NULL-SPACE PRECONDITIONERS FOR SADDLE POINT SYSTEMS

JENNIFER PESTANA,∗ TYRONE REES†

Abstract. The null-space method is a technique that has been used for many years to reduce a saddle point system to a smaller, easier to solve, symmetric positive-definite system. This method can be understood as a block factorization of the system. Here we explore the use of preconditioners based on incomplete versions of a particular null-space factorization, and compare their performance with the equivalent Schur-complement based preconditioners. We also describe how to apply the non-symmetric preconditioners proposed using the conjugate gradient method (CG) with a non-standard inner product. This requires an exact solve with the (1,1) block, and the resulting algorithm is applicable in other cases where Bramble-Pasciak CG is used. We verify the efficiency of the newly proposed preconditioners on a number of test cases from a range of applications.

AMS subject classifications. 65F08, 65F10, 65F50

Key words. Preconditioning, saddle point systems, null-space method

1. Introduction. We consider the fast solution of saddle point systems of the form

\[ \begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix} = b, \]

where \( A \in \mathbb{R}^{n \times n}, B \in \mathbb{R}^{m \times n}, m \leq n \). We suppose that \( B \) is of full rank, and that \( A \) is symmetric positive semi-definite, and positive definite on the kernel of \( B \).

Solving systems with this structure is a vital component in numerous scientific computing algorithms; for example, such systems arise naturally in constrained optimization problems [7, 13, 15, 25], Stokes and Navier-Stokes equations in fluid mechanics [5, 9, 16], time-harmonic Maxwell equations [26, 32], and the application of Kirchhoff’s laws in circuit simulation [46, 50]. For an overview of solution methods, see the survey article of Benzi, Golub and Liesen [4], and the references therein.

The null-space method is a technique for solving systems of the form (1.1). This method requires a particular solution \( \hat{x} \in \mathbb{R}^n \) such that \( B \hat{x} = g \), and a matrix \( Z \) whose columns span the null-space of \( B \). Then, since \( x = Zx_n + \hat{x} \), we can solve (1.1) by first finding \( x_n \) from

\[ Z^T A Z x_n = Z^T (f - A \hat{x}), \]

and then recovering \( y \) from the overdetermined system \( B^T y = f - A \hat{x} \); see, e.g., [4, Chapter 6] for more details. There exist many methods for obtaining a null-space matrix \( Z \) but a common choice is the fundamental basis,

\[ Z_f := \begin{bmatrix} -B_1^{-1} B_2 \\ I \end{bmatrix}, \]

where \( B = [B_1 B_2] \) and, without loss of generality, \( B_1 \in \mathbb{R}^{m \times m} \) is nonsingular. Note that because the rank of \( B \) is \( m \), we can always permute \( B \) to obtain an invertible \( B_1 \).

∗University of Strathclyde, jennifer.pestana@strath.ac.uk. This work was completed at the University of Manchester, supported by Engineering and Physical Sciences Research Council grant EP/I005293.

†STFC Rutherford Appleton Laboratory, Chilton, Didcot, UK, tyrone.rees@stfc.ac.uk. This author was supported by Engineering and Physical Sciences Research Council grant EP/1013067/1.
Even for sparse \( A \) the symmetric positive definite matrix \( Z^T A Z \) is often dense, and forming it could be costly. For larger problems, therefore, it may be better to solve (1.1) rather than (1.2). It is possible to write down a number of matrix factorizations that are equivalent to the null-space method—recently Rees and Scott gave an overview [44]. We can obtain one particular factorization of \( A \) which falls into this category by taking a \( 3 \times 3 \) blocking of \( A \) as

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

where \( B_1 \in \mathbb{R}^{m \times m} \) is, as in (1.3), assumed to be nonsingular. Consider a permuted version of \( A \),

\[
\hat{A} := \Pi A \Pi^T = \begin{bmatrix} A_{11} & B_1^T \\ B_1 & 0 \\ A_{21} & B_2^T \end{bmatrix} =: \begin{bmatrix} \hat{A} \\ \hat{B} \end{bmatrix}, \quad \Pi = \begin{bmatrix} I_m & 0 \\ I_m & I_{n-m} \end{bmatrix},
\]

(1.4)

where \( I_m \) is the identity matrix of dimension \( m \). The matrix \( \hat{A} \) is invertible with inverse

\[
\hat{A}^{-1} = \begin{bmatrix} 0 & B_1^{-1} \\ B_1^{-T} & -B_1^{-T} A_{11} B_1^{-1} \end{bmatrix}.
\]

Therefore we can apply the standard block-LDL^T factorization for saddle point systems [4, Equation (3.1)], obtaining

\[
\hat{A} = \begin{bmatrix} I & 0 \\ \hat{B} \hat{A}^{-1} & I \end{bmatrix} \begin{bmatrix} \hat{A} & 0 \\ 0 & A_{22} - \hat{B} \hat{A}^{-1} \hat{B}^T \end{bmatrix} \begin{bmatrix} I & \hat{A}^{-1} \hat{B}^T \\ 0 & I \end{bmatrix}.
\]

(1.5)

This factorization has a connection with standard Schur complement preconditioners and the range-space method. However, because of the permutation, it is also closely related to the null-space method with the fundamental basis since, using (1.3), we can re-write (1.5) as

\[
\hat{A} = \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ B_2^T B_1^{-T} & X B_1^{-1} & I \end{bmatrix} \begin{bmatrix} A_{11} & B_1^T & 0 \\ B_1 & 0 & 0 \\ 0 & 0 & N \end{bmatrix} \begin{bmatrix} I & 0 & B_1^{-1} B_2 \\ 0 & I & B_1^{-T} X^T \end{bmatrix},
\]

(1.6)

where

\[
X := Z_f^T \begin{bmatrix} A_{11} \\ A_{21} \end{bmatrix} \quad \text{and} \quad N := Z_f^T A Z_f = A_{22} - \hat{B} \hat{A}^{-1} \hat{B}^T.
\]

(1.7)

Our assumption that \( A \) is positive definite on the null-space of \( B \) means that the null-space matrix \( N \) is symmetric positive definite. The approach in Section 2 of Rees and Scott [44] applied here shows that solving \( \hat{A}(\Pi \mathbf{w}) = \Pi \mathbf{b} \) is equivalent to the null-space method with the fundamental null-space (1.3) and particular solution

\[
\hat{x} = \begin{bmatrix} B_1^{-1} g \\ 0 \end{bmatrix}.
\]

(1.8)
Applying the inverse permutations Π and Πᵀ in (1.4) to ˆA in (1.6) we can easily recover A.

One interpretation of the null-space method is therefore that the null-space matrix ZᵀfAZf is equivalent to the Schur complement of the sub-block ˆA in ˆA. Note that when the block A is invertible, the null-space matrix is equivalent to the Schur complement of the dual saddle point system

\[
\begin{bmatrix}
A^{-1} & Z \\
Z^T & 0
\end{bmatrix}
\begin{bmatrix}
w \\
x_n
\end{bmatrix} =
\begin{bmatrix}
-\hat{x} \\
-Z^Tf
\end{bmatrix}
\]

of (1.1) [4, page 32].

In this paper we present four preconditioners based on incomplete or approximate versions of the factorization (1.6). Since these preconditioners are intimately related to the Schur-complement decomposition we can apply them using conjugate gradients in a non-standard inner product [10]. However, the usual application of such preconditioners requires certain quantities to be symmetric positive definite, and this is typically attained by scaling the blocks appropriately. In our case this is not possible without destroying the structure that we exploit, since we assume that we can solve with a certain submatrix of A exactly. Accordingly, we extend the current theory of conjugate-gradients in a non-standard inner product to allow for the case where one of the sub-block solves is exact.

The rest of this paper is as follows. In Section 2, we give an eigen-analysis of the preconditioned systems and show that the eigenvalues of the preconditioned matrices are clustered when a good approximation to the null-space matrix (1.7) is known. We describe the non-standard inner product CG method in Section 3, and describe how our constraint preconditioner can be used within a projected Krylov subspace method. In Section 4, we compare our preconditioners to standard Schur-complement based methods based on the factorization

\[
\begin{bmatrix}
A & B^T \\
B & -C
\end{bmatrix} =
\begin{bmatrix}
I & 0 \\
BA^{-1} & I
\end{bmatrix}
\begin{bmatrix}
A & 0 \\
I & 0
\end{bmatrix}
\begin{bmatrix}
I & A^{-1}B^T \\
0 & I
\end{bmatrix},
\]

(1.9)

where S = C + BA⁻¹Bᵀ is the Schur complement of A. Note that this factorization can only be applied if A is invertible, whereas the null-space method may be applied even when A is singular. The problems we consider range from academic to practical, and illustrate the merits and limitations of our preconditioners. We give some conclusions in Section 5.

2. Null-space preconditioners. Taking incomplete versions of the block LDLᵀ factorization (1.9) has proved to be an effective way of constructing preconditioners for saddle point problems—see, e.g., [7, 16, 27, 30, 31, 43, 51]. The key component of such methods is an approximation of the matrix S, which is often dense. In the following we present, and give theory for, preconditioners based on the alternative decomposition (1.5). In particular, we use the null-space decomposition

\[
A =
\begin{bmatrix}
I & 0 & 0 \\
BᵀB₁⁻¹ & I & XB₁⁻¹ \\
0 & 0 & I
\end{bmatrix}
\begin{bmatrix}
A_{11} & 0 & Bᵀ₁ \\
0 & N & 0 \\
B₁ & 0 & 0
\end{bmatrix}
\begin{bmatrix}
I & B₁⁻¹B₂ & 0 \\
0 & I & 0 \\
0 & B₁⁻¹ᵀXᵀ & I
\end{bmatrix},
\]

(2.1)

as the basis of our preconditioners; see [4, Equation 10.35], [44]. We replace N by a symmetric positive definite approximation ˜N, and possibly drop one or both of L and Lᵀ.
2.1. The central-null preconditioner. First, we consider the preconditioner formed by dropping both the $L$ and $L^T$ terms from the factorization (2.1),

$$ P_{cn} = \begin{bmatrix} A_{11} & 0 & B_1^T \\ 0 & \tilde{N} & 0 \\ B_1 & 0 & 0 \end{bmatrix}, $$

where $\tilde{N} \approx N = Z_f^T A Z_f$. This corresponds to the block diagonal preconditioner in the decomposition (1.9), because $\Pi P_{cn} \Pi^T$ is block diagonal. Accordingly, the cost of applying $P_{cn}$ is a solve with each of $B_1, B_1^T$ and $\tilde{N}$, as well as a matrix-vector product with $A_{11}$. Note that in our numerical experiments we use the matrix $P_{cn}$ rather than $\Pi P_{cn} \Pi^T$, but either could be implemented in practice. This comment also applies to the other preconditioners we introduce.

