SPMR: A FAMILY OF SADDLE-POINT MINIMUM RESIDUAL SOLVERS

RON ESTRIN* AND CHEN GREIF†

Abstract. We introduce SPMR, a new family of methods for iteratively solving saddle-point systems using a minimum or quasi-minimum residual approach. No symmetry assumptions are made. The basic mechanism underlying the method is a novel simultaneous bidiagonalization procedure that yields a simplified saddle-point matrix on a projected Krylov-like subspace, and allows for a monotonic short-recurrence iterative scheme. We develop a few variants, demonstrate the advantages of our approach, derive optimality conditions, and discuss connections to existing methods. Numerical experiments illustrate the merits of this new family of methods.

Key words. saddle-point systems, iterative solvers, Krylov subspaces, bidiagonalization, minimum residual, preconditioning

AMS subject classifications. 15A06, 15A18, 65F08, 65F10, 65F25, 65F50

1. Introduction.

Consider the problem of iteratively solving large and sparse saddle-point systems of the form

$$\begin{pmatrix} A & G_1^T \\ G_2 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}, \tag{1}$$

where $A \in \mathbb{R}^{n\times n}, G_1, G_2 \in \mathbb{R}^{m\times n}, f \in \mathbb{R}^n$, and $g \in \mathbb{R}^m$. We assume, as is typically the case in most applications, that $m < n$. Throughout our discussion we will denote the matrix of (1) by $\mathcal{K}$:

$$\mathcal{K} = \begin{pmatrix} A & G_1^T \\ G_2 & 0 \end{pmatrix}.$$

Saddle-point systems arise in a large variety of applications, and numerical solution methods have been extensively explored [5, 7, 33]. But there are relatively few solvers that have been tailored specifically to the block structure of these systems. Rather, general iterative solvers are typically used, and exploiting the block structure is often reserved to the preconditioning stage. Our goal is to develop solvers for (1) that take into account the block structure of the matrix $\mathcal{K}$. We are interested in the most generic setting here, i.e., we allow A to be any matrix (from symmetric positive definite to symmetric indefinite to nonsymmetric), and allow $G_1 \neq G_2$.

We introduce a family of short recurrence solvers that are based on residual norm minimization or quasi-minimization, and call this family **<u>SPMR</u>: Saddle-Point Minimum Residual.**

One of the innovations that we offer in the derivation of SPMR is the bidiagonalization of the two off-diagonal block matrices, G_1 and G_2, using a procedure similar in spirit to the generalized Golub-Kahan bidiagonalization [2, 3, 15], along with a simultaneous diagonalization of A.

Solving saddle-point systems is a challenging task, and numerical methods typically involve potentially costly interim computations, such as inversion or the computation of a null space. The SPMR family can be split into two main sub-families: (i)

*Institute for Computational and Mathematical Engineering, Stanford University, Stanford CA, USA, restrin@stanford.edu.

†Department of Computer Science, The University of British Columbia, Vancouver BC, Canada, greif@cs.ubc.ca. The work of this author was partially supported by a Discovery Grant of the Natural Science and Engineering Council of Canada (NSERC).

methods that require the inversion of A, and (ii) methods that use null spaces of G_1 and G_2. The first set of methods, (i), is restricted to situations where A is invertible and the inversion operation is computationally inexpensive. These methods implicitly solve linear systems associated with the Schur complement,

$$S = G_2 A^{-1} G_1^T. \tag{2}$$

The second set of methods, sub-family (ii), may be appealing when the null spaces of G_1 and G_2 are relatively easy to detect or when we have basis-free procedures that can efficiently utilize these null spaces. These methods implicitly solve linear systems associated with

$$R = H_1^T A H_2, \tag{3}$$

where H_1 and H_2 are such that $G_1 H_1 = G_2 H_2 = 0$. We call R the *generalized reduced Hessian*, because it generalizes the notion of the reduced Hessian in optimization, when A is symmetric, $G_1 = G_2$ and (1) arises from a quadratic programming problem [23].

