{bmatrix} \begin{bmatrix} \begin{bmatrix} Y & Z \end{bmatrix}^{-1} & 0 \\ 0 & I \end{bmatrix} \tag{2.3}$$

is an LTL^T factorization, with L lower triangular and T reverse block triangular, that is equivalent to the null-space method. Since there are infinitely many potential bases Y, Z, this factorization is non-unique and the main difficulty of the null-space method is choosing these bases. In the following subsections, we discuss some special cases that have been proposed in the literature.

2.1. Schilders and related factorizations. One way of fixing Y in (2.1) is to extend B^T to an $n \times n$ non-singular matrix $\begin{bmatrix} B^T & V^T \end{bmatrix}$. If we choose Y and Z satisfying

$$\begin{bmatrix} Y^T \\ Z^T \end{bmatrix}^{-1} = \begin{bmatrix} B^T & V^T \end{bmatrix},$$

then it is easy to see that $BZ = 0$, and $BY = I$. The factorization (2.3) reduces to

$$\mathcal{A} = \begin{bmatrix} B^T & V^T & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} Y^T A Y & Y^T A Z & I \\ Z^T A Y & Z^T A Z & 0 \\ I & 0 & 0 \end{bmatrix} \begin{bmatrix} B & 0 \\ V & 0 \\ 0 & I \end{bmatrix}.$$

This factorization was given by Fletcher and Johnson [18, Equation (3.6)], who described it as 'readily observed (but not well known)'.

2.1.1. A first null-space factorization. Suppose we have a non-singular subset of m columns of B. We may then write, without loss of generality, $B = \begin{bmatrix} B_1 & B_2 \end{bmatrix}$, where $B_1 \in \mathbb{R}^{m \times m}$ is non-singular. If, as suggested by Fletcher and Johnson, we make the choice of $V = \begin{bmatrix} 0 & I \end{bmatrix}$, then

$$\begin{bmatrix} B^T & V^T \end{bmatrix}^{-1} = \begin{bmatrix} B_1^T & 0 \\ B_2^T & I \end{bmatrix}^{-1} = \begin{bmatrix} B_1^{-T} & 0 \\ -B_2^T B_1^{-T} & I \end{bmatrix} \left(= \begin{bmatrix} Y^T \\ Z^T \end{bmatrix} \right).$$

This gives us the bases

$$Z_f = \begin{bmatrix} -B_1^{-1} B_2 \\ I \end{bmatrix}, \qquad Y_f = \begin{bmatrix} B_1^{-1} \\ 0 \end{bmatrix}. \tag{2.4}$$

This choice for Z is often called the *fundamental* basis [3, Section 6], and we consequently label it Z_f.

Substituting (2.4) into (2.3) gives the factorization

$$\mathcal{A} = \begin{bmatrix} B_1^T & 0 & 0 \\ B_2^T & I & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} B_1^{-T} A_{11} B_1^{-1} & B_1^{-T} X^T & I \\ X B_1^{-1} & N & 0 \\ I & 0 & 0 \end{bmatrix} \begin{bmatrix} B_1 & B_2 & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix}, \tag{2.5}$$

where

$$N = Z_f^T A Z_f, \tag{2.6}$$

denotes the $(n-m) \times (n-m)$ null-space matrix and

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \text{ and } X = Z_f^T \begin{bmatrix} A_{11} \\ A_{12} \end{bmatrix},$$

with $A_{11} \in \mathbb{R}^{m \times m}$. It is easy to see that (2.5) is equivalent to

$$\mathcal{A} = \underbrace{\begin{bmatrix} I & 0 & 0 \\ B_2^T B_1^{-T} & I & 0 \\ 0 & 0 & I \end{bmatrix}}_{\mathcal{L}_1} \underbrace{\begin{bmatrix} A_{11} & X^T & B_1^T \\ X & N & 0 \\ B_1 & 0 & 0 \end{bmatrix}}_{\mathcal{T}_1} \underbrace{\begin{bmatrix} I & B_1^{-1} B_2 & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix}}_{\mathcal{L}_1^T}. \tag{2.7}$$

Indeed, this LTL^T factorization appeared in the survey paper by Benzi, Golub and Liesen [3, Equation (10.35)], where it was attributed to a personal communication from Michael Saunders and was described as being 'related to the null-space method'. We will refer to this decomposition as **Factorization 1**. The factor $\mathcal{L}_1$ is well-conditioned provided B_1 is chosen appropriately; this is discussed in Section 3. Factorization 1 is then a stable factorization of the $\mathcal{A}$ and the diagonal blocks of $\mathcal{T}_1$ (that is, B_1, N and B_1^T) accurately reflect the condition of the full system.

In practice, once we have solved

$$B_1\mathcal{B} = B_2, \tag{2.8}$$

for $\mathcal{B}$ and performed a factorization of the null-space matrix N, the only additional calculations required to generate Factorization 1 are matrix-matrix products with $\mathcal{B}$, plus some matrix additions. This is summarised in Table 2.1. Note that, in some applications, e.g., where N is large and dense, it may not be possible to explicitly compute N and form its factorization; in this case it is necessary to use an iterative solver.

solve $B_1\mathcal{B} = B_2$ for $\mathcal{B}$
mat-mat $\begin{cases} 1 \,@\, [(\,(n-m)\times m\,)\times(m\times m)] \\ 2 \,@\, [(\,(n-m)\times m\,)\times(\,(n-m)\times m\,)] \end{cases}$
mat-add $\begin{cases} 1 \,@\, (n-m)\times m \\ 2 \,@\, (n-m)\times(n-m) \end{cases}$
An $(n-m)\times(n-m)$ (possibly sparse) factorization of N

Table 2.1: Cost of forming Factorization 1 assuming B_1, B_2 chosen. mat-mat and mat-add denote a sparse matrix-matrix product and sparse matrix addition, respectively.

Table 2.2 gives the costs of applying Factorization 1 to solve the system (1.1). We give two variants: implicit and explicit. In the explicit version, we store the off-diagonal matrices (X, $\mathcal{B}$) in (2.7) — which are formed in the construction of N — and apply them via matrix-vector products to solve (1.1). In the implicit version, we discard the off-diagonal matrices and re-compute them as needed. This is computationally more expensive, but saves storing the potentially dense matrices X and $\mathcal{B}$ of size $m \times (n-m)$. It is clear that the implicit version is exactly equivalent to the null-space method as presented in Algorithm 1.

The costs in Tables 2.1 and 2.2 are upper bounds, and they may be reduced in certain circumstances. For example, as we discuss in Section 3 below, it is usual to find $B_1,\ B_2$ by forming an LU factorization of B^T

$$B^T = (PLP^T)(PUQ) = \left(P\begin{bmatrix} L_a & 0 \\ L_b & I \end{bmatrix}P^T\right)\left(P\begin{bmatrix} U_a \\ 0 \end{bmatrix}Q\right),$$

where L_a is lower triangular, U_a is upper triangular, and P and Q are permutation matrices. Then $B_1^{-1}B_2 = PL_a^{-T}L_b^TP^T$, and so this can be calculated without reference to U_a, although L_b is needed and is likely to be less sparse than B_2. Furthermore, in this case $Z = L^{-T}P^T\begin{bmatrix} I \\ 0 \end{bmatrix}$, so Z^TAZ can also be formed efficiently. See, e.g., Fletcher and Johnson [18] for more details.

2.1.2. A factorization due to Lungten, Schilders and Maubach. Assume now that A is symmetric and positive semi-definite so that N is symmetric positive definite and a Cholesky factorization of the form $N = L_2L_2^T$ exists, where L_2 is lower triangular. Then we can decompose the reverse triangular $\mathcal{T}_1$ matrix in (2.7) as

$$\mathcal{T}_1 = \begin{bmatrix} A_{11} & X^T & B_1^T \\ X & N & 0 \\ B_1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} I & 0 & L_A \\ 0 & L_2 & X \\ 0 & 0 & B_1 \end{bmatrix}\begin{bmatrix} -D_A & 0 & I \\ 0 & I & 0 \\ I & 0 & 0 \end{bmatrix}\begin{bmatrix} I & 0 & 0 \\ 0 & L_2^T & 0 \\ L_A^T & X^T & B_1^T \end{bmatrix},$$

Explicit	solve $\begin{cases} B_1\mathcal{B} = B_2 \text{ for } \mathcal{B} \text{ 2 times} \\ 2 \,@\, (n-m) \times (n-m) \text{ (triangular solves using factors of } N) \end{cases}$ mat-vec $\begin{cases} 2 \,@\, (n-m) \times m \\ 1 \,@\, m \times m \\ 2 \,@\, m \times (n-m) \end{cases}$ vec-add $\begin{cases} 4 \,@\, (n-m) \\ 1 \,@\, m \end{cases}$
Implicit	As explicit† plus: solve $B_1\mathcal{B} = B_2$ for $\mathcal{B}$ 4 times mat-vec $\begin{cases} 1 \,@\, (n-m) \times m \\ 1 \,@\, m \times m \\ 1 \,@\, m \times (n-m) \end{cases}$ vec-add: $1 \times (n-m)$

Table 2.2: Cost of applying Factorization 1. Here mat-vec denotes the product of a sparse matrix with a vector, and vec-add denotes the addition of two vectors. †The numbers of solves and matrix-vector products will be the same, but the matrices in the matrix-vector products will generally be sparser in the implicit case.

where

$$A_{11} = L_A - D_A + L_A^T, \tag{2.9}$$

with L_A a strictly lower triangular matrix and D_A a diagonal matrix. Combining the outer matrices here with the outer matrices in (2.7) yields the alternative, but equivalent, LTL^T factorization

$$\mathcal{A} = \underbrace{\begin{bmatrix} I & 0 & L_A \\ B_2^T B_1^{-T} & L_2 & K \\ 0 & 0 & B_1 \end{bmatrix}}_{\mathcal{L}_2} \underbrace{\begin{bmatrix} -D_A & 0 & I \\ 0 & I & 0 \\ I & 0 & 0 \end{bmatrix}}_{\mathcal{T}_2} \underbrace{\begin{bmatrix} I & B_1^{-1}B_2 & 0 \\ 0 & L_2^T & 0 \\ L_A^T & K^T & B_1^T \end{bmatrix}}_{\mathcal{L}_2}, \tag{2.10}$$

where $K = X + B_2^T B_1^{-T} L_A$. This factorization was recently proposed both for use as a direct method and as the basis of a preconditioner for an iterative method by Lungten, Schilders and Maubach [34]. We refer to it as **Factorization 2**, or the LSM factorization.