First, we give an eigen-analysis of the preconditioned system.

**Theorem 2.1.** Let $\tilde{N}$ be a symmetric positive definite approximation to $N$. Then the generalized eigenvalues $\lambda$ of

$$ \begin{bmatrix} A_{11} & A_{12} & B_1^T \\ A_{21} & A_{22} & B_2^T \\ B_1 & B_2 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ y \end{bmatrix} = \lambda \begin{bmatrix} A_{11} & 0 & B_1^T \\ 0 & \tilde{N} & 0 \\ B_1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ y \end{bmatrix}, \tag{2.2} $$

satisfy $\lambda = 1$, or

$$ \lambda = \frac{\mu + \sigma \pm \sqrt{(\sigma + \mu)^2 - 4\mu}}{2\mu}, \tag{2.3} $$

where $\sigma = y^T A_{22} y/y^T N y$ and $\mu = y^T \tilde{N} y/y^T N y$.

**Proof.** Since $\Pi$ from (1.4) is orthogonal, the eigenvalues of $P_{cn}^{-1} A$ are the same as those of $\Pi P_{cn}^{-1} A \Pi^T = (\Pi P_{cn} \Pi^T)^{-1}(\Pi A \Pi^T)$, and therefore the eigenvalues needed are those of the generalized eigenvalue problem

$$ \begin{bmatrix} \tilde{A} & \tilde{B}^T \\ \tilde{B} & A_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \lambda \begin{bmatrix} \tilde{A} \\ \tilde{N} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}. \tag{2.4} $$

The first block row of (2.4) gives $\tilde{B}^T y = (\lambda - 1)\tilde{A} x$, which implies that $\lambda = 1$ or $x = \tilde{A}^{-1} \tilde{B}^T y/(\lambda - 1)$. The second block row of (2.4) gives $\tilde{B} x + A_{22} y = \lambda \tilde{N} y$. If we assume that $\lambda \neq 1$ then substituting for $x$ gives that

$$ \tilde{B} \tilde{A}^{-1} \tilde{B}^T y + (\lambda - 1) A_{22} y = \lambda (\lambda - 1) \tilde{N} y. $$

Using (1.7) we have that $(\lambda A_{22} - N) y = \lambda (\lambda - 1) \tilde{N} y$. Premultiplying by $y^T$ and setting $\mu := y^T \tilde{N} y/y^T N y$ we get $y^T N y - \lambda y^T A_{22} y = \lambda(1 - \lambda) \mu y^T N y$, and hence $\lambda^2 - (1 + \sigma/\mu) \lambda + 1/\mu = 0$, where $\sigma = y^T A_{22} y/y^T N y$.

Thus,

$$ \lambda = \frac{1 + \sigma/\mu \pm \sqrt{(\sigma/\mu)^2 + (2\sigma - 4)/\mu + 1}}{2} = \frac{\mu + \sigma \pm \sqrt{(\sigma + \mu)^2 - 4\mu}}{2\mu}, $$

as required. \qed

We now present a few results that give a better understanding of the behaviour of these eigenvalues.
Corollary 2.2. Suppose that $\lambda$ is a real eigenvalue of the generalized eigenvalue problem (2.2). Then

$$\frac{1}{(\mu + \sigma)_{\text{max}}} \leq \lambda < 1 + \lambda_{\text{max}}(\tilde{N}^{-1}A_{22})$$

where $0 \leq \lambda_{\text{max}}(\tilde{N}^{-1}A_{22})$ is the largest eigenvalue of $\tilde{N}^{-1}A_{22}$ and $(\mu + \sigma)_{\text{max}}$ is the largest value of $\mu + \sigma$.

Proof. For $\lambda$ to be real we must have that $(\mu + \sigma)^2 \geq 4\mu$. The larger of the eigenvalues in (2.3), $\lambda_+$, satisfies

$$\lambda_+ = \frac{1}{2\mu} \left( \mu + \sigma + \sqrt{(\mu + \sigma)^2 - 4\mu} \right) < \frac{1}{2\mu} \left( \mu + \sigma + \sqrt{(\mu + \sigma)^2} \right) = 1 + \frac{\sigma}{\mu}.$$ 

Now, since $\sigma/\mu = y^T A_{22} y / y^T \tilde{N} y \leq \lambda_{\text{max}}(\tilde{N}^{-1}A_{22})$ we obtain the upper bound.

Now consider the smaller eigenvalue, $\lambda_-$. Note that $\lambda_- = (\gamma - \sqrt{\gamma^2 - \delta})/2$, where $\gamma = 1 + \sigma/\mu$, $\delta = 4/\mu$ and $\delta \leq \gamma^2$. For any $x \in [0,1]$, $1 - \sqrt{1-x} \geq x/2$, and so

$$\lambda_- \geq \frac{1}{4} \frac{\delta}{\gamma} = \frac{1}{\mu + \sigma} \geq \frac{1}{(\mu + \sigma)_{\text{max}}}.$$ 

It is clear from the lower bound that the real eigenvalues are positive.

Remark 2.3. In the ideal case where $\mu_{\text{min}} = \mu_{\text{max}} = 1$, which corresponds to $\tilde{N} = N$, the lower bound becomes $\lambda_- > 1/1+\sigma_{\text{max}}$. Furthermore, in this case the eigenvalues are all real if $\sigma_{\text{min}} \geq 1$.

Corollary 2.4. The complex eigenvalues of (2.2) with nonzero imaginary part satisfy

$$\frac{1}{2} \leq \text{Re}(\lambda) < \frac{1}{\sqrt{\mu_{\text{min}}}}, \quad |\text{Im}(\lambda)| < \frac{1}{\sqrt{\mu_{\text{min}}}}.$$ 

Proof. Since the eigenvalues of $N^{-1}A_{22}$ are nonnegative, any complex eigenvalue $\lambda$ satisfies $\text{Re}(\lambda) \geq \frac{1}{2}$. Furthermore, $\text{Re}(\lambda) \leq 1/2(1 + \sigma_{\text{max}}/\mu_{\text{min}}) < 1/\sqrt{\mu_{\text{min}}}$, with the last step holding since $\mu + \sigma < 2\sqrt{\mu} \Rightarrow (\mu + \sigma)/2\mu < 1/\sqrt{\mu_{\text{min}}}$. Now, the imaginary part satisfies

$$|\text{Im}(\lambda)|^2 = \frac{(\sigma + \mu)^2 - 4\mu}{4\mu^2} < \frac{(\sigma + \mu)^2}{4\mu^2} < \frac{1}{\mu_{\text{min}}}.$$ 

Note that the bound on the real part only holds for eigenvalues with a non-trivial imaginary part. For real eigenvalues there may be some $\lambda < 0.5$ if any eigenvalue of $N^{-1}A_{22}$ is larger than 1.5. It is also worth remarking that complex eigenvalues can be bounded independently of $\sigma$.

One final special case we wish to highlight is when $A_{22}$ is zero and $\mu_{\text{min}} = \mu_{\text{max}} = 1$. Here the central-null preconditioned matrix has only three distinct eigenvalues, \{1, 1+\sqrt{3}\}, and this guarantees fast convergence of certain Krylov subspace methods. (We could also obtain these same three eigenvalues in this special case by Schur-complement based arguments, following approaches found in, e.g., [18, 38].) An example of where this structure arises naturally is in the interior point method.
Table 2.1: Extreme values of $\mu$ and $\sigma$ for $\tilde{N} = I$ and for incomplete Cholesky (IC) factorizations of $N$. The IC drop tolerance is given in parentheses.

<table>
<thead>
<tr>
<th></th>
<th>AUG3CD</th>
<th>MOSARQP1</th>
<th>STCQP2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$I$</td>
<td>$I$</td>
<td>$I$</td>
</tr>
<tr>
<td>$\mu$</td>
<td>[0.0029, 0.5]</td>
<td>[0.088, 0.47]</td>
<td>[0.0019, 0.33]</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>[0.0058, 1]</td>
<td>[0.21, 1]</td>
<td>[0.23, 4.7]</td>
</tr>
</tbody>
</table>

for linear programs [54], where $A_{22}$ approaches zero at convergence. Note that when $\mu_{\min} \approx 1$ and $\mu_{\max} \approx 1$, that is, when we have a good approximation $\tilde{N}$ to $N$, the above conclusions for the ideal case are approximately satisfied.

To conclude this section we examine the bounds in Corollaries 2.2 and 2.4 for the matrices AUG3DC, MOSARQP1 and STCQP2 from the CUTEst test set (see Section 4.2 for details). We first tabulate $\mu$ and $\sigma$ (see Table 2.1) and find that, as expected, $\mu_{\min}$ and $\mu_{\max}$ are close to 1 when $\tilde{N}$ is a good approximation of $N$, and are often far from 1 otherwise. Note that if $\tilde{N} = N$, then $\mu_{\min} = \mu_{\max} = 1$. In contrast, $\sigma$ depends only on the properties of $A$. For STCQP2, $B_2 = 0$, and so $A_{22} = N$ and $\sigma_{\min} = \sigma_{\max} = 1$. However, $\sigma_{\min} \ll 1$ for AUG3DC or MOSARQP1. For all three matrices $\sigma_{\max} = 1$ although this is not guaranteed in general.

In Figures 2.1–2.3 we examine the effect of $\mu$ and $\sigma$ on the eigenvalues of $P_{cn}^{-1}A$ and the bounds in Corollaries 2.2 and 2.4. We first note that real eigenvalues seem to cluster near 1 when $\mu_{\min}$ and $\mu_{\max}$ are close to 1, but may be far from 1 when $\tilde{N}$ is not a good approximation. For all three matrices, the lower bound on real eigenvalues in Corollary 2.2 is about half the value of the smallest real eigenvalue, except when $\mu_{\min}$ is much smaller than 1, i.e., when $\tilde{N}$ does not approximate $N$ well. For AUG3DC the upper bound is also approximately twice as big as the largest real eigenvalue, while for MOSARQP1 it is up to four times as big. However, for STCQP2 (for which $\sigma_{\min} = \sigma_{\max} = 1$) the upper bound is a fairly good approximation, and is particularly descriptive when $\tilde{N}$ is not as good an approximation of $N$.