SPMR projects the given saddle-point matrix onto a smaller subspace where the (projected) matrix has a simple saddle-point block structure. In this regard, it is similar to the augmented system interpretation of LSQR [24] and LSMR [11]. We provide a characterization of the search space, show connections to other methods such as USYMQR [27], and apply an optimality criterion similar to the approach taken in the development of QMR [13]. In the specific case that A is symmetric positive definite and $G_1 = G_2$, our solvers reduce to the generalized LSQR developed by Arioli & Orban, the Projected Conjugate Gradient method developed by Hribar, Gould and Nocedal, and related solvers [3, 16, 17].

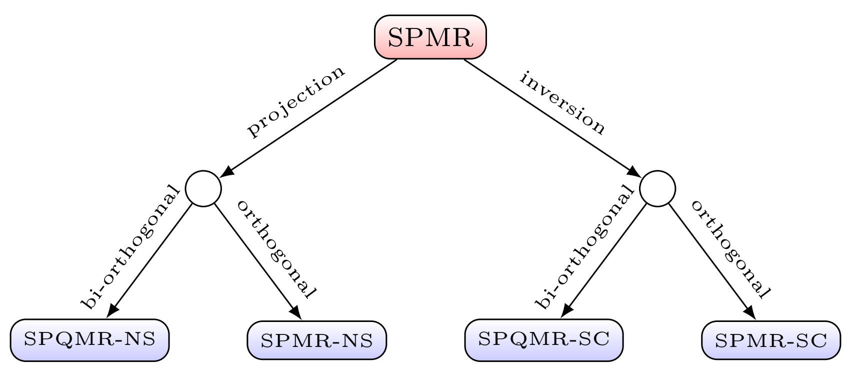


Fig. 1: Various versions of SPMR.

Fig. 1 is a schematic of the SPMR family: ‘SC’ stands for Schur complement, and ‘NS’ stands for null-space. SPMR and SPQMR differ from each other by the choice of residual minimization or quasi-minimization, respectively, when solving the relevant subproblem. As common for iterative solvers, this difference can also be characterized by orthogonalization vs. bi-orthogonalization; consider for example USYMQR vs. QMR.

In Section 2 we describe the basic principles of SPMR, including details on the bidiagonalization procedure that forms the core of our approach. Sections 3 and 4 provide the derivations of the two sub-families of SPMR: SPMR-SC, which requires the inversion of A, and SPMR-NS, which requires computation of the null spaces of G_1 and G_2. In Section 5 we discuss properties of the SPMR solvers. In Section 6 we develop a variant that we call SPQMR, which relies on residual quasi-minimization. Here again, we offer two variants, SPQMR-SC and SPQMR-NS. In Section 7 we address the important issue of preconditioning and introduce preconditioned versions of SPMR and its variants. In Section 8 we show a few examples that illustrate the various features of our new family of methods. Finally, in Section 9 we draw some conclusions.

We use standard Householder's notation throughout (capital letters for matrices, lower-case letters for vectors, and Greek letters for scalars), and unless otherwise stated, the notation $\|\cdot\|$ signifies the ℓ_2 vector norm.

2. SPMR.

We now derive SPMR and its variants. As we shall see, the core of our algorithms is a Lanczos-like procedure called *SIMBA*.

2.1. Right Hand Side Setting.

It is convenient to set the right-hand side in correlation with the family members that we choose to use. If A is efficiently invertible, general right-hand sides $(f^T, g^T)^T$ can be handled by solving $A\hat{x} = f$, and then solving

$$\begin{pmatrix} A & G_1^T \\ G_2 & 0 \end{pmatrix} \begin{pmatrix} x' \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ g - G_2\hat{x} \end{pmatrix}, \qquad x = x' + \hat{x}.$$

We could therefore assume in this case, without loss of generality, that we need to solve systems of the form

$$\begin{pmatrix} A & G_1^T \\ G_2 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ g \end{pmatrix}, \tag{4}$$

and proceed to develop methods in the 'SC' sub-family. Like the generalized LSQR method [3], we are constrained to solve systems with a zero block, which means that it is necessary to form $g - G_2\hat{x}$ on the right-hand side.