Note that forming (2.10) is more expensive than (2.7), as it requires one more matrix-matrix multiply of $\mathcal{B}^T$ (recall (2.8)) with an $m \times m$ matrix and one more $(n-m) \times m$ matrix addition, both coming from the formation of K. In terms of applying (2.10) explicitly, one matrix-vector multiply with A_{11} is replaced by matrix-vector multiplies with its strictly upper, lower and diagonal parts, and two extra $m \times m$ matrix additions; there is a similar increase in cost when applying the factorization implicitly. This suggests that, in terms of the computational cost, Factorization 1 is preferable; we perform tests with both versions in Section 3.

Lungten et al. [34] focus on problems for which the non-singular matrix B_1 is also upper triangular (or it is easy to transform the problem into this form). In this

case, if the factorization (2.10) is formed via an adapted Cholesky algorithm, they show that it takes

$$\frac{1}{3}(n^3m^3) + \frac{1}{2}(n^2 - 7m^2) - \frac{1}{6}(5n + m) + nm(n - m + 4)$$

flops to factorize the saddle point matrix $\mathcal{A}$.

2.1.3. The Schilders factorization. We next consider the relationship of these factorizations to the so-called Schilders factorization. Dollar and Wathen [14] were interested in developing constraint preconditioners for symmetric systems of the form (1.1). Such preconditioners represent the blocks B exactly, but approximate the (1,1) block A. Dollar and Wathen choose symmetric matrices $E_1 \in \mathbb{R}^{m\times m}$, $E_2 \in \mathbb{R}^{(n-m)\times(n-m)}$ (E_2 non-singular), together with matrices $F_1 \in \mathbb{R}^{m\times m}$, $F_2 \in \mathbb{R}^{(n-m)\times(n-m)}$ (F_2 non-singular), and $M \in \mathbb{R}^{(n-m)\times m}$. To obtain an inexpensive preconditioner, E_2, F_2 are chosen so that they are easy to invert. The Schilders factorization is then given by

$$\begin{bmatrix} A_{11} & A_{21}^T & B_1^T \\ A_{21} & A_{22} & B_2^T \\ B_1 & B_2 & 0 \end{bmatrix} = \begin{bmatrix} B_1^T & 0 & F_1 \\ B_2^T & F_2 & M \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} E_1 & 0 & I \\ 0 & E_2 & 0 \\ I & 0 & 0 \end{bmatrix} \begin{bmatrix} B_1 & B_2 & 0 \\ 0 & F_2^T & 0 \\ F_1^T & M^T & I \end{bmatrix}, \qquad (2.11)$$

where

$$\begin{aligned} A_{11} &= F_1B_1 + B_1^TF_1^T + B_1^TE_1B_1 \\ A_{21} &= B_2^TF_1^T + MB_1 + B_2^TE_1B_2 \\ A_{22} &= F_2E_2F_2^T + MB_2 + B_2^TM^T + B_2^TE_1B_2. \end{aligned}$$

Note that the (1,1) block A is implicitly defined by the choices of E_i, F_i. Nevertheless, we can use this construction to give a factorization for a given A.

One possible choice for E_1, F_1 is

$$E_1 = -B_1^{-T}D_AB_1^{-1}, \qquad F_1 = L_AB_1^{-1},$$

for the D_A, L_A in (2.9). The matrices M, E_2 and F_2 are then given by the relations

$$\begin{aligned} M &= (A_{21} - B_2^TF_1^T - B_2^TE_1B_2)B_1^{-1}, \\ F_2E_2F_2^T &= MB_2 + B_2^TM^T + B_2^TE_1B_2 - A_{22}. \end{aligned}$$

With these choices, transferring a factor of the block diagonal matrix with diagonal blocks B_1^T, I and B_1^{-1} from the left outer matrix to the central matrix in (2.11) again gives Factorization 2.

The original Schilders factorization [48], of which the formulation (2.11) is a generalization, was given only for matrices for which B_1 is upper triangular, and used the choice

$$E_1 = \texttt{diag}(B_1^{-T}A_{11}B_1^{-1}), \qquad F_1 = B_1^T\texttt{lower}(B_1^{-T}A_{11}B_1^{-1}),$$

where `diag`() and `lower`() denote the diagonal and strictly lower triangular parts of a matrix, respectively. Again, it is straightforward to show the equivalence of this factorization to (2.10). Generating this factorization is clearly significantly more work than (2.7), not least because we are unable to re-use the matrix $\mathcal{B}$ in forming the sub-blocks.

There are, of course, other ways of rearranging the decomposition (2.7). Dollar et al. [13] give a list of forms the factorization can take. As already observed, their focus was on implicit factorizations of constraint preconditioners and only five of their factorizations are applicable in the case of an arbitrary symmetric (1,1) block.

2.2. Relationship to the Schur complement factorization. A commonly used block LDL^T factorization for generalized saddle point systems (which have a negative definite (2,2) block) where A is nonsingular is:

$$\begin{bmatrix} A & B^T \\ B & -C \end{bmatrix} = \begin{bmatrix} I & 0 \\ BA^{-1} & I \end{bmatrix} \begin{bmatrix} A & 0 \\ 0 & -(C + BA^{-1}B^T) \end{bmatrix} \begin{bmatrix} I & A^{-1}B^T \\ 0 & I \end{bmatrix}, \tag{2.12}$$

see, e.g., [3, Equation (3.9)]. Approximating the terms of this factorization has proved an effective strategy for developing preconditioners for saddle point systems. It can also be used to develop another factorization that is equivalent to the null-space method. First, note that

$$\mathcal{A} = \begin{bmatrix} I & 0 & 0 \\ 0 & 0 & I \\ 0 & I & 0 \end{bmatrix} \begin{bmatrix} A_{11} & B_1^T & A_{12} \\ B_1 & 0 & B_2 \\ A_{21} & B_2^T & A_{22} \end{bmatrix} \begin{bmatrix} I & 0 & 0 \\ 0 & 0 & I \\ 0 & I & 0 \end{bmatrix},$$

where we are again assuming, without loss of generality, that B_1 is a non-singular $m \times m$ sub-block of B. Applying (2.12) with $C = A_{22}$, we obtain

$$\begin{aligned} \mathcal{A} = & \begin{bmatrix} I & 0 & 0 \\ 0 & 0 & I \\ 0 & I & 0 \end{bmatrix} \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ B_2^T B_1^{-T} & A_{21}B_1^{-1} - B_2^T B_1^{-T} A_{11} B_1^{-1} & I \end{bmatrix} \\ & \begin{bmatrix} A_{11} & B_1^T & 0 \\ B_1 & 0 & 0 \\ 0 & 0 & S \end{bmatrix} \begin{bmatrix} I & 0 & B_1^{-1}B_2 \\ 0 & I & B_1^{-T}A_{21} - B_1^{-T}A_{11}B_1^{-1}B_2 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} I & 0 & 0 \\ 0 & 0 & I \\ 0 & I & 0 \end{bmatrix} \\ = & \begin{bmatrix} I & 0 & 0 \\ B_2^T B_1^{-T} & XB_1^{-1} & I \\ 0 & I & 0 \end{bmatrix} \begin{bmatrix} A_{11} & B_1^T & 0 \\ B_1 & 0 & 0 \\ 0 & 0 & S \end{bmatrix} \begin{bmatrix} I & B_1^{-1}B_2 & 0 \\ 0 & B_1^{-T}X^T & I \\ 0 & I & 0 \end{bmatrix}. \end{aligned}$$

Here S denotes the Schur complement, which satisfies

$$\begin{aligned} S &= A_{22} - \begin{bmatrix} A_{21} & B_2^T \end{bmatrix} \begin{bmatrix} A_{11} & B_1^T \\ B_1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} A_{12} \\ B_2 \end{bmatrix} \\ &= A_{22} - \begin{bmatrix} A_{21} & B_2^T \end{bmatrix} \begin{bmatrix} 0 & B_1^{-1} \\ B_1^{-T} & -B_1^{-T}A_{11}B_1^{-1} \end{bmatrix} \begin{bmatrix} A_{12} \\ B_2 \end{bmatrix} \\ &= A_{22} - B_2^T B_1^{-T} A_{21} - A_{12}B_1^{-1}B_2 + B_2^T B_1^{-T} A_{11} B_1^{-1} B_2 \\ &= Z_f^T A Z_f = N. \end{aligned}$$

It follows that the null-space matrix (2.6) is the Schur complement for an alternative blocking of the matrix, and we have the factorization

$$\mathcal{A} = \begin{bmatrix} I & 0 & 0 \\ B_2^T B_1^{-T} & XB_1^{-1} & I \\ 0 & I & 0 \end{bmatrix} \begin{bmatrix} A_{11} & B_1^T & 0 \\ B_1 & 0 & 0 \\ 0 & 0 & N \end{bmatrix} \begin{bmatrix} I & B_1^{-1}B_2 & 0 \\ 0 & B_1^{-T}X^T & I \\ 0 & I & 0 \end{bmatrix}. \tag{2.13}$$

Again, this can be derived from equation (2.7) by simply noting that the reverse triangular matrix $\mathcal{T}_1$ is equal to the product

$$T = \begin{bmatrix} I & 0 & 0 \\ 0 & XB_1^{-1} & I \\ 0 & I & 0 \end{bmatrix} \begin{bmatrix} A_{11} & B_1^T & 0 \\ B_1 & 0 & 0 \\ 0 & 0 & N \end{bmatrix} \begin{bmatrix} I & 0 & 0 \\ 0 & B_1^{-T}X^T & I \\ 0 & I & 0 \end{bmatrix}.$$

This factorization is therefore not 'new' — and it is significantly more expensive to form than (2.7) — but it highlights the connection between Schur complement methods and null-space factorizations. In particular, given the success in finding approximations to the Schur complement using techniques from functional analysis (see, e.g., Mardel and Winther [35]) it is hoped that viewing the null-space matrix this way could yield alternative preconditioners for certain classes of saddle point systems; this is developed further in Section 4.