Our bounds for complex eigenvalues also indicate that a good approximation of $N$ ensures that the complex eigenvalues are bounded in a small region away from the origin. For both AUG3DC and MOSARQP1, the bounds on the real part of complex eigenvalues in Corollary 2.4 seems most descriptive when $\mu_{\min}$ is close to 1, and the upper bound is particularly poor when $\tilde{N} = I$. On the other hand, the upper bound on the imaginary part of complex eigenvalues seems to be fairly descriptive even when $\tilde{N}$ is a poor approximation to $N$. (Note that all eigenvalues of STCQP2 are real since $A_{22} = N$.)

Overall, these results indicate that for these matrices the quality of $\tilde{N}$ has the greatest effect on the location of eigenvalues, and that our bounds are reasonably descriptive when $\tilde{N}$ is a reasonable approximation of $N$. However, the distribution of eigenvalues within the region prescribed by the bounds is more difficult to ascertain and is influenced by the eigenvalues of $N^{-1}A_{22}$. This eigenvalue distribution can affect the convergence of Krylov methods for the central-null preconditioner, as we will see in Section 4.
2.2. The lower-null and upper-null preconditioners. Next, we drop the $L^T$-term in (2.1) to form the preconditioner

$$
\mathcal{P}_{ln} := \begin{bmatrix}
A_{11} & 0 & B_1^T \\
A_{21} & \tilde{N} & B_2^T \\
B_1 & 0 & 0
\end{bmatrix}.
$$

Since $\Pi\mathcal{P}_{ln}\Pi^T$ corresponds to a block lower-triangular preconditioner we refer to this as the lower-null preconditioner. Indeed, this preconditioner is ‘psychologically block-lower triangular’, in that we can easily identify blocks with which we can solve this system using a substitution method. As well as solves with $B_1$, $B_1^T$ and $\tilde{N}$, and a matrix-vector product with $A_{11}$, we require matrix-vector products with $A_{21}$ and $B_2^T$.

The following result holds for the eigenvalues.

**Theorem 2.5.** Let $\tilde{N}$ be an invertible approximation to $N$. Consider the generalized eigenvalue problem $\mathcal{P}_{ln}z = \lambda Az$. Then $\lambda = 1$ or $\lambda$ is an eigenvalue of $\tilde{N}^{-1}N$.

**Proof.** As in the proof of Theorem 2.1, we use the fact that the required eigenvalues are the same as those of $(\Pi\mathcal{P}_{ln}\Pi^T)^{-1}\Pi\mathcal{P}_{ln}\Pi^T$. Recalling (1.7) we have that

$$
(\Pi\mathcal{P}_{ln}\Pi^T)^{-1}\Pi\mathcal{P}_{ln}\Pi^T = \begin{bmatrix}
I & \tilde{A}^{-1}\tilde{B} \\
0 & \tilde{N}^{-1}(A_{22} - \tilde{B}\tilde{A}^{-1}\tilde{B}^T)
\end{bmatrix} = \begin{bmatrix}
I & \tilde{A}^{-1}\tilde{B} \\
0 & \tilde{N}^{-1}N
\end{bmatrix}.
$$

The result follows. \[\square\]
Fig. 2.2: MOSARQP1: The eigenvalues (+) of $P_{cn}^{-1}A$ and the bounds in Corollaries 2.2 and 2.4 for different choices of $\tilde{N}$. The black crosses show the bounds on real eigenvalues and the red dashed lines show the bounds on complex eigenvalues with nonzero imaginary part.

An apparent drawback of this preconditioner is that $P_{ln}$ is a non-symmetric preconditioner for a symmetric problem, and hence we have to use a non-symmetric iterative method, such as GMRES [48] or BiCGStab [53]. However, Theorem 2.5 shows that if $N = \tilde{N}$, GMRES applied to (1.1) with preconditioner $P_{ln}$ converges in two steps. When $\tilde{N} \neq N$ the eigenvalues may not tell us everything about convergence [24] although it is commonly observed that tightly clustered eigenvalues do predict convergence of GMRES in non-pathological cases—see, e.g., Pestana and Wathen [40] for a discussion. An additional benefit of using $\tilde{N}$ within a preconditioner for the whole matrix $A$, and not explicitly for the null-space matrix, is that, in principle, all that is needed is (the action of the inverse of) an approximation, $\tilde{N}$. We envisage (as is often the case with standard Schur-complement preconditioning; see, e.g., [16]) applications where $\tilde{N}$ can be applied without explicitly performing the (often costly) procedure of forming $N$. To give two examples, the matrix $N$ is known as the system flexibility matrix in structural analysis [28], and the reduced Hessian matrix in optimization [37, Section 16.5]; in both these cases approximations to this matrix, which will fit into this framework, have been proposed in the literature (see, e.g., [8, 19, 41]). Finally, we note that in some cases $B_2 = 0$, as is the case for the CUTEst test matrix STCQP2 discussed at the end of Section 2.1. In this case $N = A_{22}$, which is trivial to compute. See [34] for the construction of ideal preconditioners for Schur complement methods.
Fig. 2.3: STCQP2: The eigenvalues (+) of $P_{cn}^{-1}A$ and the bounds in Corollaries 2.2 and 2.4 for different choices of $\tilde{N}$. The black crosses show the bounds on real eigenvalues.

If we instead drop the $L$ term from (2.1) we get

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

For reasons analogous to those given above we refer to this as the upper-null preconditioner, and we have the following result.

**Theorem 2.6.** Let $\tilde{N}$ be an invertible approximation to $N$. Consider the generalized eigenvalue problem $P_{un}z = \lambda A z$. Then $\lambda = 1$ or $\lambda$ is an eigenvalue of $\tilde{N}^{-1}N$.

**Proof.** Since $(\Pi P_{un} \Pi^T)^{-1}(\Pi A \Pi^T)$ is similar to $(\Pi A \Pi^T)(\Pi P_{un} \Pi^T)^{-1}$, a similar argument to that in the proof of Theorem 2.5 gives the result. \ 

The preconditioner $P_{un}$ therefore has the same eigenvalues, with the same multiplicity, as $P_{ln}$. In spite of possible effects of non-normality, in practice upper and lower block triangular preconditioners often exhibit similar behaviour (see [39] and the references therein), and this was our experience in the tests reported in Section 4. Additionally, the cost of applying this preconditioner is the same as the cost of applying $P_{ln}$.
### 2.3. A constraint preconditioner

Now consider the preconditioner obtained by replacing $N$ by $\tilde{N}$ in (2.1) without dropping any terms, namely

$$
\mathcal{P}_{\text{con}} = \begin{bmatrix}
A_{11} & 0 & B_1^T \\
A_{21} & \tilde{N} & B_2^T \\
B_1 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
I & B_1^{-1}B_2 & 0 \\
0 & I & 0 \\
0 & B_1^{-T}X^T & I
\end{bmatrix}.
$$

Note that this preconditioner is numerically the same as the preconditioner defined by the GALAHAD [22] subroutine SBLS. The authors are not aware of an eigen-analysis of this preconditioner in the literature.

**Theorem 2.7.** The preconditioner $\mathcal{P}_{\text{con}}$ is a constraint preconditioner, i.e.,

$$
\mathcal{P}_{\text{con}} = \begin{bmatrix}
G & B^T \\
B & 0
\end{bmatrix}
$$

for some matrix $G$.

**Proof.** Direct computation shows that

$$
\Pi\mathcal{P}_{\text{con}}\Pi^T = \begin{bmatrix}
\hat{A} & \hat{B}^T \\
\hat{B} & A_{22} - N + \tilde{N}
\end{bmatrix}
$$

and

$$
\mathcal{P}_{\text{con}} = \begin{bmatrix}
A_{11} & A_{12} & B_1^T \\
A_{21} & A_{22} - N + \tilde{N} & B_2^T \\
B_1 & B_2 & 0
\end{bmatrix}.
$$

We can show the following result about the eigenvalues for the constraint preconditioner here:

**Theorem 2.8.** Let $\tilde{N}$ be an invertible approximation to $N$. Consider the generalized eigenvalue problem $Ax = \lambda\mathcal{P}_{\text{con}}x$. Then $\lambda = 1$ or $\lambda$ is an eigenvalue of $\tilde{N}^{-1}N$.

**Proof.** As in previous sections, we can compute the eigenvalues of $\mathcal{P}_{\text{con}}^{-1}A$ by solving a generalized eigenvalue problem to find the eigenvalues of $(\Pi\mathcal{P}_{\text{con}}\Pi^T)^{-1}(\Pi A\Pi^T)$. In particular, using (2.6), we have that

$$
\begin{bmatrix}
\hat{A} & \hat{B}^T \\
\hat{B} & A_{22}
\end{bmatrix}
\begin{bmatrix}
x \\
y
\end{bmatrix}
= \lambda
\begin{bmatrix}
\hat{A} & \hat{B}^T \\
\hat{B} & A_{22} - N + \tilde{N}
\end{bmatrix}
\begin{bmatrix}
x \\
y
\end{bmatrix}.
$$

The first equation shows that $\lambda = 1$ or that $\hat{A}x + \hat{B}^T y = 0$. Since $\hat{A}$ is invertible, in the latter case we have that $x = -\hat{A}^{-1}\hat{B}^T y$.

The second equation gives that $(1 - \lambda)Bx + (1 - \lambda)A_{22}y = \lambda(\tilde{N} - N)y$. If $\lambda \neq 1$ then after substituting for $x$ we find that

$$
-(1 - \lambda)B\hat{A}^{-1}\hat{B}^T y + (1 - \lambda)A_{22}y = \lambda(\tilde{N} - N)y.
$$

Using (1.7) and simplifying shows that $Ny = \lambda\tilde{N}y$ as required.