On the other hand, if we are solving with general right-hand side $(f^T, g^T)^T$ and we wish to avoid inverting A, if we are able to find a particular solution $G_2\hat{x} = g$, then we can instead solve

$$\begin{pmatrix} A & G_1^T \\ G_2 & 0 \end{pmatrix} \begin{pmatrix} x' \\ y \end{pmatrix} = \begin{pmatrix} f - A\hat{x} \\ 0 \end{pmatrix}, \qquad x = x' + \hat{x}.$$

We can then focus on saddle-point systems of the form

$$\begin{pmatrix} A & G_1^T \\ G_2 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix}. \tag{5}$$

In this case it is possible to have A singular, and our focus will be on developing 'NS'-type methods, which require using the null spaces of G_1 and G_2.

2.2. The Dual Saddle-Point System.

Let H_1 and H_2 be null-space bases so that $G_1H_1 = G_2H_2 = 0$. From (5) we can see that since $G_2x = 0$, then $x = H_2q$ for some q. Furthermore, if we consider the first equation $Ax + G_1^T y = f$, we can see that by applying H_1^T from the left, we get

$$\begin{aligned} H_1^T f &= H_1^T A x + H_1^T G_1^T y \\ &= H_1^T A H_2 q = Rq, \end{aligned} \tag{6}$$

where R is the generalized reduced Hessian defined in (3).

If A were invertible, then we could recognize (6) as the range-space method (referred to also as the Schur complement method) applied to the *dual saddle-point system*, described in [5]:

$$\begin{pmatrix} A^{-1} & H_2 \\ H_1^T & 0 \end{pmatrix} \begin{pmatrix} \tilde{p} \\ q \end{pmatrix} = \begin{pmatrix} 0 \\ -H_1^T f \end{pmatrix}. \tag{7}$$

Notice also that in that case, if A were invertible, (6) would be equivalent to the system

$$H_1^T f = (H_1^T A) A^{-1} (A H_2) q.$$

But the above is nothing but the system corresponding to the range-space method applied to the saddle-point system

$$\begin{pmatrix} A & AH_2 \\ H_1^T A & 0 \end{pmatrix} \begin{pmatrix} p \\ q \end{pmatrix} = \begin{pmatrix} 0 \\ -H_1^T f \end{pmatrix}. \tag{8}$$

We call (8) *the inverse-free dual saddle-point system,* and we will denote the matrix by

$$\mathcal{K}_D = \begin{pmatrix} A & AH_2 \\ H_1^T A & 0 \end{pmatrix}.$$

Moving forward, we will use the shorthand expression "dual system" in reference to (8) rather than (7), since the need to use an inverse-free version is central. A key point here is that once we have defined this dual system, there is no longer a need to assume that A is invertible, even though we assumed that in order to obtain (8).

At first glance, it would appear that the system in (8) has some issues pertaining to singularity: if either A or the H_i are singular, then the system itself is singular. Let us alleviate those concerns with the following theorem.

THEOREM 1. *Suppose that $\mathcal{K}$ is nonsingular, without further assumptions on A. Let x and y be the unique solution to* (5)*. Then there exists a solution to* (8) *such that $p \in \ker(G_2)$. For this p, we can recover x and y, as follows: set $x = -p$ and obtain y from the consistent overdetermined system*

$$G_1^T y = f + Ap = f - Ax.$$

Proof. We first show that there exists $p \in \ker(G_2)$ which solves (8). Note that there exist unique x, y which solve (5) since $\mathcal{K}$ is nonsingular, and that $x = H_2 q \in \ker(G_2)$ for the q chosen in (6); we therefore choose $p = -x$ and show that this choice satisfies (8). We have

$$\begin{pmatrix} A & AH_2 \\ H_1^T A & 0 \end{pmatrix} \begin{pmatrix} p \\ q \end{pmatrix} = \begin{pmatrix} -AH_2 q + AH_2 q \\ -H_1^T A H_2 q \end{pmatrix} = \begin{pmatrix} 0 \\ -Rq \end{pmatrix} = \begin{pmatrix} 0 \\ -H_1^T f \end{pmatrix},$$

so this choice of $p \in \ker(G_2)$ and q indeed solves (8).

We now show that if $p \in \ker(G_2)$ and p, q solve (8), then $x = -p$ solves (5) and $G_1^T y = f - Ax$ is consistent. We have $G_2 x = 0$ since $x = -p \in \ker(G_2)$, and from (8) we have

$$0 = H_1^T (f + Ap) = H_1^T (f - Ax),$$

so that $f - Ax \in \ker(H_1^T) = \text{range}(G_1^T)$; therefore $G_1^T y = f - Ax$ is consistent. ∎

2.3. SIMBA: Simultaneous Bidiagonalization via A-Conjugacy. A cornerstone of our method is a technique of simultaneous bidiagonalization. We construct a projected subspace that includes a diagonal reduction of the leading block and bidiagonalized versions of the off diagonal blocks. We call it SIMBA: Simultaneous Bidiagonalization via A-conjugacy.