2.3. Connection with Cholesky decomposition. Schilders developed his original factorization and the subsequent variant (2.10), by considering what he terms a *micro-factorization*. In this formulation, the matrix (1.1) is reordered by pairing every entry on the diagonal of the (1,1) block A with a corresponding non-zero entry in the constraint block B, so that, after permutations, the entries on the diagonal form micro saddle point systems. This is known as a *tiling* in the optimization community. Below is an illustrative example of this ordering for $n = 3,\ m = 2$:

$$P_1^T \mathcal{A} P_1 = \widetilde{\mathcal{A}} = \begin{bmatrix} a_{11} & b_{11} & a_{12} & b_{21} & a_{13} & b_{31} & a_{14} & a_{15} \\ b_{11} & 0 & b_{12} & 0 & b_{13} & 0 & b_{14} & b_{15} \\ a_{21} & b_{12} & a_{22} & b_{22} & a_{23} & b_{32} & a_{24} & a_{25} \\ b_{21} & 0 & b_{22} & 0 & b_{23} & 0 & b_{24} & b_{25} \\ a_{31} & b_{13} & a_{32} & b_{23} & a_{33} & b_{33} & a_{34} & a_{35} \\ b_{31} & 0 & b_{32} & 0 & b_{33} & 0 & b_{34} & b_{35} \\ a_{41} & b_{14} & a_{42} & b_{24} & a_{43} & b_{34} & a_{44} & a_{54} \\ a_{51} & b_{15} & a_{52} & b_{25} & a_{53} & b_{35} & a_{45} & a_{55} \end{bmatrix}$$

Note that there is no requirement for a_{ij} to be combined with b_{ij} and b_{ji}; instead, a suitable pairing that preserves sparsity and is numerically stable should be chosen — see Section 3 for further discussion. This is an example of a constrained ordering [8, 49, 50].

Since the entries on the (block-)diagonal are now mini saddle point systems, that are chosen to be non-singular, a modified sparse Cholesky code can be used to solve this system, and it is guaranteed (at least in exact arithmetic) that this will not break down [48, Section 3]. By this process, a factorization $\widetilde{\mathcal{A}} = LDL^T$ can be computed, where D has 1×1 and 2×2 blocks on the diagonal in the appropriate places. Furthermore, uniqueness results in, e.g., [39], show that the factors generated by the Cholesky process will be equivalent to those generated by the (macro) factorizations described earlier in this section.

In addition to the work by Schilders et al. [39, 48, 34], this approach has been considered by Forsgren and Murray [19] (whose focus was on inertia control), Gould [22] and de Niet and Wubs [12]; each of these works, to varying degrees, made the connection between the micro-factorization and the null-space method.

2.4. The antitriangular factorization. An alternative matrix factorization can be obtained by assuming we have a QR factorization

$$B^T = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R \\ 0 \end{bmatrix}, \tag{2.14}$$

where $Q = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix}$ is $n \times n$ orthogonal and R is $m \times m$ upper triangular and non-singular. Then Q_1 spans the range of B^T and Q_2 spans the null space of B. We can

therefore substitute $Y = Q_1$ and $Z = Q_2$ into the factorization (2.3) to obtain

$$\mathcal{A} = \underbrace{\begin{bmatrix} Q_1 & Q_2 & 0 \\ 0 & 0 & I \end{bmatrix}}_{\mathcal{L}_3} \underbrace{\begin{bmatrix} Q_1^T A Q_1 & Q_1^T A Q_2 & R \\ Q_2^T A Q_1 & Q_2^T A Q_2 & 0 \\ R^T & 0 & 0 \end{bmatrix}}_{\mathcal{T}_3} \underbrace{\begin{bmatrix} Q_1^T & 0 \\ Q_2^T & 0 \\ 0 & I \end{bmatrix}}_{\mathcal{L}_3}. \tag{2.15}$$

We refer to this as **Factorization 3**, or the AT (for antitriangular) factorization.

The reverse triangular matrix $\mathcal{T}_3$ in (2.15) appeared in the proof of Theorem 2.1 in Keller, Gould and Wathen [32], which established eigenvalue bounds for systems (1.1) preconditioned with a constraint preconditioner, although the link to a factorization of $\mathcal{A}$ does not appear to have been made there. Very recently, Pestana and Wathen [44] derived Factorization 3 in the case A is symmetric and B is of full rank, and showed that it is (up to a permutation) the representation of the antitriangular factorization of Mastronardi and van Dooren [37] applied to a saddle point system. They also gave a flop-count for forming the factorization in the dense case. The work is dominated by performing the QR factorization (2.14), with additional work being two $n \times n$ matrix-matrix products and a subsequent factorization of $Q_2^T A Q_2$.

It is clear from the above formulation that — provided a rank-revealing QR is applied — Factorization 3 is well defined for a B with linearly dependent rows, and also for non-symmetric A.

An advantage of having an orthogonal null basis is that the reduced system (2.2) is guaranteed to be well conditioned if A is. However, even for sparse problems, Q_1 and Q_2 may be rather dense. The explicit version of the factorization requires storing $Q_1, Q_2, Q_1^T A Q_2$ and the lower triangular part of $Q_1^T A Q_1$ in addition to keeping the factors of $Q_2^T A Q_2$. For large systems this incurs prohibitive storage costs; see Section 3. Many sparse QR routines — e.g., SuiteSparseQR [11] – allow the user to compute the action of Q_i with a vector using the stored Householder reflectors, and using this product form of the factorization with this facility is the preferred option here.

Pestana and Wathen [44] give a complexity analysis of the factorization (2.15) in the dense case, and show that the number of flops required is

$$8mn^2 - 2m^2 - \frac{2}{3}m^3.$$

An alternative way of obtaining factorization (2.15), which again focused on the dense case, is given by Mastronardi and van Dooren [38], and the number of flops there is dependent on the size of m relative to n; see [44, Table 2.1]. Details of an efficient implementation which uses Level 3 BLAS are given by Bujanović and Kressner [9].

2.5. Other bases and converting between factorizations. Consider again the general factorization (2.3). As already observed, the basis matrices Z and Y are not unique. Let $G \in \mathbb{R}^{(n-m)\times(n-m)}$ be a non-singular matrix. Then, given some basis matrix Z, another factorization that is equivalent to (2.3) — and hence Algorithm 1 — is

$$\mathcal{A} = \begin{bmatrix} \begin{bmatrix} \widehat{Y}^T \\ Z^T G^T \end{bmatrix}^{-1} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \widehat{Y}^T A \widehat{Y} & \widehat{Y}^T A Z G & \widehat{Y}^T B^T \\ G^T Z^T A \widehat{Y} & G^T Z^T A Z G & 0 \\ B\widehat{Y} & 0 & 0 \end{bmatrix} \begin{bmatrix} \begin{bmatrix} \widehat{Y} & ZG \end{bmatrix}^{-1} & 0 \\ 0 \quad 0 & I \end{bmatrix}, \tag{2.16}$$

where $\widehat{Y}$ is a matrix such that $\begin{bmatrix}\widehat{Y} & GZ\end{bmatrix}$ spans $\mathbb{R}^n$.

For example, suppose we take the fundamental basis Z_f as (2.4), and assume we have its 'skinny' QR factorization $Z_f = Q_2 R_2$. Then, taking $G = R_2^{-1}$, we obtain $Z_f G = Q_2$. Now, let $B^T = Q_1 R_1$ be the 'skinny' QR factorization of B^T. Since B is assumed to be of full rank, we have that $\begin{bmatrix}Q_1 & Q_2\end{bmatrix}$ spans $\mathbb{R}^n$. Substituting this into (2.16), we see that these choices transform Factorization 1 (2.7) into Factorization 3 (2.15). Note that this construction is not advisable in practice, but we include it as a demonstration of how the different factorizations relate to each other.

3. Stability and sparsity. In this section, we assess the viability of the factorizations described in Section 2 as the basis of a sparse direct solver. Here we concentrate on the solution of a single problem (1.1) and do not exploit the advantages the null space factorizations can offer over a general indefinite solver in terms of updating the factors after a row is added to or removed from $\mathcal{A}$. For a factorization to be effective for solving sparse linear systems, it must be stable in the classical sense [28] and the factors should be as sparse as possible. We discuss these issues below.

To motivate the discussion, we present results using Factorizations 1, 2 and 3—i.e, equations (2.7), (2.10) and (2.15), respectively—as the basis of a direct method. We perform tests using MATLAB Version R2014a. We factorize the matrix N using `cholmod` [10] and use the `SuiteSparseQR` algorithm [11], both from Tim Davis' SuiteSparse package and applied via a MATLAB interface. For comparison, results for the MATLAB command `ldl` (with default settings) are also given. `ldl` employs the package `MA57` [15] from the HSL mathematical software library [30]. `MA57` is a sparse direct solver that is designed to efficiently and robustly solve symmetric indefinite sparse linear systems. While it was not exclusively intended for saddle point systems, it was at least partly designed with such systems in mind and is extensively used for their solution. To ensure a fair comparison, in all the tests we pre-scale the matrix using the scaling calculated by `ldl`.