Comparison with Theorems 2.5 and 2.6 shows that all three preconditioned matrices $\mathcal{P}_{\text{in}}^{-1}A$, $\mathcal{P}_{\text{un}}^{-1}A$ and $\mathcal{P}_{\text{con}}^{-1}A$ have the same eigenvalues, with the same multiplicities. The constraint preconditioner benefits from certain advantages, as we discuss in Sections 2.4 and 3.4 below. However, this comes at a price, as $\mathcal{P}_{\text{con}}$ is more expensive to apply than $\mathcal{P}_{\text{in}}$ and $\mathcal{P}_{\text{un}}$. To see this, first note that $\mathcal{P}_{\text{con}} = \mathcal{P}_{\text{in}}\mathcal{P}_{\text{r}}$, where

$$
\mathcal{P}_{\text{r}} = \begin{bmatrix}
I & B_1^{-1}B_2 & 0 \\
0 & I & 0 \\
0 & B_1^{-T}X^T & I
\end{bmatrix}.
$$

The additional cost of solving a system with $\mathcal{P}_{\text{con}}$, rather than $\mathcal{P}_{\text{in}}$ or $\mathcal{P}_{\text{un}}$ is thus one solve with each of $B_1$ and $B_1^T$ and a matrix-vector multiplication with each of $B_2$, $A_{11}$ and $A_{12}$.
2.4. Discussion. The preconditioners described above are the result of thinking about the null-space method in terms of a matrix factorization. The preconditioners \( P_{\ln} \) and \( P_{\un} \) are particularly promising. They have the drawback that applying them requires solves with \( B_1 \) and \( B_1^T \), as well as the solve with \( \tilde{N} \) and a number of matrix-vector multiplications. It is also somewhat jarring that we are proposing non-symmetric preconditioners for a symmetric problem (although a short-term recurrence method in a non-standard inner product can be applied as discussed in Section 3.3). Balancing these issues is the fact that the eigenvalue clustering is as good as possible.

The central null preconditioner is indefinite, which means that short-term recurrence methods cannot be straightforwardly applied. Additionally, the spectrum of the preconditioned matrix depends on the eigenvalues of \( N^{-1}A_{22} \) (or \( \tilde{N}^{-1}A_{22} \)), and we shall see in Section 4 that \( P_{\cn} \) is most useful when these eigenvalues are tightly clustered.

Constraint preconditioners possess favourable properties, such as the iterates staying on a certain manifold [47], which can be useful in certain applications. If such properties are desired then \( P_{\con} \) provides them, even with an approximate \( \tilde{N} \), at the expense of extra linear system solves with the matrices \( B_1 \) and \( B_1^T \). This is in contrast to the equivalent Schur-complement formulation [7], which gives an *inexact* constraint preconditioner. As such, null-space preconditioners could be useful in optimization, where solution methods that remain on the constraint manifold are often required; see, e.g., [1]. Additionally, it is possible to use a projected CG or MINRES method in this case, as described in Section 3.4. We envisage that this preconditioner will be particularly useful in fields where many systems have to be solved with the same \( B \) block, possibly with \( A \) changing; an important example that generates linear systems with this structure is the interior point method in optimization [54].

Null-space preconditioners require us to find an invertible subset of the constraint matrix \( B \), which is an additional computational cost that is not present in, for instance, Schur-complement based approaches. However, there are a number of applications we are aware of where this is not problematic; we discuss a few of these in more detail in Section 4.

We note that for problems with maximally rank-deficient \( A \), i.e., problems for which \( \text{rank}(A) = n - m \), alternative block diagonal preconditioners were recently proposed by Estrin and Greif [17] that rely on a matrix \( C \) whose columns span the null-space of \( A \). Estrin and Greif show that under certain conditions, the preconditioned systems can be solved by (standard) conjugate gradients; otherwise a standard non-symmetric Krylov method can be used.

3. Using conjugate gradients and MINRES with the null-space preconditioners. As discussed in Section 2.4, although \( P_{\ln}^{-1}A \), \( P_{\un}^{-1}A \) and \( P_{\con}^{-1}A \) have nice spectra when \( \tilde{N} \) is a good approximation of \( N \), the preconditioners are not symmetric positive definite. Accordingly, they cannot be used with standard MINRES or CG. However, since \( P_{\con} \) is a constraint preconditioner, it can be used with the projected conjugate gradient [20] or projected MINRES [21] methods. On the other hand, although \( P_{\ln} \) is non-symmetric, it can be used in conjunction with the conjugate gradient method in a non-standard inner product. We discuss both these approaches in this section.

3.1. Non-standard inner products. In this section, we show that it is possible to use the conjugate gradient method in a non-standard inner product to solve (1.1)
with the preconditioner (2.5). We consider general equations of the form

$$
\begin{bmatrix}
A & B^T \\
B & C
\end{bmatrix}
\begin{bmatrix}
u \\
v
\end{bmatrix}
= 
\begin{bmatrix}
c \\
d
\end{bmatrix}
$$

(3.1)

where we assume that $A$ is invertible, $A \in \mathbb{R}^{n \times n}$ is symmetric and invertible, $B \in \mathbb{R}^{m \times n}$, $m \leq n$, and $C \in \mathbb{R}^{m \times m}$ is symmetric. We additionally assume that the Schur complement $C - BA^{-1}B^T$ is positive definite, although our results extend trivially to the case where $C - BA^{-1}B^T$ is negative definite. Note that the proceeding results hold for $A = \hat{A}$, $B = \hat{B}$, $C = A_{22}$ in (1.4), as we show in Section 3.3, but are more generally applicable.

Although the saddle point matrix $A$ is indefinite, so that the standard conjugate gradient method cannot be reliably applied to solve (1.1), a judicious choice of preconditioner can make $A$ self-adjoint and positive definite with respect to a non-standard inner product. A number of preconditioners that achieve this goal have been proposed [6, 10, 14, 15, 33, 51]. Many, although not all, fall into the class of preconditioners and inner products discussed by Krzyżanowski [30], who showed the following:

**Proposition 3.1** (Krzyżanowski [30], Proposition 2.1). Suppose we wish to solve the system (3.1). Consider the preconditioner given by

$$
P^{-1} = 
\begin{bmatrix}
I & -dA_0^{-1}B^T \\
0 & I
\end{bmatrix}
\begin{bmatrix}
A_0^{-1} & 0 \\
0 & S_0^{-1}
\end{bmatrix}
\begin{bmatrix}
I & 0 \\
-cBA_0^{-1}B^T & I
\end{bmatrix},
$$

(3.2)

for fixed scalars $c$, $d$, and where $A_0$ and $S_0$ are symmetric and nonsingular. Let $\delta \in \{-1, +1\}$ and $H$ be the block diagonal matrix

$$
H = \delta 
\begin{bmatrix}
A_0 - cA & 0 \\
0 & S_0 + cdBA_0^{-1}B^T - dC
\end{bmatrix}.
$$

(3.3)

Then $HP^{-1}A$ is symmetric.

This means that, even though $P^{-1}A$ is non-symmetric, it is self-adjoint with respect to the bilinear form $\langle \cdot, \cdot \rangle_H$, where $\langle x, y \rangle_H = y^T H x$. If $A_0$, $S_0$ and $\delta$ are chosen so that $H$ is symmetric positive definite, and inner products are understood to be $H$-inner products, then we can apply CG to solve (1.1). Algorithm 1, adapted from Algorithm 3.2 of Dollar, Gould, Stoll and Wathen [15], is one such method which does this.

**3.2. Semidefinite $H$.** Although $P_{ln}$ is not block lower triangular, $\Pi P_{ln} \Pi^T$ is, while $\tilde{A} = \Pi A \Pi^T$ is a generalized saddle point matrix of the form in (3.1). Accordingly, we can apply the results of Krzyżanowski to this permuted matrix and preconditioner. Since the (1,1) block of $\Pi P_{ln} \Pi^T$ is identical to that of $\tilde{A}$, we are in the situation that $c = 1$, $d = 0$ and $A_0 = A$ in (3.2). This may indicate that $P_{ln}$ is a good preconditioner but it also means that $H$ in (3.3) is singular. Despite this, a more careful examination of Algorithm 1 will reveal that this singularity causes no difficulties for computing the solution of (3.1).

Accordingly, we will consider the case $A_0 = A$, $c = 1$ and $d = 0$ in (3.2) and (3.3) in more detail, which arises in our application, but also more widely when solving, e.g., problems in PDE constrained optimization [42, 45]. We assume that $S_0$ is symmetric positive definite, giving

$$
H_{1,0} = 
\begin{bmatrix}
0 & 0 \\
0 & S_0
\end{bmatrix}, \quad P_{1,0} = 
\begin{bmatrix}
A & 0 \\
B & S_0
\end{bmatrix}.
$$

(3.4)
Given $x_0$, set $r_0 = b - Ax_0$, $z_0 = P^{-1}r_0$ and $p_0 = z_0$;

for $k = 0, 1, \ldots$ do

\[
\alpha = \frac{z_k^T H z_k}{p_k^T H P^{-1} A p_k};
\]

\[
x_{k+1} = x_k + \alpha p_k;
\]

\[
r_{k+1} = r_k - \alpha A p_k;
\]

\[
z_{k+1} = P^{-1} r_{k+1};
\]

\[
\beta = \frac{z_{k+1}^T H z_{k+1}}{z_k^T H z_k};
\]

\[
p_{k+1} = z_{k+1} + \beta p_k;
\]

end

Algorithm 1: CG in the scalar product defined by $H$.

Thus, the matrix $H_{1,0}$ is semidefinite with rank $m$.

Letting each vector $v$ in Algorithm 1 be decomposed as $v = \begin{bmatrix} (v^{(1)})^T & (v^{(2)})^T \end{bmatrix}^T$, $v^{(1)} \in \mathbb{R}^n$, it is straightforward to show that Algorithm 1 is equivalent to Algorithm 2. However, it is not yet clear that the iterates generated by Algorithms 1 and 2 converge to the solution of (3.1). To show that Algorithm 2 does indeed solve (3.1) we compare Algorithm 2 to the preconditioned conjugate gradient method applied to the system obtained from the range-space method, i.e, the method based on a factorization like (1.9).

Given $x_0 = \begin{bmatrix} x_0^{(1)} \\ x_0^{(2)} \end{bmatrix}$, set $z_0 = \begin{bmatrix} A^{-1}(c - B^T x_0^{(2)}) - x_0^{(1)} \\ S_0^{-1}(d - BA^{-1} c - (C - BA^{-1} B^T) x_0^{(2)}) \end{bmatrix}$ and $p_0 = z_0$;

for $k = 0, 1, \ldots$ do

\[
\alpha = \frac{(z_k^{(2)})^T S_0 (z_k^{(2)})}{(p_k^{(2)})^T (C - BA^{-1} B^T) p_k^{(2)});
\]

\[
\begin{bmatrix} x_{k+1}^{(1)} \\ x_{k+1}^{(2)} \end{bmatrix} = \begin{bmatrix} x_k^{(1)} + \alpha p_k^{(1)} \\ x_k^{(2)} + \alpha p_k^{(2)} \end{bmatrix};
\]

\[
\begin{bmatrix} z_{k+1}^{(1)} \\ z_{k+1}^{(2)} \end{bmatrix} = \begin{bmatrix} z_k^{(1)} - \alpha (p_k^{(1)} + A^{-1} B^T p_k^{(2)}) \\ z_k^{(2)} - \alpha S_0^{-1}(C - BA^{-1} B^T) p_k^{(2)} \end{bmatrix};
\]

\[
\beta = \frac{(z_{k+1}^{(2)})^T S_0 (z_{k+1}^{(2)})}{(z_k^{(2)})^T S_0 (z_k^{(2)})};
\]

\[
\begin{bmatrix} p_{k+1}^{(1)} \\ p_{k+1}^{(2)} \end{bmatrix} = \begin{bmatrix} z_{k+1}^{(1)} + \beta p_k^{(1)} \\ z_{k+1}^{(2)} + \beta p_k^{(2)} \end{bmatrix};
\]

end

Algorithm 2: Simplified CG in the scalar product defined by $H_{1,0}$.