SIMBA

has two variants: one that relies on inverting A (when applicable), and one that relies on null spaces of G_1 and G_2. In the latter case A may be singular, and we will turn to using the dual system, (8).

Define

$$B_k = \begin{pmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & \beta_k & \alpha_k \\ & & & \beta_{k+1} \end{pmatrix} = \begin{pmatrix} L_k \\ \beta_{k+1} e_k^T \end{pmatrix} \tag{9}$$

and

$$C_k = \begin{pmatrix} \gamma_1 & & & \\ \delta_2 & \gamma_2 & & \\ & \ddots & \ddots & \\ & & \delta_k & \gamma_k \\ & & & \delta_{k+1} \end{pmatrix} = \begin{pmatrix} M_k \\ \delta_{k+1} e_k^T \end{pmatrix}. \tag{10}$$

We will construct bases

$$\begin{aligned} U_k &= [u_1 \dots u_k], & V_k &= [v_1 \dots v_k], \\ W_k &= [w_1 \dots w_k], & Z_k &= [z_1 \dots z_k], \end{aligned} \tag{11}$$

where the construction depends on whether we use A inversions, or whether we rely on null spaces of G_1 and G_2.

2.3.1. SIMBA-SC: Using A Inversion. Suppose that A is invertible, and that inverting A is computationally inexpensive and may be done throughout the iteration. We construct the matrices specified in (9)-(11) such that the following relations are satisfied:

$$\begin{aligned} G_1^T V_k &= A U_k J_k L_k^T, & W_k^T A U_k &= J_k, \\ G_1 W_k &= V_{k+1} B_k, & V_k^T V_k &= I, \\ G_2^T Z_k &= A^T W_k J_k M_k^T, & Z_k^T Z_k &= I, \\ G_2 U_k &= Z_{k+1} C_k, \end{aligned} \tag{12}$$

where J_k is diagonal such that $(J_k)_{j,j} = \xi_j = \pm 1$.

In the case where A is symmetric and $G_1 = G_2$, we will have $U_k = W_k$ and $V_k = Z_k$, allowing us to cut the computational and storage requirements in half. This is because even if A is indefinite, by Silvester's Law of Inertia, we use J_k to absorb the indefiniteness of $U_n^T A U_n$.

The above relations lead to Algorithm 1, which in exact arithmetic produces orthogonal V_k, Z_k, and biconjugate U_k, W_k. This is one variant of the SIMBA procedure, which we call SIMBA-SC, because it relies an implicit construction of the Schur complement, S. We describe the algorithm using separate columns for the computation of u_k, v_k and w_k, z_k to highlight the symmetry between the two pairs of vectors.

Algorithm 1 SIMBA-SC: Simultaneous Bidiagonalization via A-conjugacy, using A inversion and an implicit construction of the Schur complement.

INPUT: A, G_1, G_2, b, c
// Recall that $\|v_k\| = \|z_k\| = 1$ for all k

$\beta_1 v_1 \leftarrow b$	$\delta_1 z_1 \leftarrow c$
$\hat{u}_1 \leftarrow G_1^T v_1$	$\hat{w}_1 \leftarrow G_2^T z_1$
$u_1 \leftarrow A^{-1} \hat{u}_1$	$w_1 \leftarrow A^{-T} \hat{w}_1$
$\xi_1 \leftarrow \operatorname{sgn}(w_1^T \hat{u}_1)$	
$\alpha_1 \leftarrow \lvert w_1^T \hat{u}_1 \rvert^{1/2}$	$\gamma_1 \leftarrow \alpha_1$
$u_1 \leftarrow \xi_1 u_1 / \alpha_1$	$w_1 \leftarrow \xi_1 w_1 / \gamma_1$