In our experiments, we report the scaled backwards error for (1.1) given by

$$\|\mathcal{A}\mathbf{w} - \mathbf{b}\|_2 / \|\mathbf{b}\|_2, \tag{3.1}$$

where $\mathbf{w} = \begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}$ and $\mathbf{b} = \begin{bmatrix}\mathbf{f}\\ \mathbf{g}\end{bmatrix}$. For the explicit methods, the measure of fill we use to measure sparsity of the factorization is

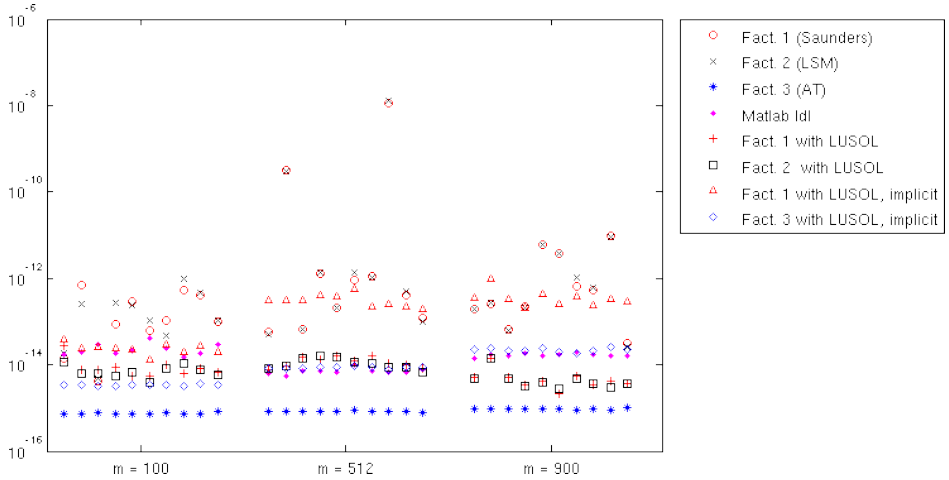
$$\text{fill} = \frac{\text{nnz}(\mathcal{L}_i) + \text{nnz}(\mathcal{T}_i)}{\text{nnz}(\mathcal{A})}, \tag{3.2}$$

where $\text{nnz}(\cdot)$ denotes the number of non-zeros required to store the matrix, $\mathcal{L}_i$ is the leftmost matrix in Factorization i, and $\mathcal{T}_i$ is the central matrix. For symmetric matrices, this is the number of non-zeros in the lower triangular part. Where we need to solve for blocks in the factorization — e.g., with B_1 and N in (2.7) — we replace the number of non-zeros in the original block with the number of non-zeros in an LU or Cholesky decomposition, as appropriate. For implicit methods, we only store the number of non-zeros needed by the factorizations of blocks we need to solve for; we assume that we have access to the original matrix, and do not count entries that can be obtained from there.

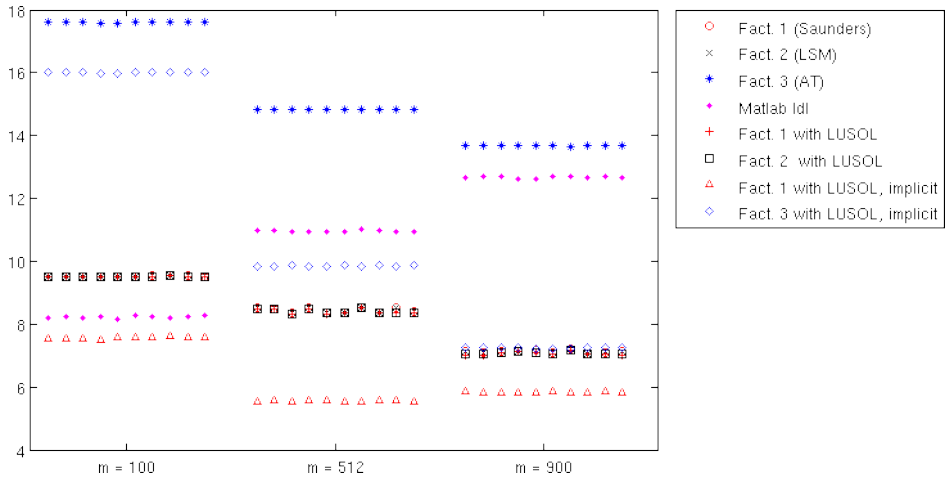
3.1. An Academic example. We start by considering some small academic matrices. We form a system of the form (1.1) using the MATLAB commands:

```
A = sprandsym(n,0.1,1e-1,2);
B = sprand(m,n,0.1);
B(1:m,1:m) = B(1:m,1:m) + 10*diag(rand(m,1));
b = [A B'; B sparse(m,m)]*rand(n+m,1);
```

We take $n = 1024$, and $m = \{100, 512, 900\}$ and construct the matrices so that the leading m columns of B form a non-singular sub-matrix; these can be used without permutation for B_1.



(a) Backwards error (3.1), without iterative refinement.



(b) Fill, as defined in (3.2).

Fig. 3.1: Measures of stability and sparsity for various factorizations. The results are averaged over ten runs with random matrices.

Figure 3.1 presents plots showing the backward error and fill. We see that, choosing B_1 to be the leading columns of B gives relatively sparse factors for Factorizations 1 and 2 but they are numerically unstable. If we perform an LU factorization on B^T (incorporating pivoting to ensure the L factor of B^T is well conditioned) and use this to obtain B_1, then $B_2^T B_1^{-T}$ is bounded and $\mathcal{L}_i$ $(i = 1, 2)$ is well-conditioned [18, 21]. In our tests, we use the package LUSOL [47] with partial pivoting (with a pivot tolerance of 1.9, as suggested for basis repair by the LUSOL documentation) to perform the sparse factorization of B^T. LUSOL is called via a MATLAB interface [27]. The resulting factorizations of (1.1) are stable.

Our results for the academic problems indicate that Factorization 3 (the antitriangular factorization) is the most numerically stable of the null-space factorizations, while exhibiting the worst sparsity properties (Figure 3.1b). The antitriangular factorization was shown to be backward stable by Mastronardi and van Dooren [37]. This is a consequence of the fact that the only calculation needed is a stable QR factorization; see also Gill and Murray [20]. On the other hand, even when using a sparse QR factorization routine, as expected, the Q matrices can fill in significantly.

Consider again the Factorizations 1 and 2, that require the identification of a non-singular sub-block B_1 of B. By using a factorization of this form we are essentially pre-determining an ordering that may or may not produce sparser factors than that used by a standard sparse direct solver. A weakness of a direct solver such as `MA57` for solving saddle point systems is that the fill in the computed factors may be significantly higher than is predicted for the chosen ordering on the basis of the sparsity pattern of the matrix alone. This is because numerical stability considerations may require the ordering to be modified during the factorization. This pivoting not only leads to a higher operation count and denser factors but also prohibits the exploitation of parallelism and significantly complicates the code development. The fact that the null space approach can give a direct method with a predictable level of fill without pivoting (other than to find B_1) may be an attractive feature.

Using a sparse direct LU solver to choose B_1 adds a computational overhead and can lead to a poor basis choice in terms of sparsity. It is possible to choose B_1 so that Z_f is sparse, for example, by ensuring B_1^{-1} is sparse. Pinar, Chow and Pothen [45] describe several ways to do this using methods based on graph matchings and hypergraph partitioning. However, such approaches are reportedly time consuming, and no results about the stability of the resulting decomposition are given. Murray [42] describes a method for picking Z so that $Z^T AZ$ is diagonal. However, to quote Fletcher [17, Section 10.1], this 'may be doubtful in terms of stability'.

An alternative approach is to apply Factorizations 1 and 2 to permuted matrices that will guarantee a stable factorization. Forsgren and Murray [19] — thinking in terms of a micro-factorization — describe a pivoting strategy to do this. Their method was refined further by Gould [22]. The disadvantage of these approaches is that the ordering is selected on the basis of stability, normally at the expense of sparsity, leading to the null-space matrix N being dense.

3.2. Optimization examples. We next consider examples from the optimization community; namely, a selection of relatively small quadratic programs from the CUTEst [25] test collection. In particular, the problems are taken from the Maros and Meszaros quadratic programming test set, and are solved without

consideration for any inequality constraints. These are of the form

$$\min \frac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{f}^T\mathbf{x}$$
$$\text{s.t. } B\mathbf{x} = \mathbf{g}.$$

We chose the subset of problems for which A is symmetric positive semi-definite, has at least $n - m$ non-zeros on the diagonal, and where $n > m$. Our test problems, and the relative sizes of their sub-matrices, are listed in Table 3.1.

Problem	m	n	$(n-m)/(n+m)$	Problem	m	n	$(n-m)/(n+m)$
LISWET1	10000	10002	9.999e-05	DPKLO1	77	133	2.667e-01
LISWET10	10000	10002	9.999e-05	STCQP2	2052	4097	3.326e-01
LISWET11	10000	10002	9.999e-05	CVXQP1_M	500	1000	3.333e-01
LISWET12	10000	10002	9.999e-05	CVXQP1_S	50	100	3.333e-01
LISWET2	10000	10002	9.999e-05	HS21	1	2	3.333e-01
LISWET3	10000	10002	9.999e-05	TAME	1	2	3.333e-01
LISWET4	10000	10002	9.999e-05	GOULDQP3	349	699	3.334e-01
LISWET5	10000	10002	9.999e-05	AUG2D	10000	20200	3.378e-01
LISWET6	10000	10002	9.999e-05	AUG2DC	10000	20200	3.378e-01
LISWET7	10000	10002	9.999e-05	AUG2DCQP	10000	20200	3.378e-01
LISWET8	10000	10002	9.999e-05	HS35	1	3	5.000e-01
LISWET9	10000	10002	9.999e-05	HS35MOD	1	3	5.000e-01
YAO	2000	2002	4.998e-04	MOSARQP1	700	2500	5.625e-01
LASER	1000	1002	9.990e-04	PRIMAL1	85	325	5.854e-01
CONT-300	90298	90597	1.653e-03	AUG3DC	1000	3873	5.896e-01
CONT-201	40198	40397	2.469e-03	AUG3DCQP	1000	3873	5.896e-01
CONT-101	10098	10197	4.878e-03	CVXQP2_M	250	1000	6.000e-01
CONT-200	39601	40397	9.950e-03	CVXQP2_S	25	100	6.000e-01
CONT-100	9801	10197	1.980e-02	PRIMAL3	111	745	7.407e-01
CONT-050	2401	2597	3.922e-02	PRIMAL2	96	649	7.423e-01
GENHS28	8	10	1.111e-01	PRIMAL4	75	1489	9.041e-01
QPCSTAIR	356	467	1.349e-01	PRIMALC1	9	230	9.247e-01
CVXQP3_M	750	1000	1.429e-01	PRIMALC2	7	231	9.412e-01
CVXQP3_S	75	100	1.429e-01	PRIMALC5	8	287	9.458e-01
HS76	3	4	1.429e-01	PRIMALC8	8	520	9.697e-01
MOSARQP2	600	900	2.000e-01	DUAL4	1	75	9.737e-01
DTOC3	9998	14999	2.001e-01	DUAL1	1	85	9.767e-01
HS51	3	5	2.500e-01	DUAL2	1	96	9.794e-01
HS52	3	5	2.500e-01	DUAL3	1	111	9.821e-01
HS53	3	5	2.500e-01	HUES-MOD	2	10000	9.996e-01
LOTSCHD	7	12	2.631e-01	HUESTIS	2	10000	9.996e-01

Table 3.1: The CUTEst test matrices.