The range-space method—which is equivalent to solving (3.1)—proceeds in two stages; first we seek a solution to

\[
(C - BA^{-1} B^T) v = d - BA^{-1} c,
\]

before recovering $u$ by solving

\[
u = A^{-1}(c - B^T v).
\]
More details can be found in e.g., [4, Chapter 5].

Since $C - BA^{-1}B^T$ is positive definite we can solve (3.5) by a preconditioned conjugate gradient method with a symmetric positive definite precondioner $S_0$ as in Algorithm 3. Note that when applied to (1.2), our Schur complement approach is actually a null-space method, as we show in Section 3.3 (cf. Algorithm 2.1 in Gould, Hribar and Nocedal [20]).

Given $v_0$, set $z_0 = S_0^{-1}(d - BA^{-1}c - (C - BA^{-1}B^T)v_0)$ and $p_0 = z_0$:

\begin{algorithm}
\begin{align*}
\text{for } & k = 0, 1, \ldots \text{ do} \\
& \alpha = \frac{z_k^T S_0 z_k}{p_k^T (C - BA^{-1}B^T)p_k}; \\
& v_{k+1} = v_k + \alpha p_k; \\
& z_{k+1} = z_k - \alpha S_0^{-1}(C - BA^{-1}B^T)p_k; \\
& \beta = \frac{z_{k+1}^T S_0 z_{k+1}}{z_k^T S_0 z_k}; \\
& p_{k+1} = z_{k+1} + \beta p_k; \\
\end{align*}
\end{algorithm}

Comparison of Algorithms 2 and 3 show that whenever $x_0^{(2)} = v_0$, the vectors $z_0^{(2)}$ and $p_0^{(2)}$ in Algorithm 2 are the same as $z_0$ and $p_0$ in Algorithm 3. Moreover, since the scalars $\alpha$ and $\beta$ in the two algorithms are equivalent, $x_k^{(2)}$, $z_k^{(2)}$ and $p_k^{(2)}$ in Algorithm 2 are the same as $v_k$, $z_k$ and $p_k$ in Algorithm 3 for all iterations $k \geq 0$. It follows from the convergence theory for Algorithm 3 that $\alpha$, $\beta$, $x_k^{(2)}$, $z_k^{(2)}$ and $p_k^{(2)}$ are all well defined and that the iterates $x_k^{(2)}$ in Algorithm 2 are approximations of $v$. However, Algorithm 2 also yields approximations of $u$ as the next result shows.

**Lemma 3.2.** Let $x_k^{(1)}$, $x_k^{(2)}$ and $z_k^{(1)}$ be as in Algorithm 2, $k \geq 0$. Additionally, let

\[ u_k = A^{-1}(c - B^T x_k^{(2)}). \]  \hfill (3.7)

Then $u_k = x_k^{(1)} + z_k^{(1)}$.

**Proof.** We show that $z_k^{(1)} = u_k - x_k^{(1)}$ by induction. First, since $z_0^{(1)} = A^{-1}(c - B^T x_0^{(2)}) - x_0^{(1)}$, we see from (3.7) that $z_0^{(1)} = u_0 - x_0^{(1)}$. Now assume that for some $j \geq 0$, $z_j^{(1)} = u_j - x_j^{(1)}$. From the updates for $x_{j+1}^{(1)}$ and $x_{j+1}^{(2)}$ in Algorithm 2 we see that $\alpha p_j^{(1)} = x_{j+1}^{(1)} - x_j^{(1)}$ and $\alpha p_j^{(2)} = x_{j+1}^{(2)} - x_j^{(2)}$. Substituting these formulae into the equation for $z_{j+1}^{(1)}$ and using (3.7) gives that $z_{j+1}^{(1)} = z_j^{(1)} + (x_j^{(1)} - x_{j+1}^{(1)}) + (u_{j+1} - u_j)$. This shows that whenever $z_j^{(1)} = u_j - x_j^{(1)}$ holds, $z_{j+1}^{(1)} = u_{j+1} - x_{j+1}^{(1)}$. Since $z_0 = u_0 + x_0^{(1)}$, the stated result is proved. \( \Box \)

Now, because $x_k^{(2)}$ approximates $v$, the vector $x_k^{(1)} + z_k^{(1)}$ approximates $u$ and so we obtain approximations of both $u$ and $v$ from Algorithm 2. Thus, Algorithm 2 is well defined and can be used to solve (3.1).

As a final point, since to solve (3.1) we only need $u_k = x_k^{(1)} + z_k^{(1)}$ and $p_k$, it is straightforward to show that nothing is lost in Algorithm 2 by setting $p_k^{(1)} = 0$, $k > 0$, and that some computational savings are made by avoiding the vector update.
Additionally, in our experience such a step can be useful for reducing the effect of rounding errors.

3.3. A non-standard conjugate gradient method for $\mathcal{P}_{ln}$. Now let us apply the results of this section to our system (1.1) with preconditioner (2.5).

Letting

$$\Pi \mathcal{H}_{ln} \Pi^T = \begin{bmatrix} 0 \\ \hat{N} \end{bmatrix},$$

and recalling (1.4) and (2.5), we see that $\Pi \mathcal{H}_{ln} \Pi^T$, $\Pi \mathcal{P}_{ln} \Pi^T$ and $\Pi A \Pi^T$ are in the form of $\mathcal{H}_{1,0}$ and $\mathcal{P}_{1,0}$ in (3.4) and $A$ in (3.1). Accordingly, we can apply the non-standard conjugate gradient method in Algorithm 2 to

$$\Pi(\mathcal{H}_{ln}(A) \Pi^T (\Pi w)) = \Pi \mathcal{H}_{ln} \mathcal{P}_{ln} b,$$

where $w = [x_1^T, x_2^T, y^T]^T$ and $b = [f_1, f_2, g]^T$, with $x = [x_1^T, x_2^T]^T$ and $f = [f_1^T, f_2^T]^T$ and $x_1, f_1 \in \mathbb{R}^m$. As shown in the previous section, this is equivalent to applying preconditioned CG to the reduced system (3.5), given by $Nv = d - \hat{B} \hat{A}^{-1}c$ where, because of the permutation matrix $\Pi$, $v = x_2$, $d = f_2$ and $c = [f_1^T, g]^T$. The vector $u$ in (3.6) is $u = [x_1^T, y]^T$.

Comparison with the null-space method shows that solving this reduced system is the same as using the null-space method with the fundamental basis $Z_f$ and the particular solution (1.8). It follows that applying non-standard CG to (3.8) is equivalent to applying the null-space method using the fundamental basis (1.3).

3.4. The constraint preconditioner, $\mathcal{P}_{cn}$. Finally for this section, we mention the preconditioner $\mathcal{P}_{cn}$. As this is a constraint preconditioner, we can apply it with projected conjugate gradients [20], provided that the (1,1) block is symmetric positive definite on the nullspace of $B$. If $A$ is indefinite, or if we require a method that minimizes the residual, then we can still utilize a short-term recurrence method by using projected MINRES [21]. Therefore standard methods work out-of-the-box, and no extra theory is needed in this case.

4. Numerical Results. In this section we apply null-space preconditioners to matrices arising in applications, each chosen to highlight a specific feature of the proposed preconditioners. We compare their behaviour with the Schur-complement based preconditioners

$$\mathcal{P}_{ls} := \begin{bmatrix} A & 0 \\ B & -S_0 \end{bmatrix}, \quad \mathcal{P}_{cs} := \begin{bmatrix} A & 0 \\ 0 & S_0 \end{bmatrix}, \quad \mathcal{P}_{cons} := \begin{bmatrix} A & B^T \\ B & BA^{-1}B^T - S_0 \end{bmatrix},$$

and use a similar approximation to the Schur complement and the null-space matrix in each case. We also tested $\mathcal{P}_{us}$ and $\mathcal{P}_{um}$, the transposes of $\mathcal{P}_{ls}$ and $\mathcal{P}_{ln}$, but the results were broadly the same as for $\mathcal{P}_{ls}$ and $\mathcal{P}_{ln}$.

To apply $\mathcal{P}_{ls}$ we must solve a system with $A$ and another with $S_0$, while $\mathcal{P}_{ln}$ additionally requires a matrix-vector product with $B$. Applying $\mathcal{P}_{cn}$ requires the same operations as for $\mathcal{P}_{ln}$, as well as an extra matrix-vector product with $B^T$ and a solve with $A$.

Unless otherwise stated we apply right-preconditioned GMRES or non-standard CG, which we terminate when the relative residual satisfies $\|r_k\|_2/\|r_0\|_2 < 10^{-8}$, or when min{$n + m, maxit$} iterations are reached, where $maxit$ is specified for each example below.
For some problems we approximate $N$ and $S$ by incomplete Cholesky factorizations. These we compute by the MATLAB routine `ichol` with a default drop tolerance of $10^{-2}$, which seemed suitable for most of our problems. If `ichol` fails we reduce the drop tolerance by a factor of 10 until a factorization is computed. The smallest drop tolerance used is $10^{-8}$. We point out that this factorization is rather costly to compute, as it first involves forming $N$ or $S$ exactly. Moreover, the cost of the incomplete Cholesky factorization depends on the size of $N$ (or $S$), its sparsity pattern, and the drop tolerance used; here we use larger drop tolerances where possible to reduce this cost. However, these incomplete Cholesky preconditioners may be cheaper to apply than $N$ and $S$, and they may provide information about what to expect for reasonable approximations to $N$ and $S$.

### 4.1. Random saddle point matrices

**Example 4.1.** Consider the pseudo-random sparse matrix generated by the MATLAB code

\[
A = \text{sprandsym}(n,0.1,1e-2,1);
\]
\[
B = \text{sprand}(m,n,0.5);
\]
\[
K = [A \; B'; \; B \; \text{sparsym}(m,m)];
\]

We take $n = 100$ and $m = 10$, $50$, $90$. For these problems $\text{maxit} = 200$.