for $k = 1, 2, \ldots$ **do**

$\beta_{k+1} v_{k+1} \leftarrow G_1 w_k - \alpha_k v_k$	$\delta_{k+1} z_{k+1} \leftarrow G_2 u_k - \gamma_k z_k$
$\hat{u}_{k+1} \leftarrow G_1^T v_{k+1} / \beta_{k+1}$	$\hat{w}_{k+1} \leftarrow G_2^T z_{k+1} / \delta_{k+1}$
$u_{k+1} \leftarrow A^{-1} \hat{u}_{k+1} - \xi_k \beta_{k+1} u_k$	$w_{k+1} \leftarrow A^{-T} \hat{w}_{k+1} - \xi_k \delta_{k+1} w_k$
$\xi_{k+1} \leftarrow \operatorname{sgn}(w_{k+1}^T \hat{u}_{k+1})$	
$\alpha_{k+1} \leftarrow \lvert w_{k+1}^T \hat{u}_{k+1} \rvert^{1/2}$	$\gamma_{k+1} \leftarrow \alpha_{k+1}$
$u_{k+1} \leftarrow \xi_{k+1} u_{k+1} / \alpha_{k+1}$	$w_{k+1} \leftarrow \xi_{k+1} w_{k+1} / \gamma_{k+1}$

end for

2.3.2. SIMBA-NS: Using Null Spaces of G_1 and G_2. Suppose now that computing null spaces of G_1 and G_2 is computationally viable, whereas inverting A is not computationally attractive or is impossible due to singularity. We first notice that mathematically, if A is invertible, when we apply SIMBA-SC in Algorithm 1 to the dual system in (8), all inverses by A and A^T will cancel with the off-diagonal blocks AH_2 and $A^T H_1$. It is thus possible to derive an A inversion-free version of SIMBA-SC. This version requires the availability of the null spaces of G_1 and G_2.

Suppose H_1 and H_2 are given, such that $G_1 H_1 = 0$ and $G_2 H_2 = 0$. We define B_k as in (9) and C_k as in (10), and then construct bases as in (11), but with (12) replaced by

$$\begin{aligned} H_2 V_k &= U_k J_k L_k^T, & W_k^T A U_k &= J_k, \\ H_2^T A^T W_k &= V_{k+1} B_k, & V_k^T V_k &= I, \\ H_1 Z_k &= W_k J_k M_k^T, & Z_k^T Z_k &= I, \\ H_1^T A U_k &= Z_{k+1} C_k, & & \end{aligned} \tag{13}$$

where again J_k is diagonal such that $(J_k)_{j,j} = \xi_j = \pm 1$.

Algorithm 2 thus gives us an alternative formulation of SIMBA. We call it SIMBA-NS, to mark its reliance on null spaces.

2.4. Characterization of the Search Subspace. The following theorem states that Algorithm 1 and Algorithm 2 produce the desired bidiagonalizations. The proof of this theorem is by induction, similarly to the way the Lanczos method is derived, and is omitted for the sake of brevity.

THEOREM 2. *In exact arithmetic, the vectors generated by Algorithm 1 and Algorithm 2 satisfy the relationships in (12) and (13), respectively.*

The construction makes it clear that the simultaneous bidiagonalization is unique up to the choice of starting vectors v_1 and z_1, and the choice in the relative scaling

Algorithm 2 SIMBA-NS: Simultaneous Bidiagonalization via A-conjugacy, using the null spaces of G_1 and G_2, namely H_1 and H_2 such that $G_1H_1 = 0$ and $G_2H_2 = 0$.

INPUT: A, H_1, H_2, b, c

// Recall that $\|v_k\| = \|z_k\| = 1$ for all k

$$
\begin{array}{ll}
\beta_1 v_1 \leftarrow b & \delta_1 z_1 \leftarrow c \\
\quad u_1 \leftarrow H_2 v_1 & \quad w_1 \leftarrow H_1 z_1 \\
\quad \hat{u}_1 \leftarrow A u_1 & \quad \hat{w}_1 \leftarrow A^T w_1 \\
\quad \xi_1 \leftarrow \operatorname{sgn}(w_1^T \hat{u}_1) & \\
\quad \alpha_1 \leftarrow |w_1^T \hat{u}_1|^{1/2} & \quad \gamma_1 \leftarrow \alpha_1 \\
\quad u_1 \leftarrow \xi_1 u_1 / \alpha_1 & \quad w_1 \leftarrow \xi_1 w_1 / \gamma_1
\end{array}
$$