We again select the basis B_1 using LUSOL with partial pivoting. We do not compare the antitriangular factorization here, as the cost of doing a QR factorization proved to be excessive (both in terms of timing and storage) for many of these problems.

The results are shown in Figures 3.2 and 3.3. First we consider the stability results from Figure 3.2. Both the null-space factorizations and the factorization formed by `ldl` give similar performance. The null-space factorizations give slightly better accuracy on the `CONT-xxx` problems, which are known to be difficult for direct solvers. If we apply one step of iterative refinement (as is usual in practical applications) then, as we see in Figure 3.2b, all methods behave comparably.

In Figure 3.3 we compare the sparsity of the factors. Factorizations 1 and 2 behave similarly, and, with a few exceptions, give denser factors than `ldl`. As expected, the implicit version of Factorization 1 requires less storage than the full version (at the cost of more computation). The implicit version of Factorization 1 needs the least storage overall for 46 problems, whereas `ldl` performs best for 19 problems (and the difference is negligible for the four `DUALx` problems). In particular, `ldl` does better for the `AUGxx` and `PRIMALxx` families of problems. The implicit version of Factorization 1 sometimes makes significant savings compared with the other methods. For example, for the `CONT-xxx` family it requires about a third of the storage needed by the next most effective method (`ldl`). However, in the cases where `ldl` give sparser factors, it often does so emphatically; about one hundred times more storage is required by Factorization 1 (implicit) compared to `ldl` for some of the `AUGxx` problems, and over fifteen hundred times the storage is needed for `HUES-MOD` and `HUESTIS`. Note that the cases where the null-space factorizations require significantly more storage tend to be where $n - m$ is large (in an absolute sense).

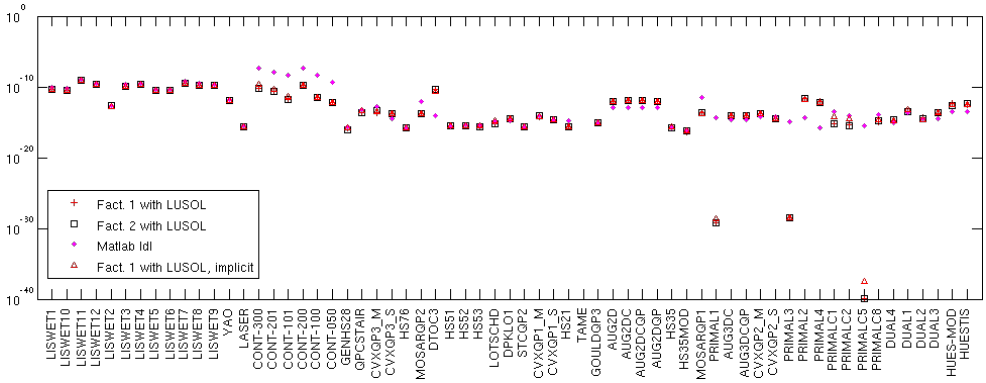
We do not report timings, as we cannot give a fair comparison between our proof-of-concept MATLAB code and the compiled `ldl` command, but we observed the null-space factorizations to be significantly slower. They do, however, potentially offer more scope for parallelization, as the bulk of the work in forming the factorization is in forming and factorizing N. The columns of N can be formed in parallel and, since the Cholesky factorization does not require pivoting, it can be parallelized more effectively than an indefinite sparse solver (see, for example, Hogg, Reid and Scott [29]). However, it is still necessary to factorize the non-square matrix B^T.

3.3. $\mathcal{F}$-matrices. Favourable sparsity *and* stability properties are possible for certain classes of matrices $\mathcal{A}$. de Niet and Wubs [12] show this for $\mathcal{F}-$matrices. An $\mathcal{F}-$matrix is a saddle point matrix (1.1) in which A is symmetric positive definite and B is a *gradient matrix*, that is, B has at most two entries per row, and if there are two entries their sum is zero. Such matrices appear naturally in certain formulations of fluid flow problems, e.g. [1], and also in electronic simulations, where they arise as the incidence matrix of the graph representing the circuit [52]. The original Schilders factorization [48] was developed specifically for solving systems with this structure.

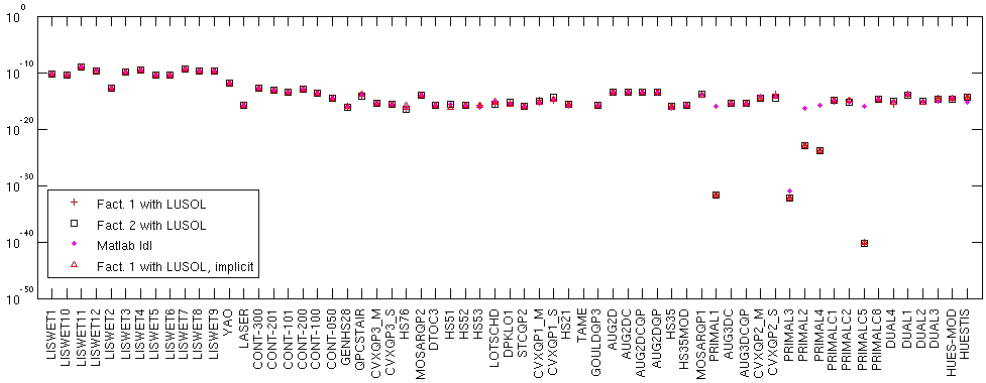
De Niet and Wubs essentially find the basis B_1 implicitly by considering a micro-factorization (recall Section 2.3). They pair entries in A with entries in B in such a way as to ensure the stability of the factorization. Moreover, de Niet and Wubs show that the number of non-zeros in the symbolic factorization of $F(A) \cup F(BB^T)$ (where $F(\cdot)$ denotes the sparsity pattern) 'provides a reasonable estimate' of the number of non-zeros in the factorization.

We give two examples below; one from resistor networks, and one from fluid flow. In the former the (1,1) block is a diagonal matrix, whereas we have a non-diagonal, but symmetric positive definite, (1,1) block in the latter.

3.3.1. Resistor networks. An important problem arising from the electronics industry is to find the voltage, V, and current, I, of a network of resistors. The current and voltages are related by the equation $AI + BV = 0$, where A is a diagonal



(a) Backwards error (3.1), no iterative refinement.



(b) Backwards error (3.1), one step of iterative refinement.

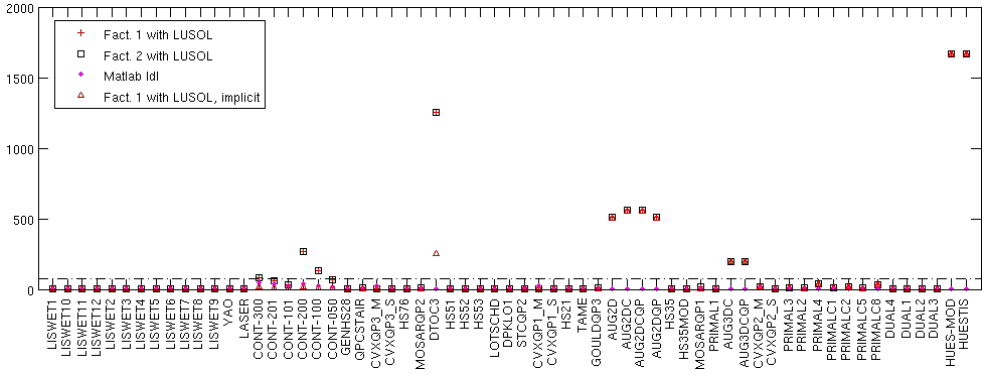
Fig. 3.2: Stability results for matrices from the CUTEst test set.

matrix containing the values of the resistors, and B is the incidence matrix of the network. From Kirchhoff's law, we also have $B^T I = 0$.

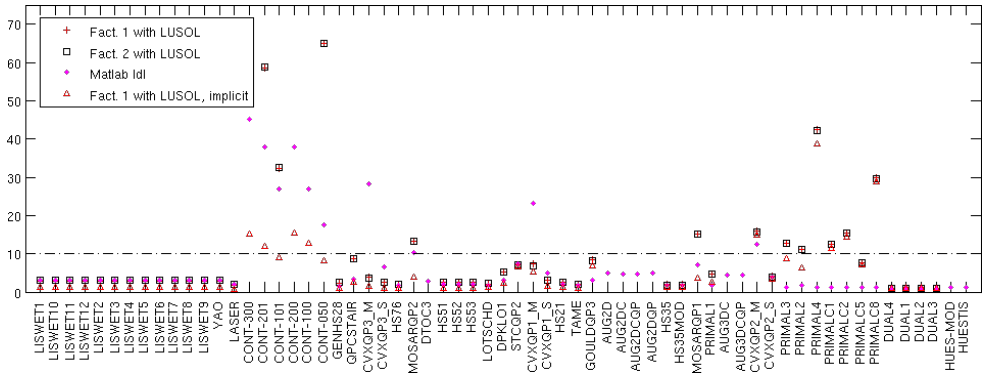
One node is usually grounded (algebraically, we remove a column of B) so that B has full rank. It is clear that putting these two equations together we get a system of the form (1.1) that is also an $\mathcal{F}-$matrix.

We run tests on these systems with $n = 1024$ resistors, joined at $m = \{100, 250, 512\}$ nodes at random (while forming a complete network). The resistor values are chosen at random from a uniform distribution between 0 and 10^{-2}. Plots showing backward errors and fill are given in Figure 3.4. A property of matrices of this type is that it is possible to permute B to make it upper trapezoidal, and so a well conditioned, easy to invert, block B_1 is possible to achieve without arithmetic (see [3, Chapter 6] for a discussion and references).