In this example we test values of $n-m$ that are small, moderate, and large in comparison with $n$. We compare the null-space preconditioners $P_{cn}$ and $P_{un}$, and corresponding Schur complement preconditioners in (4.1). For our approximations to the Schur complement and the null-space matrix we take the identity matrix of the appropriate dimension. We apply these preconditioners using MATLAB’s inbuilt `gmres` routine, and report the results in Figure 4.1. Note that the null-space preconditioners $P_{un}$ and $P_{con}$ give broadly the same behaviour as $P_{ln}$, as does $P_{us}$ and $P_{cons}$ when compared with $P_{ls}$, and so for clarity those results are omitted from Figure 4.1.

Here, by choosing a weak approximation of the Schur complement and the null-space matrix (namely, the identity matrix), convergence depends entirely on how important this component piece is to the overall approximation. Therefore the null space based preconditioners do well for small $n-m$, the Schur complement based preconditioners do well for $n-m$ close to $n$, and there is no clear winner in the intermediate case when $n = m/2$. This suggests that null-space preconditioners may be a better choice over Schur complement preconditioners if we have an application where $n-m$ is small, particularly if we do not have a good Schur complement approximation. We see that for both the Schur complement an null-space preconditioners, the central approximations take roughly twice the number of iterations, but are cheaper to apply. This phenomenon has been noted by others, see e.g. [18].

**Example 4.2.** Consider now the sparse matrix generated by the MATLAB code

\[
A = 10*\text{speye}(n,n);
\]
\[
B = \text{sprand}(m,n,0.5);
\]
\[
K = [A \; B'; \; B \; \text{sparsym}(m,m)];
\]

We take $n = 100$ and $m = 10$, $50$, $90$. In Example 4.2 we have the same $B$ as in Example 4.1, but instead take a scaled identity matrix for $A$. We test the same preconditioners that we used in Example 4.1, namely using an identity matrix as the Schur complement/null space approximation. We give the results in Figure 4.2. Again, $\text{maxit} = 200$, and since $P_{un}$, (resp. $P_{us}$) differs only slightly from $P_{ln}$ and $P_{con}$ (resp. $P_{ls}$ and $P_{cons}$), we report only the former.

Here, in contrast to Example 4.1, the null-space preconditioners perform well for both large and small values of $n-m$. This is because the $(1,1)$ block in $A$ is a scaled identity matrix here and, as a consequence of Theorem 2.1, the eigenvalues of $P_{cn}^{-1}$, $A$
Fig. 4.1: Comparison: Schur complement and null space preconditioners, pseudo-random example from Example 4.1. Note that upper-null and constraint-null (and the respective Schur complement preconditioners) give essentially the same behaviour as lower-null or upper-schur (as appropriate), and so are omitted.

are well clustered. Again, the pattern that the central-based preconditioners take about twice the iterations of the others is in evidence.
Fig. 4.2: Comparison: Schur complement and null space preconditioners, pseudo-random example from Example 4.2. Note that upper-null and constraint-null (and the respective Schur complement preconditioners) give essentially the same behaviour as lower-null or upper-schur (as appropriate), and so are omitted.

4.2. Optimization and interior point methods. Here we consider quadratic programming problems of the form

\[
\begin{align*}
\min & \quad \frac{1}{2} x^T H x + f^T x \\
\text{s.t.} & \quad Bx = g, \\
& \quad x \geq 0.
\end{align*}
\]  

(4.2)
Table 4.1: Problem sizes, CUTEst set matrices from Example 4.2. Matrices for which \( A \) is singular are denoted by an asterisk.

<table>
<thead>
<tr>
<th>Matrix</th>
<th>( n )</th>
<th>( m )</th>
<th>Matrix</th>
<th>( n )</th>
<th>( m )</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUG3DC</td>
<td>3873</td>
<td>1000</td>
<td>HUESTIS</td>
<td>10000</td>
<td>2</td>
</tr>
<tr>
<td>AUG3DCQP</td>
<td>3873</td>
<td>1000</td>
<td>LASER</td>
<td>1002</td>
<td>1000</td>
</tr>
<tr>
<td>CONT-050</td>
<td>2597</td>
<td>2401</td>
<td>LISWET1</td>
<td>10002</td>
<td>10000</td>
</tr>
<tr>
<td>CONT-100</td>
<td>10197</td>
<td>9801</td>
<td>MOSARQP1</td>
<td>2500</td>
<td>700</td>
</tr>
<tr>
<td>CONT-101</td>
<td>10197</td>
<td>10098</td>
<td>MOSARQP2</td>
<td>900</td>
<td>600</td>
</tr>
<tr>
<td>CONT-200</td>
<td>40397</td>
<td>39601</td>
<td>PRIMAL1</td>
<td>325</td>
<td>85</td>
</tr>
<tr>
<td>CVXQP3S</td>
<td>100</td>
<td>75</td>
<td>QPCSTAIR</td>
<td>467</td>
<td>356</td>
</tr>
<tr>
<td>DTOC3</td>
<td>14999</td>
<td>9998</td>
<td>STCQP2</td>
<td>4097</td>
<td>2052</td>
</tr>
<tr>
<td>GOULDQP3</td>
<td>699</td>
<td>349</td>
<td>YAO</td>
<td>2002</td>
<td>2000</td>
</tr>
<tr>
<td>HUES-MOD</td>
<td>10000</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If we solve such a problem using a primal-dual interior point method [54] then at iteration \( k \) of the optimization algorithm we must solve a system of the form (1.1), where \( A = H + X_k^{-1}Z_k \) for diagonal matrices \( X_k, Z_k \).

In this context it is common to solve the linear system (1.1) by reducing it to the null-space matrix \( N \), which the optimization community refer to as the reduced Hessian [8], [37, Section 16.2]. Since forming the matrix \( N \) is expensive, it is common in optimization to approximate this, e.g., by missing out cross terms [12, 36].

In this setting we need to solve a sequence of linear systems as the interior point method converges, but the ‘constraint’ blocks \( B \) do not change. Therefore we may justify the cost of using a direct method such as LUSOL [49], say, to find a basis of \( B \), since we can reuse this basis over all interior point iterations. Although we do not explore the possibility here, it is also possible to use the interior point method itself to predict an invertible sub-block \( B_1 \)—see, e.g., Al-Jeiroudi, Gondzio, and Hall [2]. Note that, but using LUSOL to find a suitable \( B_1 \), we can subsequently solve with \( B_1 \) without further factorization.

In these experiments, we solve for \( A_{11} \) in the null-space preconditioner case and \( A \) in the Schur complement preconditioner case using a direct method (backslash). In many of the examples here this matrices are diagonal, and so such a solve is not a problem. If \( A \) is not diagonal, and a factorization is required, we note that this must only be done once in the interior point method, and the factorization can then be updated using, for example, CHOLMOD [11].

To give a flavour of how we can expect such methods to work we run through some problems from the CUTEst test set [23], comparing standard Schur-complement preconditioners and null-space preconditioners. These problems, and their dimensions, are listed in Table 4.1. We highlight that, as described in a previous section, in optimization it is often important that the inexact solution of this subproblem remains on the constraint manifold. This is a property afforded by constraint preconditioners, and the only true constraint preconditioner tested here is \( P_{con} \).

In our experiments we choose \( X_k \) and \( Z_k \) so that \( X_k^{-1}Z_k = I \), the identity matrix of dimension \( n \); a system of this form may be used in practice to find an initial guess for the interior point method. Additionally, we set \( maxit = 1000 \).

Our first tests are for the ideal case where we take the exact matrices \( S \) or \( N \) (see Table 4.2); these are not practical, but give an idea of the best we can expect the respective method to work in practice. The fast convergence rates for both the Schur-complement and null-space preconditioners, with the exception of the central-null
Table 4.2: Iteration counts for the Schur complement preconditioners with \( S_0 = S \) and the null-space preconditioners with \( N_0 = N \) for the CUTEst set matrices from Example 4.2. * stands for did not converge after 1000 iterations.

<table>
<thead>
<tr>
<th>Matrix</th>
<th>( P_{ls} )</th>
<th>( P_{ln} )</th>
<th>( P_{cs} )</th>
<th>( P_{cn} )</th>
<th>( P_{ls} ) (NSCG)</th>
<th>( P_{ln} ) (NSCG)</th>
<th>( P_{cs} ) (NSCG)</th>
<th>( P_{cn} ) (NSCG)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUG3DC</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>AUG3DCQP</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>CONT-050</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>CONT-100</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>CONT-101</td>
<td>4</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>30</td>
<td>2</td>
<td>*</td>
</tr>
<tr>
<td>CONT-200</td>
<td>4</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>8</td>
<td>25</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>CVXQP3_S</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>34</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>DTOC3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>8</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>GOULDQ3P</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>27</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>HUES-MOD</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>HUESTIS</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>LASER</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>LISWET1</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>MOSARQP1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>21</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>MOSARQP2</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>19</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>PRIMAL1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>22</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>QPCSTAIR</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>31</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>STCQP2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>YAO</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

preconditioner, are to be expected from theoretical spectral results (see Theorems 2.5, 2.6 and 2.8 for the null space preconditioners and [29, 35] for the Schur complement preconditioners). Note that the performance of the central-null preconditioner, in contrast to the other null-space and Schur-complement preconditioners, depends on the eigenvalues of \( N^{-1}A_{22} \) (see Theorem 2.1), which are not necessarily clustered. We find that iteration counts are slightly higher for CONT-101 and CONT-200 than the theory predicts. However, for these matrices \( N \) and \( S \) are quite ill-conditioned.

In a further test we consider the simplest approximation to the matrices \( S \) and \( N \), namely the identity matrix of appropriate size (see Table 4.3). Since the identity is generally a poor approximation of \( S \) and \( N \) we find that, similarly to Example 4.1, the size of \( n - m \) relative to \( m \) largely determines whether the null-space-based preconditioners are more effective than the Schur-complement-based preconditioners. When \( m \) is large the Schur-complement preconditioned iterative methods perform poorly, as might be expected, and do not always converge to the desired tolerance within 1000 iterations. On the other hand, the null-space preconditioners can perform badly when \( n - m \) is large, although these preconditioners do seem somewhat more robust.

Both the central-null and central-Schur preconditioners tend to require twice as many iterations as the other preconditioners. The constraint preconditioner \( P_{cn} \) seems to be generally less effective than \( P_{cn} \) when the approximation to the null-space is poor. However, it does have the benefit of being an exact constraint preconditioner. The non-standard inner product CG method can also require more iterations than using the lower-, upper- or constraint-preconditioners with GMRES, but requires only short-term recurrences.