for $k = 1, 2, \ldots$ **do**

$$
\begin{array}{ll}
\beta_{k+1} v_{k+1} \leftarrow H_2^T \hat{w}_k / \gamma_k - \alpha_k v_k & \delta_{k+1} z_{k+1} \leftarrow H_1^T \hat{u}_k / \alpha_k - \gamma_k z_k \\
\quad u_{k+1} \leftarrow H_2 v_{k+1} / \beta_{k+1} - \xi_k \beta_{k+1} u_k & \quad w_{k+1} \leftarrow H_1 z_{k+1} / \delta_{k+1} - \xi_k \delta_{k+1} w_k \\
\quad \hat{u}_{k+1} \leftarrow A u_{k+1} & \quad \hat{w}_{k+1} \leftarrow A w_{k+1} \\
\quad \xi_{k+1} \leftarrow \operatorname{sgn}(w_{k+1}^T \hat{u}_{k+1}) & \\
\quad \alpha_{k+1} \leftarrow |w_{k+1}^T \hat{u}_{k+1}|^{1/2} & \quad \gamma_{k+1} \leftarrow \alpha_{k+1} \\
\quad u_{k+1} \leftarrow \xi_{k+1} u_{k+1} / \alpha_{k+1} & \quad w_{k+1} \leftarrow \xi_{k+1} w_{k+1} / \gamma_{k+1}
\end{array}
$$

end for

and sign of α_k and γ_k. We choose to set $\alpha_k = \gamma_k > 0$.

Let us characterize the subspace which each of the bases specified in SIMBA-SC and SIMBA-NS span. For notational convenience, let us denote by T either the Schur complement in the case of SIMBA-SC or the generalized reduced Hessian in the case of SIMBA-NS. That is,

$$
T \equiv \begin{cases} S, \text{ defined in (2), if SIMBA-SC is considered,} \\ R, \text{ defined in (3), if SIMBA-NS is considered.} \end{cases} \tag{14}
$$

THEOREM 3. *Let T denote either S or R, as specified in (14). Let $\beta_1 v_1 = b$ and $\delta_1 z_1 = c$. Then the basis vectors generated in Algorithm 1 and Algorithm 2 satisfy*

$$
\begin{aligned}
v_{2k} &\in \operatorname{span}\left\{b, T^T T b, \ldots, (T^T T)^{k-1} b, T^T c, T^T T T^T c, \ldots, (T^T T)^{k-1} T^T c\right\}, \\
v_{2k+1} &\in \operatorname{span}\left\{b, T^T T b, \ldots, (T^T T)^{k} b, T^T c, T^T T T^T c, \ldots, (T^T T)^{k-1} T^T c\right\}, \\
z_{2k} &\in \operatorname{span}\left\{c, T T^T c, \ldots, (T T^T)^{k-1} c, T b, T T^T T b, \ldots, (T T^T)^{k-1} T b\right\}, \\
z_{2k+1} &\in \operatorname{span}\left\{c, T T^T c, \ldots, (T T^T)^{k} c, T b, T T^T T b, \ldots, (T T^T)^{k-1} T b\right\}.
\end{aligned}
$$

For SIMBA-SC the basis vectors satisfy

$$
u_k \in \operatorname{span}\left\{A^{-1} G_1^T V_k\right\}; \quad w_k \in \operatorname{span}\left\{A^{-T} G_2^T Z_k\right\},
$$

and for SIMBA-NS the basis vectors satisfy

$$
u_k \in \operatorname{span}\left\{H_2 V_k\right\}; \quad w_k \in \operatorname{span}\left\{H_1 Z_k\right\}.
$$

Proof. The result follows by induction on k. ∎

Notice that these spaces are not quite Krylov subspaces, but rather an interleaving of two Krylov subspaces related to SS^T and $S^T S$ in the case of SIMBA-SC, and an interleaving of two Krylov subspaces related to RR^T and $R^T R$ for SIMBA-NS. Each iteration alternates between an application of S or S^T in one case and R or R^T in the other, rather than repeated applications of the same operator.