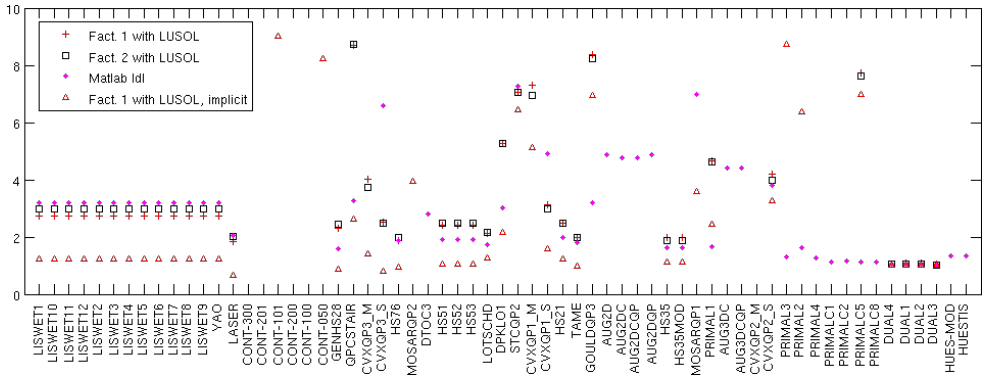
The results illustrate that the null-space factorizations are stable (as predicted by the theory in [12]); indeed, they produce smaller backward errors than `ldl` (without iterative refinement). The most stable factorization appears to be the implicit variant of Factorization 1. In terms of sparsity, the antitriangular factorization (Factorization 3) is again the poorest while Factorization 1 gives the sparsest factors.



(a) Fill, as defined in (3.2).

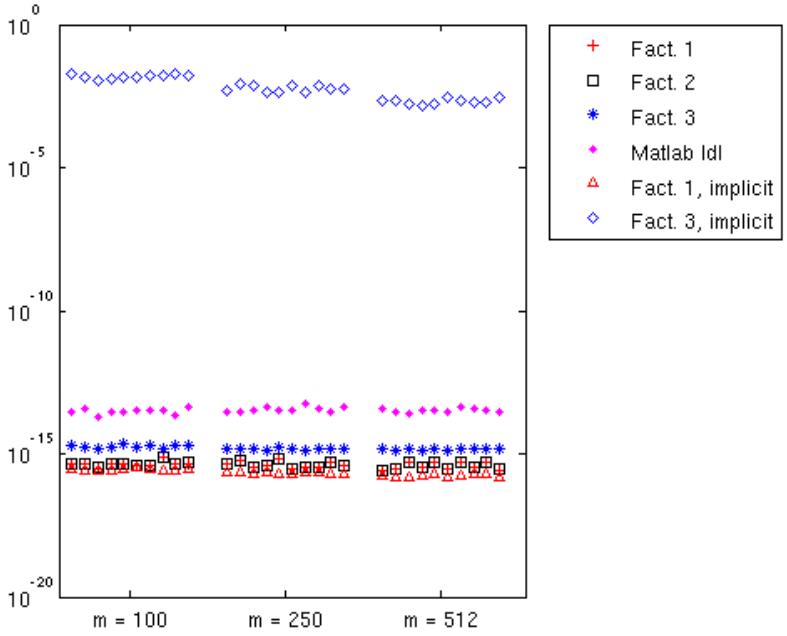


(b) Detail from Figure 3.3a.

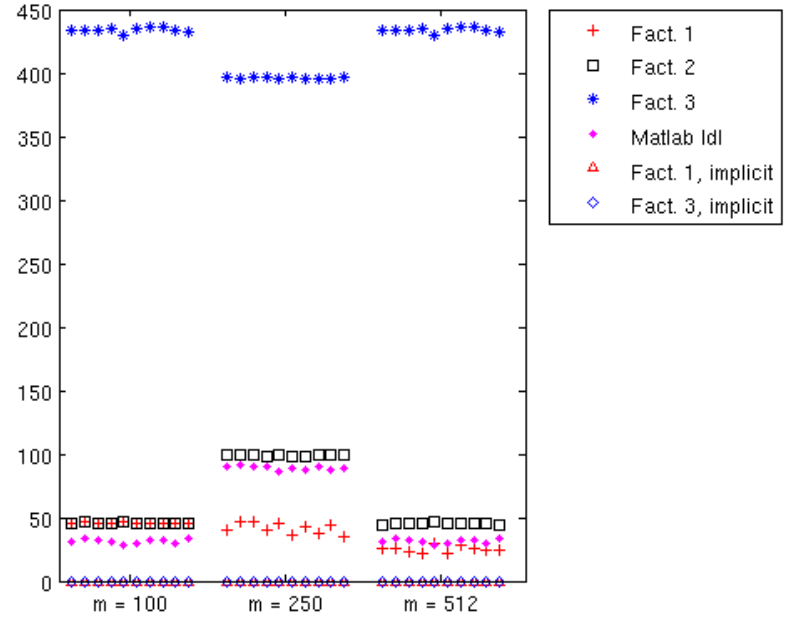


(c) Detail from Figure 3.3b.

Fig. 3.3: Sparsity results for matrices from the CUTEst test set.



(a) Backwards error (3.1), no iterative refinement.



(b) Fill, as defined in (3.2).

Fig. 3.4: Stability and sparsity results for the resistor network problem, as described in Section 3.3.1.

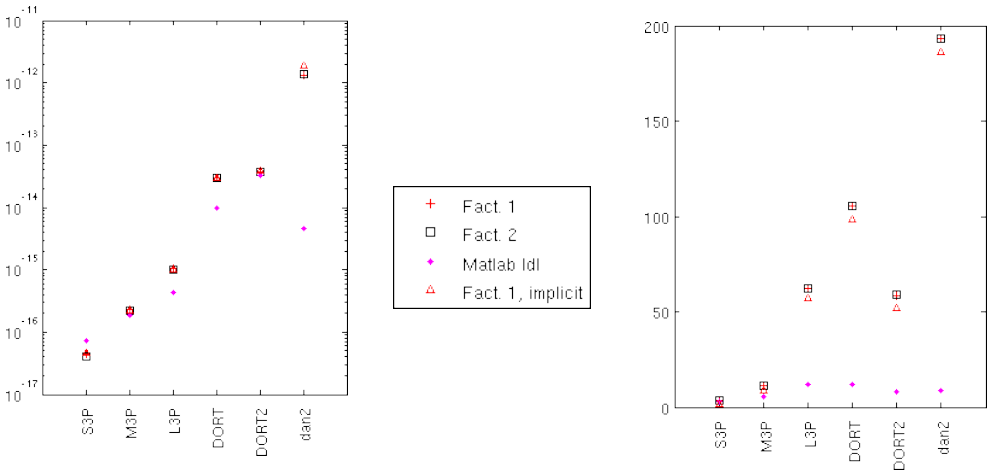
As we do not need to factorize a block of B, the fill for the factorizations based on the fundamental basis is negligible.

3.3.2. Fluid flow problems. Another source of $\mathcal{F}-$matrices is in fluid flow problems. In particular, we test some matrices derived from the mixed-hybrid finite element approximation of Darcy's law and the continuity equation, which describes fluid flow through porous media [36]. The test matrices we use[1] are described in Table 3.2; the same examples were used as test cases by, e.g., Tůma [51] and de Neit and Wubs [12].

Figure 3.5 shows sparsity and stability results for these matrices.

Problem	m	n	$(n-m)/(n+m)$
`S3P`	207	270	0.13208
`M3P`	1584	2160	0.15385
`L3P`	12384	17280	0.16505
`DORT`	9607	13360	0.16341
`DORT2`	5477	7515	0.15687
`dan2`	46661	63750	0.15478

Table 3.2: $\mathcal{F}-$matrices from a problem in fluid flow



(a) Backwards error (3.1), no iterative refinement.

(b) Fill, as defined in (3.2).

Fig. 3.5: Stability and sparsity results for $\mathcal{F}-$matrices from a problem in fluid flow, as described in Section 3.3.2.

For these real-word problems—especially the larger problems—the null-space factorizations perform markedly worse than `ldl`. Figure 3.5a shows that the backwards error is larger for the null-space factorizations than for `ldl`, although further tests showed that one step of iterative refinement is enough to make all the

[1] We thank Miroslav Tůma for providing us with these matrices.

backwards errors comparable (and of order machine precision). In terms of storage, the null-space factorizations can take over twenty times the storage that `ldl` needs.

4. As a preconditioner. As we saw in Section 2.1, the null-space method can be thought of as a Schur-complement factorization (2.13), incomplete versions of which have proved effective for preconditioning saddle point systems. Motivated by this, we present some eigenvalue bounds for incomplete versions of the factorization (2.13). Note that, due to the indefiniteness of the (1,1) block, the standard Murphy, Golub and Wathen [41] results are not applicable here. While convergence is not prescribed completely by the eigenvalue bounds for non-symmetric systems [26] they are often seen to be indicative of the convergence behaviour [43], and indeed that has been our experience here.

4.1. The central matrix. First, we consider the preconditioner formed by taking (a permutation of) the central matrix in the factorization (2.13),

$$\mathcal{P}_1 = \begin{bmatrix} A_{11} & 0 & B_1^T \\ 0 & N & 0 \\ B_1 & 0 & 0 \end{bmatrix}.$$

This would correspond to the block diagonal preconditioner in the Schur complement decomposition (2.12). First, we give an eigen-analysis of the preconditioned system.

THEOREM 4.1. *The generalized eigenvalues such that*

$$\begin{bmatrix} A_{11} & A_{12} & B_1^T \\ A_{21} & A_{22} & B_2^T \\ B_1 & B_2 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{y} \end{bmatrix} = \lambda \begin{bmatrix} A_{11} & 0 & B_1^T \\ 0 & N & 0 \\ B_1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{y} \end{bmatrix}, \tag{4.1}$$

where $N = Z_f^T A Z_f$, *satisfy* $\lambda = 1$ *or*

$$\frac{1 + \sigma_{\min} \pm \sqrt{\sigma_{\min}^2 + 2\sigma_{\min} - 3}}{2} \leq \lambda \leq \frac{1 + \sigma_{\max} \pm \sqrt{\sigma_{\max}^2 + 2\sigma_{\max} - 3}}{2},$$

where $\sigma_{\min}$ *and* $\sigma_{\max}$ *are the smallest and largest eigenvalues of* $N^{-1}A_{22}$.