Finally, we give results (Table 4.4) for the same tests with a more accurate approximation of \( S \) or \( N \), namely the incomplete Cholesky factorization described at the start of this section. Generally, using these better approximations of \( N \) and \( S \) improves the iteration counts for the Schur-complement preconditioners and the null-
Table 4.3: Iteration counts for the Schur complement preconditioners with \( S_0 = I_n \) and the null-space preconditioners with \( N_0 = I_{n-m} \) for the CUTEst set matrices from Example 4.2. * stands for did not converge after 1000 iterations.

<table>
<thead>
<tr>
<th>Matrix</th>
<th>( P_{ls} )</th>
<th>( P_{cs} )</th>
<th>( P_{cons} )</th>
<th>( P_{ls} ) (NSCG)</th>
<th>( P_{cs} )</th>
<th>( P_{cons} )</th>
<th>( P_{ln} ) (NSCG)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUG3DC</td>
<td>37</td>
<td>71</td>
<td>35</td>
<td>35</td>
<td>88</td>
<td>166</td>
<td>91</td>
</tr>
<tr>
<td>AUG3DCQP</td>
<td>38</td>
<td>73</td>
<td>36</td>
<td>36</td>
<td>88</td>
<td>166</td>
<td>91</td>
</tr>
<tr>
<td>CONT-050</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>16</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>CONT-100</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>21</td>
<td>41</td>
<td>21</td>
</tr>
<tr>
<td>CONT-101</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>21</td>
<td>40</td>
<td>31</td>
</tr>
<tr>
<td>CONT-200</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>56</td>
<td>55</td>
<td>28</td>
</tr>
<tr>
<td>CVXQP3S</td>
<td>83</td>
<td>150</td>
<td>83</td>
<td>*</td>
<td>26</td>
<td>44</td>
<td>26</td>
</tr>
<tr>
<td>DTOC3</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>5</td>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>GOULDQP3</td>
<td>21</td>
<td>39</td>
<td>19</td>
<td>19</td>
<td>40</td>
<td>71</td>
<td>41</td>
</tr>
<tr>
<td>HUES-MOD</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>HUESTIS</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>11</td>
</tr>
<tr>
<td>LASER</td>
<td>66</td>
<td>130</td>
<td>65</td>
<td>67</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>LISWET1</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>3</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>MOSARQP1</td>
<td>467</td>
<td>927</td>
<td>464</td>
<td>550</td>
<td>15</td>
<td>29</td>
<td>15</td>
</tr>
<tr>
<td>MOSARQP2</td>
<td>447</td>
<td>889</td>
<td>444</td>
<td>614</td>
<td>17</td>
<td>38</td>
<td>17</td>
</tr>
<tr>
<td>PRIMAL1</td>
<td>79</td>
<td>154</td>
<td>77</td>
<td>137</td>
<td>41</td>
<td>79</td>
<td>41</td>
</tr>
<tr>
<td>QPCSTAIR</td>
<td>249</td>
<td>490</td>
<td>247</td>
<td>551</td>
<td>53</td>
<td>93</td>
<td>53</td>
</tr>
<tr>
<td>STCQP2</td>
<td>269</td>
<td>528</td>
<td>267</td>
<td>615</td>
<td>94</td>
<td>95</td>
<td>93</td>
</tr>
<tr>
<td>YAO</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>3</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

space preconditioners. Again, the null-space preconditioners are more robust than their Schur-complement counterparts, and for no problems do we see the high iteration counts that the Schur-complement preconditioners give for CONT-100, CONT-101 and COND-200.

4.3. Analysis of computational costs for optimization and interior point method matrices. Next we examine in greater detail the computational costs involved in constructing and applying the null-space and Schur complement preconditioners for the CUTEst matrices AUG3DC, MOSARQP1 and STCQP2 from Section 4.2. These matrices were chosen because they exhibit the full range of behaviours observed in the larger test set. The drop tolerances for the incomplete Cholesky preconditioners for these problems are given in Table 4.5.

Table 4.6 shows the times required to compute the preconditioners \( N_0 \) and \( S_0 \) and the number of nonzeros in the resulting factors. The cost of forming the full matrices \( N \) and \( S \) depends on the cost of solving systems with \( B_1 \) and \( A \), respectively. For STCQP2, \( n - m \approx m \) and in this case \( N \) is quicker to compute than \( S \); this is likely because \( N = A_{22} \) is much sparser than \( S \). However, \( m \) is significantly smaller than \( n - m \) for the other matrices, and so it is unsurprising that \( N \) is more expensive to form. The cost of the incomplete Cholesky factorizations, and number of nonzeros in the factors, additionally depends on the drop tolerance and the nonzero pattern of \( N \) and \( S \), as discussed at the start of this section.

In Table 4.7 we examine the time to solve the linear system. We see that for both STCQP2 and MOSARQP1 the timings are generally better for the null-space preconditioners, while Schur-complement preconditioners perform better for AUG3DC. For AUG3DC and STCQP2, this correlates with the number of iterations. However, for STCQP2, fewer iterations are generally needed when Schur-complement based pre-
Table 4.4: Iteration counts for the Schur complement preconditioners and the null-space preconditioners, with incomplete Cholesky preconditioners for $S_0$ and $N_0$ for the CUTEst set matrices from Example 4.2. * stands for did not converge after 1000 iterations.

<table>
<thead>
<tr>
<th>Matrix</th>
<th>$P_{ls}$</th>
<th>$P_{cs}$</th>
<th>$P_{cons}$</th>
<th>$P_{ls}$ (NSCG)</th>
<th>$P_{cn}$</th>
<th>$P_{cn}$</th>
<th>$P_{cons}$</th>
<th>$P_{cn}$ (NSCG)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUG3DC</td>
<td>11</td>
<td>21</td>
<td>9</td>
<td>10</td>
<td>16</td>
<td>33</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>AUG3DCQP</td>
<td>11</td>
<td>21</td>
<td>9</td>
<td>9</td>
<td>16</td>
<td>33</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>CONT-050</td>
<td>86</td>
<td>170</td>
<td>85</td>
<td>86</td>
<td>18</td>
<td>34</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>CONT-100</td>
<td>377</td>
<td>752</td>
<td>376</td>
<td>417</td>
<td>40</td>
<td>59</td>
<td>39</td>
<td>44</td>
</tr>
<tr>
<td>CONT-101</td>
<td>419</td>
<td>837</td>
<td>418</td>
<td>536</td>
<td>26</td>
<td>48</td>
<td>25</td>
<td>*</td>
</tr>
<tr>
<td>CONT-200</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>57</td>
<td>43</td>
<td>23</td>
<td>23</td>
</tr>
<tr>
<td>CVXQP3S</td>
<td>9</td>
<td>14</td>
<td>7</td>
<td>7</td>
<td>6</td>
<td>33</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>DTOC3</td>
<td>9</td>
<td>12</td>
<td>6</td>
<td>7</td>
<td>5</td>
<td>10</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>GOULDQP3</td>
<td>7</td>
<td>13</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td>27</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>HUES-MOD</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>9</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>HUESTIS</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>10</td>
<td>7</td>
<td>*</td>
</tr>
<tr>
<td>LASER</td>
<td>12</td>
<td>20</td>
<td>10</td>
<td>10</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>LISWET1</td>
<td>6</td>
<td>9</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>MOSARQP1</td>
<td>26</td>
<td>51</td>
<td>25</td>
<td>25</td>
<td>7</td>
<td>22</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>MOSARQP2</td>
<td>28</td>
<td>53</td>
<td>26</td>
<td>26</td>
<td>7</td>
<td>19</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>PRIMAL1</td>
<td>5</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>13</td>
<td>25</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>QPCSTAIR</td>
<td>11</td>
<td>20</td>
<td>10</td>
<td>10</td>
<td>20</td>
<td>40</td>
<td>19</td>
<td>20</td>
</tr>
<tr>
<td>STCQP2</td>
<td>16</td>
<td>26</td>
<td>13</td>
<td>13</td>
<td>21</td>
<td>22</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>YAO</td>
<td>5</td>
<td>9</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.5: Drop tolerances for the incomplete Cholesky preconditioners used.

<table>
<thead>
<tr>
<th>Matrix</th>
<th>$N_0$</th>
<th>$S_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IC1</td>
<td>IC2</td>
</tr>
<tr>
<td>AUG3DC</td>
<td>$10^{-4}$</td>
<td>$10^{-4}$</td>
</tr>
<tr>
<td>MOSARQP1</td>
<td>$10^{-1}$</td>
<td>$10^{-2}$</td>
</tr>
<tr>
<td>STCQP2</td>
<td>$10^{-1}$</td>
<td>$10^{-2}$</td>
</tr>
</tbody>
</table>

conditioners are applied, which shows that the cost of applying the preconditioners is important to consider. For STCQP2, $N_0$ is much sparser than $S$ which explains the difference between iteration counts and timings.

Finally, we remark that $\sigma = y^T A_{22} y / y^T N y$ in Theorem 2.1 appears to influence the number of iterations of $P_{cn}$ significantly. When $\sigma$ is near 1, as for STCQP2, few iterations are required. However, for AUG3DC and MOSARQP1, for which $\sigma_{\text{min}} \ll 1$ (see Table 2.1), the number of iterations is quite a bit higher.

4.4. $\mathcal{F}$−matrices. Let $A$ be a saddle point matrix of the form (1.1) where $A$ is symmetric positive definite and $B$ is a gradient matrix, i.e., $B$ has at most two entries per row, and if there are two entries they sum to zero; we call such a matrix $A$ an $\mathcal{F}$−matrix [52]. Such matrices arise naturally in, e.g., discretizations of fluid-flow [3], or in electrical networks [50].

Due to the special structure of $B$ it is possible to find an invertible sub-block $B_1$ without performing any arithmetic—see, e.g., [44, 50]. This property makes $\mathcal{F}$−matrices an ideal candidate for null-space preconditioning. We test our preconditioners for a number of $\mathcal{F}$−matrices$^2$, listed in Table 4.8. We set $\text{maxit} = 1000$.

$^2$We would like to thank Miroslav Tůma for providing these test matrices.
Table 4.6: Time to compute $N_0$ or $S_0$, including the time to compute $N$ or $S$ when incomplete Cholesky factorizations are used, and the number of nonzeros in $N_0$ and $S_0$. For incomplete Cholesky preconditioners we report the number of nonzeros in one factor.