Proof. First, the last equation in (4.1) gives

$$B_1\mathbf{x}_1 + B_2\mathbf{x}_2 = \lambda B_1\mathbf{x}_1, \Rightarrow (1-\lambda)B_1\mathbf{x}_1 = -B_2\mathbf{x}_2.$$

Then, either $\lambda = 1$ (and hence $B_2\mathbf{x}_2 = \mathbf{0}$), or

$$\mathbf{x}_1 = \frac{-1}{1-\lambda} B_1^{-1} B_2 \mathbf{x}_2. \tag{4.2}$$

Now, the first equation in (4.1) gives

$$(1-\lambda)A_{11}\mathbf{x}_1 + A_{12}\mathbf{x}_2 = (\lambda - 1)B_1^T\mathbf{y},$$

and re-arranging and substituting in (4.2) gives

$$\begin{aligned} \frac{1}{1-\lambda} A_{11} B_1^{-1} B_2 \mathbf{x}_2 - \frac{1}{1-\lambda} A_{12}\mathbf{x}_2 &= B_1^T \mathbf{y} \\ \frac{1}{1-\lambda} B_2^T B_1^{-T} A_{11} B_1^{-1} B_2 \mathbf{x}_2 - \frac{1}{1-\lambda} B_2^T B_1^{-T} A_{12}\mathbf{x}_2 &= B_2^T \mathbf{y}. \end{aligned} \tag{4.3}$$

We now turn our attention to the second equation in (4.1):

$$A_{21}\mathbf{x}_1 + A_{22}\mathbf{x}_2 + B_2^T\mathbf{y} = \lambda N\mathbf{x}_2.$$

Substituting in (4.2) and (4.3) gives

$$\frac{-1}{1-\lambda}A_{21}B_1^{-1}B_2\mathbf{x}_2 + A_{22}\mathbf{x}_2 + \frac{1}{1-\lambda}B_2^TB_1^{-T}A_{11}B_1^{-1}B_2\mathbf{x}_2 - \frac{1}{1-\lambda}B_2^TB_1^{-T}A_{12}\mathbf{x}_2 = \lambda N\mathbf{x}_2.$$

Rearranging and using the definition of N, we get

$$\begin{aligned}\frac{1}{1-\lambda}N\mathbf{x}_2 + \left(1 - \frac{1}{1-\lambda}\right)A_{22}\mathbf{x}_2 &= \lambda N\mathbf{x}_2 \\ (1-\lambda+\lambda^2)N\mathbf{x}_2 &= \lambda A_{22}\mathbf{x}_2. \end{aligned} \tag{4.4}$$

Now if we pre-multiply by $\mathbf{x}_2^T$ and rearrange we obtain

$$(1-\lambda+\lambda^2) = \lambda\frac{\mathbf{x}_2^TA_{22}\mathbf{x}_2}{\mathbf{x}_2^TN\mathbf{x}_2} = \lambda\sigma,$$

where $\sigma := \mathbf{x}_2^TA_{22}\mathbf{x}_2/\mathbf{x}_2^TN\mathbf{x}_2$. It follows that

$$\lambda = \frac{1+\sigma \pm \sqrt{\sigma^2+2\sigma-3}}{2}. \tag{4.5}$$

Note that, since σ is a Rayleigh quotient, we have that

$$0 < \lambda_{\min}(N^{-1}A_{22}) \leq \sigma \leq \lambda_{\max}(N^{-1}A_{22}).$$

The result follows. □

If the smallest eigenvalue of $N^{-1}A_{22}$ is greater than unity, the eigenvalues λ of (4.1) are all real. Since the eigenvalues of $N^{-1}A_{22}$ are positive, the real part of λ must be bounded below by $\frac{1}{2}$.

This approach is particularly attractive if A_{22} is zero, as we then have three distinct eigenvalues, $\{1, \frac{1\pm\sqrt{3}i}{2}\}$. An example of where this structure arises naturally would be in the interior point method for linear programs [53], where A_{22} approaches zero at convergence.

We can say something about the quality of this preconditioner if N is replaced by an approximation, $\tilde{N}$.

COROLLARY 4.2. *Let $\tilde{N}$ be a symmetric positive definite approximation to N such that*

$$\lambda(\tilde{N}^{-1}N) \in [\mu_{\min}, \mu_{\max}].$$

Then the generalized eigenvalues satisfying

$$\begin{bmatrix} A_{11} & A_{12} & B_1^T \\ A_{21} & A_{22} & B_2^T \\ B_1 & B_2 & 0 \end{bmatrix}\begin{bmatrix}\mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{y}\end{bmatrix} = \lambda\begin{bmatrix} A_{11} & 0 & B_1^T \\ 0 & \tilde{N} & 0 \\ B_1 & 0 & 0 \end{bmatrix}\begin{bmatrix}\mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{y}\end{bmatrix}, \tag{4.6}$$

are such that $\lambda = 1$ or

$$\begin{aligned}\lambda \in \Bigg[&\left(1+\sigma_{\min}/\mu_{\min} \pm \sqrt{(\sigma_{\min}/\mu_{\min})^2 + (2\sigma_{\min}-4)/\mu_{\min}+1}\right)/2, \\ &\left(1+\sigma_{\max}/\mu_{\max} \pm \sqrt{(\sigma_{\max}/\mu_{\max})^2 + (2\sigma_{\max}-4)/\mu_{\max}+1}\right)/2\Bigg].\end{aligned}$$

Proof. The proof of Theorem 4.1 goes through until (4.4), which here becomes

$$N\mathbf{x}_2 - \lambda A_{22}\mathbf{x}_2 = \lambda(1-\lambda)\tilde{N}\mathbf{x}_2.$$

Pre-multiplying by $\mathbf{x}_2^T$ and setting $\mu := \frac{\mathbf{x}^T\tilde{N}\mathbf{x}_2}{\mathbf{x}^T N\mathbf{x}_2}$ we get

$$\mathbf{x}_2^T N\mathbf{x}_2 - \lambda\mathbf{x}_2^T A_{22}\mathbf{x}_2 = \lambda(1-\lambda)\mu\mathbf{x}_2^T N\mathbf{x}_2,$$

and hence

$$\lambda^2 - (1+\sigma/\mu)\lambda + 1/\mu = 0,$$

and the result follows as in the proof of Theorem 4.1. ∎

4.2. An alternative preconditioner. Now we consider the preconditioner formed from the second and third terms in the factorization (2.13),

$$\mathcal{P}_2 := \begin{bmatrix} A_{11} & A_{12} & B_1^T \\ 0 & \tilde{N} & 0 \\ B_1 & B_2 & 0 \end{bmatrix}.$$

This corresponds to the block upper-triangular preconditioner in the standard factorizations. We have the following result about the eigenvalues:

THEOREM 4.3. *Let $\tilde{N}$ be an approximation to $N = Z_f^T A Z_f$. Consider the generalized eigenvalue problem*

$$\begin{bmatrix} A_{11} & A_{12} & B_1^T \\ A_{21} & A_{22} & B_2^T \\ B_1 & B_2 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{y} \end{bmatrix} = \lambda \begin{bmatrix} A_{11} & A_{12} & B_1^T \\ 0 & \tilde{N} & 0 \\ B_1 & B_2 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{y} \end{bmatrix}. \tag{4.7}$$

Then either $\lambda = 1$, or λ satisfies $N\mathbf{x} = \lambda\tilde{N}\mathbf{x}$.

Proof. Taking the last equation in (4.7) and rearranging we get that either $\lambda = 1$, or

$$\mathbf{x}_1 = -B_1^{-1}B_2\mathbf{x}_2. \tag{4.8}$$

Substituting this into the first equation of (4.7) and rearranging gives

$$\mathbf{y} = B_1^{-T}(A_{11}B_1^{-1}B_2 - A_{12})\mathbf{x}_2. \tag{4.9}$$

Finally, taking the second equation of (4.7), rearranging, and substituting in (4.8) and (4.9) we have

$$-A_{21}B_1^{-1}B_2\mathbf{x}_2 + A_{22}\mathbf{x}_2 + B_2^T B_1^{-1}A_{11}B_1^{-1}B_2\mathbf{x}_2 - B_2^T B_1^{-1}A_{12}\mathbf{x}_2 = \lambda\tilde{N}\mathbf{x}_2$$

The coefficient matrices on the left hand side are equal to N, giving the result. ∎

4.3. Discussion. The two preconditioners described above are the result of thinking about the null-space method in terms of a matrix factorization. The preconditioner $\mathcal{P}_2$ is particularly promising. It has the drawback that it is a non-symmetric preconditioner for a symmetric problem, but it only requires two more matrix-vector products to apply compared with $\mathcal{P}_1$, and for that we get eigenvalue clustering that is as good as possible.

For all preconditioners derived from a null-space factorization it is necessary to find a well-conditioned non-singular block, B_1. This approach is therefore most promising for systems where such a block is easy to find—e.g., when solving $\mathcal{F}-$matrices, or PDE-constrained optimization problems. If we have to do a factorization of B^T to find B_1, then such a preconditioner may offer an advantage in fields where many systems have to be solved with the same B block, with A changing; an important example that generates linear systems with this structure is the interior point method in optimization [53].

It may also be of interest for problems in, e.g., optimization that naturally lend themselves to the null-space method, but for which forming the exact reduced Hessian is prohibitive; a perfect example would be PDE-constrained optimization. With both $\mathcal{P}_1$ and $\mathcal{P}_2$ there is no need to use an inexact matrix-vector multiply with N, as is often done when solving systems with N using a Krylov method, and more accurate solutions could be obtained.

5. Conclusion. The null-space method for solving systems of the form (1.1) can be thought of in terms of matrix factorizations. We have described a number of such factorizations, and have highlighted relationships between them.

A direct solver based on such a factorization is potentially attractive as it avoids the need for numerical pivoting and facilitates the exploitation of parallelism. In the case of a symmetric, positive semi-definite $(1,1)$ block, provided we have a good basis for the null-space of B, a Cholesky solver can be used on a reduced (but possibly dense) system. In Section 3, we investigated the stability and sparsity properties of various null-space factorizations using a number of academic and practical applications. Our results suggest that these factorizations — particularly the form (2.7) — have the potential to be competitive with a sparse symmetric indefinite solver; further experiments on more and larger systems using efficient implementations are required to better understand where to recommend a null-space factorization. For the special case of $\mathcal{F}-$matrices, there is theory to justify stability of the factorization and the B_1 matrix can be found trivially.

Finally, we have proposed two preconditioners based on incomplete versions of the null-space factorization, and presented eigenvalue bounds. The clustering of the eigenvalues of $\mathcal{P}_2^{-1}\mathcal{A}$ depends entirely and simply on the quality of the approximation to the null-space matrix N. A preconditioner of this form would seem ideally suited to applications in optimization, and this will be the subject of future work.

6. Acknowledgements. We would like to thank Nick Gould, Chen Greif, Sangye Lungten, Jen Pestana, Wil Schilders, and Andy Wathen for reading and commenting on a draft of this report. We are particularly grateful to Michael Saunders for his detailed and constructive feedback and encouragement.

REFERENCES

[1] M. Arioli and G. Manzini, *A null space algorithm for mixed finite-element approximations of Darcy's equation*, Commun. Numer. Meth. Engng **18** (2002), no. 9, 645–657.

[2] C. Ashcraft and R. Grimes, *On direct elimination of constraints in KKT systems*, Sparse Days 2011, CERFACS, Toulouse, 2011.

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

[4] L. T. Biegler, J. Nocedal, and C. Schmid, *A reduced Hessian method for large-scale constrained optimization*, SIAM J. Opt. **5** (1995), no. 2, 314–347.

[5] G. Biros and O. Ghattas, *A Lagrange-Newton-Krylov-Schur method for PDE-constrained optimization*, SIAG/OPT Views and News **11** (2000), no. 2, 1–6.
[6] P. Bochev and D. Ridzal, *Additive operator decomposition and optimization-based reconnection with applications*, Large-Scale Scientific Computing, Springer, 2010, pp. 645–652.
[7] J. H. Bramble and J. E. Pasciak, *A preconditioning technique for indefinite systems resulting from mixed approximations of elliptic problems*, Math. Comp. **50** (1988), 1–17.
[8] R. Bridson, *An ordering method for the direct solution of saddle-point matrices*, 2007, Unpublished preprint available from `http://www.cs.ubc.ca/~rbridson/kktdirect/`.
[9] Z. Bujanovic and D. Kressner, *A block algorithm for computing antitriangular factorizations of symmetric matrices*, Tech. Report, Numerical Analysis and High-Performance Computing, EPFL, Switzerland, 2014, Available from `http://anchp.epfl.ch`.
[10] Y. Chen, T. A. Davis, W. W. Hager, and S. Rajamanickam, *Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate*, ACM Trans. Math. Softw. **35** (2008), Article 22, 14 pages.
[11] T. A. Davis, *Algorithm 915, SuiteSparseQR: Multifrontal multithreaded rank-revealing sparse QR factorization*, ACM Trans. Math. Softw. **38** (2011), no. 1, 8:1–8:22.
[12] A. C. de Niet and F. W. Wubs, *Numerically stable LDL^T-factorization of $\mathcal{F}$-type saddle point matrices*, IMA J. Numer. Anal. **29** (2009), no. 1, 208–234.
[13] H. S. Dollar, N. I. M. Gould, W. H. A. Schilders, and A. J. Wathen, *Implicit-factorization preconditioning and iterative solvers for regularized saddle-point systems*, SIAM J. Matrix Anal. Applics **28** (2006), no. 1, 170–189.
[14] H. S. Dollar and A. J. Wathen, *Approximate factorization constraint preconditioners for saddle-point matrices*, SIAM J. Sci. Comput. **27** (2006), no. 5, 1555–1572.
[15] I. S. Duff, *MA57—a code for the solution of sparse symmetric definite and indefinite systems*, ACM Trans. Math. Softw. **30** (2004), no. 2, 118–144.
[16] H. Elman, D. Silvester, and A. Wathen, *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics*, Numerical Mathematics and Scientific Computation, Oxford University Press, 2005.
[17] R. Fletcher, *Practical Methods of Optimization*, John Wiley & Sons, 2013.
[18] R. Fletcher and T. Johnson, *On the stability of null-space methods for KKT systems*, SIAM J. Matrix Anal. Applics **18** (1997), no. 4, 938–958.
[19] A. Forsgren and W. Murray, *Newton methods for large-scale linear equality-constrained minimization*, SIAM J. Matrix Anal. Applics **14** (1993), no. 2, 560–587.
[20] P. E. Gill and W. Murray, *Newton-type methods for unconstrained and linearly constrained optimization*, Math. Program. **7** (1974), no. 1, 311–350.
[21] P. E. Gill, W. Murray, and M. A. Saunders, *SNOPT: An SQP algorithm for large-scale constrained optimization*, SIAM Rev. **47** (2005), no. 1, 99–131.
[22] N. I. M. Gould, *On modified factorizations for large-scale linearly constrained optimization*, SIAM J. Opt. **9** (1999), no. 4, 1041–1063.
[23] N. I. M. Gould, M. E. Hribar, and J. Nocedal, *On the solution of equality constrained quadratic programming problems arising in optimization*, SIAM J. Sci. Comput. **23** (2001), 1376–1395.
[24] N. I. M. Gould, D. Orban, and T. Rees, *Projected Krylov methods for saddle-point systems*, Tech. Report RAL-P-2013-006, STFC Rutherford Appleton Laboratory, 2013.
[25] N. I. M. Gould, D. Orban, and Ph. L. Toint, *CUTEst: a constrained and unconstrained testing environment with safe threads*, Tech. Report RAL-TR-2013-005, STFC Rutherford Appleton Laboratory, 2013.
[26] A. Greenbaum, V. Pták, and Z. Strakoš, *Any nonincreasing convergence curve is possible for GMRES*, SIAM J. Matrix Anal. Applics **17** (1996), 465–469.
[27] N. Henderson, *Mex interface to LUSOL*, 2014, `http://github.com/nwh/lusol`.
[28] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed., SIAM, 2002.
[29] J. D. Hogg, J. K. Reid, and J. A. Scott, *Design of a multicore sparse Cholesky factorization using DAGs*, SIAM J. Sci. Comput. **32** (2010), 3627–3649.
[30] *HSL. A collection of Fortran codes for large-scale scientific computation*, 2013, `http://www.hsl.rl.ac.uk`.
[31] Q. Jiang and G. Geng, *A reduced-space interior point method for transient stability constrained optimal power flow*, IEEE Trans. Power Systems **25** (2010), no. 3, 1232–1240.
[32] C. Keller, N. I. M. Gould, and A. J. Wathen, *Constraint preconditioning for indefinite linear systems*, SIAM J. Matrix Anal. Applics **21** (2000), 1300–1317.
[33] H. K. Kim, M. Flexman, D. J. Yamashiro, J. J. Kandel, and A. H. Hielscher, *PDE-constrained multispectral imaging of tissue chromophores with the equation of radiative transfer*, Biomedical Optics Express **1** (2010), 812–824.

[34] S. Lungten, W. H. A. Schilders, and J. M. L. Maubach, *Sparse block factorization of saddle point matrices*, Tech. Report CASA-Report 14-23, TU Eindhoven, 2014.
[35] K.-A. Mardal and R. Winther, *Preconditioning discretizations of systems of partial differential equations*, Numer. Lin. Alg. Applics **18** (2011), no. 1, 1–40.
[36] J. Maryska, M. Rozložník, and M. Tůma, *Schur complement systems in the mixed-hybrid finite element approximation of the potential fluid flow problem*, SIAM J. Sci. Comput. **22** (2000), no. 2, 704–723.
[37] N. Mastronardi and P. Van Dooren, *The antitriangular factorization of symmetric matrices*, SIAM J. Matrix Anal. Applics **34** (2013), no. 1, 173–196.
[38] N. Mastronardi and P. Van Dooren, *An algorithm for solving the indefinite least squares problem with equality constraints*, BIT **54** (2014), 201–218.
[39] J. Maubach and W. Schilders, *Micro- and macro-block factorizations for regularized saddle point systems*, Tech. Report CASA-Report 12-07, TU Eindhoven, 2012.
[40] P. P. Moghaddam and F. J. Herrmann, *Randomized full-waveform inversion: a dimenstionality-reduction approach*, SEG Technical Program Expanded Abstracts, vol. 29, 2010, pp. 977–982.
[41] M. F. Murphy, G. H. Golub, and A. J. Wathen, *A note on preconditioning for indefinite linear systems*, SIAM J. Sci. Comput. **21** (2000), no. 6, 1969–1972.
[42] W. Murray, *An algorithm for finding a local minimum of an indefinite quadratic program*, Tech. report, NPL NAC 1, 1971.
[43] J. Pestana and A. J. Wathen, *On the choice of preconditioner for minimum residual methods for non-Hermitian matrices*, J. Comp. Applied Maths **249** (2013), 57–68.
[44] ———, *The antitriangular factorization of saddle point matrices*, SIAM J. Matrix Anal. Applics **35** (2014), no. 2, 339–353.
[45] A. Pinar, E. Chow, and A. Pothen, *Combinatorial algorithms for computing column space bases that have sparse inverses*, Elec. Trans. Numer. Anal. **22** (2006), 122–145.
[46] T. Rees, H. S. Dollar, and A. J. Wathen, *Optimal solvers for PDE-constrained optimization*, SIAM J. Sci. Comput. **32** (2010), 271–298.
[47] M. Saunders, *LUSOL: A basis package for constrained optimization*, SOL, Stanford University (2013), `http://web.stanford.edu/group/SOL/software/lusol/`.
[48] W. H. A. Schilders, *Solution of indefinite linear systems using an LQ decomposition for the linear constraints*, Lin. Alg. Applics **431** (2009), 381–395.
[49] J. A Scott, *A note on a simple constrained ordering for saddle-point systems*, Tech. Report RAL-TR-2009-007, STFC Rutherford Appleton Laboratory, 2009.
[50] J. A. Scott and M. Tůma, *On signed incomplete Cholesky factorization preconditioners for saddle-point systems*, Tech. Report RAL-P-2014-003, STFC Rutherford Appleton Laboratory, 2014.
[51] M. Tůma, *A note on the LDL^T decomposition of matrices from saddle-point problems*, SIAM J. Matrix Anal. Applics. **23** (2002), 903–915.
[52] S. A. Vavasis, *Stable numerical algorithms for equilibrium systems*, SIAM J. Matrix Anal. Applics **15** (1994), 1108–1131.
[53] S. J. Wright, *Primal-dual Interior-point Methods*, vol. 54, SIAM, Philadelphia, 1997.