<table>
<thead>
<tr>
<th></th>
<th>Matrix</th>
<th>$N_0$</th>
<th>$N_0$</th>
<th>$N_0$</th>
<th>$S_0$</th>
<th>$S_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>IC1</td>
<td>IC2</td>
<td></td>
<td>IC1</td>
</tr>
<tr>
<td>Time</td>
<td>AUG3DC</td>
<td>0.0025</td>
<td>0.013</td>
<td>0.036</td>
<td>0.0008</td>
<td>0.00042</td>
</tr>
<tr>
<td></td>
<td>MOSARQP1</td>
<td>0.39</td>
<td>0.39</td>
<td>0.39</td>
<td>0.013</td>
<td>0.015</td>
</tr>
<tr>
<td></td>
<td>STCQP2</td>
<td>0.00062</td>
<td>0.00076</td>
<td>0.0014</td>
<td>1.2</td>
<td>1.2</td>
</tr>
<tr>
<td>nnz</td>
<td>AUG3DC</td>
<td>7.8E4</td>
<td>1.1E5</td>
<td>2.7E5</td>
<td>6.3E3</td>
<td>1.8E3</td>
</tr>
<tr>
<td></td>
<td>MOSARQP1</td>
<td>5E3</td>
<td>1.8E3</td>
<td>2.2E3</td>
<td>8.6E3</td>
<td>1.4E4</td>
</tr>
<tr>
<td></td>
<td>STCQP2</td>
<td>1.4E4</td>
<td>2.5E3</td>
<td>1E4</td>
<td>6.1E5</td>
<td>1.2E5</td>
</tr>
</tbody>
</table>

Table 4.7: Time to compute the solution using preconditioned GMRES or preconditioned nonstandard CG.

<table>
<thead>
<tr>
<th></th>
<th>Matrix</th>
<th>$P_{ls}$</th>
<th>$P_{ls}$</th>
<th>$P_{cons}$</th>
<th>$P_{ls}$ (NSCG)</th>
<th>$P_{ln}$</th>
<th>$P_{ln}$</th>
<th>$P_{cons}$</th>
<th>$P_{ln}$ (NSCG)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>N</td>
<td>0.030</td>
<td>0.032</td>
<td>0.023</td>
<td>0.0078</td>
<td>0.046</td>
<td>0.27</td>
<td>0.034</td>
<td>0.024</td>
</tr>
<tr>
<td></td>
<td>I</td>
<td>0.047</td>
<td>0.099</td>
<td>0.057</td>
<td>0.018</td>
<td>0.12</td>
<td>0.65</td>
<td>0.23</td>
<td>0.066</td>
</tr>
<tr>
<td></td>
<td>IC1</td>
<td>0.070</td>
<td>0.10</td>
<td>0.096</td>
<td>0.020</td>
<td>0.076</td>
<td>0.18</td>
<td>0.086</td>
<td>0.14</td>
</tr>
<tr>
<td></td>
<td>IC2</td>
<td>0.026</td>
<td>0.046</td>
<td>0.039</td>
<td>0.010</td>
<td>0.073</td>
<td>0.21</td>
<td>0.053</td>
<td>0.22</td>
</tr>
<tr>
<td></td>
<td>MOSARQP1</td>
<td>0.029</td>
<td>0.032</td>
<td>0.027</td>
<td>0.0076</td>
<td>0.025</td>
<td>0.076</td>
<td>0.044</td>
<td>0.012</td>
</tr>
<tr>
<td></td>
<td>IC1</td>
<td>0.069</td>
<td>0.10</td>
<td>0.096</td>
<td>0.029</td>
<td>0.054</td>
<td>0.11</td>
<td>0.084</td>
<td>0.030</td>
</tr>
<tr>
<td></td>
<td>IC2</td>
<td>0.026</td>
<td>0.033</td>
<td>0.030</td>
<td>0.020</td>
<td>0.032</td>
<td>0.079</td>
<td>0.062</td>
<td>0.021</td>
</tr>
<tr>
<td></td>
<td>STCQP2</td>
<td>0.26</td>
<td>0.27</td>
<td>0.18</td>
<td>0.17</td>
<td>0.028</td>
<td>0.031</td>
<td>0.027</td>
<td>0.013</td>
</tr>
<tr>
<td></td>
<td>I</td>
<td>3.3</td>
<td>7.1</td>
<td>6.5</td>
<td>3.8</td>
<td>0.44</td>
<td>0.17</td>
<td>0.36</td>
<td>0.11</td>
</tr>
<tr>
<td></td>
<td>IC1</td>
<td>0.17</td>
<td>0.27</td>
<td>0.29</td>
<td>0.23</td>
<td>0.098</td>
<td>0.083</td>
<td>0.14</td>
<td>0.056</td>
</tr>
<tr>
<td></td>
<td>IC2</td>
<td>0.12</td>
<td>0.16</td>
<td>0.15</td>
<td>0.27</td>
<td>0.064</td>
<td>0.094</td>
<td>0.075</td>
<td>0.047</td>
</tr>
</tbody>
</table>

When we use the cheap, but inaccurate, approximations $S_0 = I$ and $N_0 = I$ (see Table 4.9) the iteration counts can be quite high for all preconditioners. However, the null-space based preconditioners consistently give lower iteration counts; this can partly be explained by the dimensions of the problems, since in general $n - m$ is significantly smaller than $m$. As in the previous example, the non-standard CG method with $P_{ls}$ or $P_{ln}$ seems to be less robust than right-preconditioned GMRES. Similarly to other examples in this section, the central-Schur and central-null preconditioners tend to take twice as many iterations as the other preconditioners. When $\tilde{N}$ and $S$ are replaced by incomplete Cholesky preconditioners, the iteration counts drop for all preconditioners, but the same trends are evident. In particular, the null-space preconditioners are particularly well-suited to these $F$-matrices.

5. Conclusion. We have presented novel preconditioners based on a null-space factorization. By dropping, or approximating, different terms in the null-space factorization, in a similar manner to standard Schur complement preconditioners, we arrived at four different null-space preconditioners.

We have given eigenvalue bounds for these preconditioners, and have shown that the eigenvalues of the upper-null, lower-null and constraint preconditioners are clustered when a good approximation to the null-space matrix can be found. Additionally, two of the preconditioners, although indefinite and non-symmetric, can be applied with a Krylov method with a short term recurrence.
Table 4.8: Problem sizes, $\mathcal{F}$-matrices in Example 4.4.

<table>
<thead>
<tr>
<th>Matrix</th>
<th>$n$</th>
<th>$m$</th>
<th>Matrix</th>
<th>$n$</th>
<th>$m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>DORT</td>
<td>13360</td>
<td>9607</td>
<td>M3P</td>
<td>2160</td>
<td>1584</td>
</tr>
<tr>
<td>DORT2</td>
<td>7515</td>
<td>5477</td>
<td>S3P</td>
<td>270</td>
<td>207</td>
</tr>
<tr>
<td>L3P</td>
<td>17280</td>
<td>12384</td>
<td>dan2</td>
<td>63750</td>
<td>46661</td>
</tr>
</tbody>
</table>

Table 4.9: Iteration counts for the Schur complement preconditioners with $S_0 = I_n$ and the null-space preconditioners with $N_0 = I_{n-m}$ for the $\mathcal{F}$-matrices in Example 4.4.

* stands for did not converge after 1000 iterations.

<table>
<thead>
<tr>
<th>Matrix</th>
<th>$P_{ls}$</th>
<th>$P_{cs}$</th>
<th>$P_{cons}$</th>
<th>$P_{ls}$ (NSCG)</th>
<th>$P_{ln}$</th>
<th>$P_{cn}$</th>
<th>$P_{conn}$</th>
<th>$P_{ln}$ (NSCG)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DORT</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>763</td>
<td>*</td>
<td>750</td>
<td>*</td>
</tr>
<tr>
<td>DORT2</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>481</td>
<td>946</td>
<td>473</td>
<td>*</td>
</tr>
<tr>
<td>L3P</td>
<td>360</td>
<td>717</td>
<td>373</td>
<td>375</td>
<td>223</td>
<td>441</td>
<td>217</td>
<td>298</td>
</tr>
<tr>
<td>M3P</td>
<td>205</td>
<td>407</td>
<td>224</td>
<td>207</td>
<td>91</td>
<td>177</td>
<td>89</td>
<td>108</td>
</tr>
<tr>
<td>S3P</td>
<td>114</td>
<td>225</td>
<td>126</td>
<td>115</td>
<td>36</td>
<td>65</td>
<td>36</td>
<td>34</td>
</tr>
<tr>
<td>dan2</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

Finally, we investigated the effectiveness of these preconditioners at reducing the number of iterations of Krylov subspace methods. We found that the preconditioners were more robust than equivalent Schur-complement based preconditioners, and were more effective when a reasonable approximation to the null-space matrix was available or when the dimension of $n - m$ was small.

Acknowledgements. The authors extend their thanks to Jennifer Scott and Nick Gould for reading an earlier version of this manuscript, and for their valuable comments and suggestions. We also thank two anonymous referees for their careful reading and constructive comments.

REFERENCES

Table 4.10: Iteration counts for the Schur complement preconditioners and the null-space preconditioners, with incomplete Cholesky preconditioners for $S_0$ and $N_0$, for the $F$-matrices in Example 4.4.

<table>
<thead>
<tr>
<th>Matrix</th>
<th>$P_{ls}$</th>
<th>$P_{cs}$</th>
<th>$P_{cons}$</th>
<th>$P_{ls}$ (NSCG)</th>
<th>$P_{ln}$</th>
<th>$P_{cn}$</th>
<th>$P_{conn}$</th>
<th>$P_{ln}$ (NSCG)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DORT</td>
<td>121</td>
<td>237</td>
<td>120</td>
<td>125</td>
<td>14</td>
<td>32</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>DORT2</td>
<td>117</td>
<td>233</td>
<td>116</td>
<td>120</td>
<td>10</td>
<td>30</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>L3P</td>
<td>44</td>
<td>87</td>
<td>43</td>
<td>43</td>
<td>18</td>
<td>36</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>M3P</td>
<td>24</td>
<td>47</td>
<td>23</td>
<td>23</td>
<td>15</td>
<td>31</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>S3P</td>
<td>12</td>
<td>23</td>
<td>11</td>
<td>11</td>
<td>12</td>
<td>28</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>dan2</td>
<td>7</td>
<td>13</td>
<td>6</td>
<td>6</td>
<td>11</td>
<td>48</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>