2.5. Relationship to Orthogonal Tridiagonalization of the Schur Complement. We demonstrate that in exact arithmetic SIMBA-SC applied to $\mathcal{K}$ is mathematically equivalent to applying orthogonal tridiagonalization to the Schur complement, $S = G_2 A^{-1} G_1^T$. It is worth stressing that in ill-conditioned cases, as we show in the numerical experiments, SIMBA-SC may be more numerically stable than directly applying orthogonal tridiagonalization to the Schur complement. This result is analogous to the way in which applying Golub-Kahan is more numerically stable than applying Lanczos to the normal equations [11, 24].

Recall that orthogonal tridiagonalization generates two orthogonal bases V_k^Q, Z_k^Q such that

$$(Z_{k+1}^Q)^T S V_k^Q = \bar{T}_k,$$

where $\bar{T}_k \in \mathbb{R}^{(k+1)\times k}$ is tridiagonal. It was further shown in [27] that Z^Q and V^Q (and therefore $\bar{T}_k$) are unique up to the choice of v_1^Q and z_1^Q.

Suppose that $v_1 = v_1^Q$ and $z_1 = z_1^Q$. Using V_k and Z_k generated by SIMBA-SC, we have that

$$\begin{aligned} SV_k &= G_2 A^{-1} G_1^T V_k \\ &= G_2 A^{-1} A U_k J_k L_k^T \\ &= G_2 U_k J_k L_k^T \\ &= Z_{k+1} C_k J_k L_k^T. \end{aligned}$$

Since C_k and L_k^T are lower and upper bidiagonal respectively, and J_k is diagonal, then $C_k J_k L_k^T$ is tridiagonal. Therefore by [27, Theorem 1], this is the unique tridiagonalization of S, and thus $Z_k = Z_k^Q$, $V_k = V_k^Q$ and $\bar{T}_k = C_k J_k L_k^T$.

Note that the above also applies to SIMBA-NS, as it is equivalent to orthogonal tridiagonalization of the generalized reduced Hessian. This equivalence between orthogonal tridiagonalization and SIMBA will allow us to explore relationships between members of the SPMR family and existing iterative methods.

3. SPMR-SC: an A-inversion Version of SPMR.

We are now ready to derive members of the SPMR family, which rely on the SIMBA process. We will start with the version that involves inversion of A. Suppose indeed that A is invertible. Armed with Algorithm 1, we can observe the following relations. Define

$$K_k = \begin{pmatrix} J_k & L_k^T \\ C_k & 0 \end{pmatrix}, \tag{15}$$

and note that

$$\begin{pmatrix} A & G_1^T \\ G_2 & 0 \end{pmatrix} \begin{pmatrix} U_k & 0 \\ 0 & V_k \end{pmatrix} = \begin{pmatrix} AU_k J_k & 0 \\ 0 & Z_{k+1} \end{pmatrix} \begin{pmatrix} J_k & L_k^T \\ C_k & 0 \end{pmatrix}. \tag{16}$$

As mentioned at the outset of Section 2, if A is assumed (easily) invertible and we pursue a method based on using A^{-1}, then it makes sense to consider a right-hand side vector of the form $(0^T, g^T)^T$. Let the iterates be $x_k = U_k \bar{x}_k$ and $y_k = V_k \bar{y}_k$, so that

$$\begin{aligned} \begin{pmatrix} A & G_1^T \\ G_2 & 0 \end{pmatrix} \begin{pmatrix} x_k \\ y_k \end{pmatrix} - \begin{pmatrix} 0 \\ g \end{pmatrix} &= \begin{pmatrix} A & G_1^T \\ G_2 & 0 \end{pmatrix} \begin{pmatrix} U_k & 0 \\ 0 & V_k \end{pmatrix} \begin{pmatrix} \bar{x}_k \\ \bar{y}_k \end{pmatrix} - \begin{pmatrix} 0 \\ g \end{pmatrix} \\ &= \begin{pmatrix} AU_k J_k & 0 \\ 0 & Z_{k+1} \end{pmatrix} \left(K_k \begin{pmatrix} \bar{x}_k \\ \bar{y}_k \end{pmatrix} - \begin{pmatrix} 0 \\ \delta_1 e_1 \end{pmatrix} \right). \end{aligned}$$

It is then reasonable to adopt a quasi-minimum residual approach [13] and choose x_k and y_k which satisfy

$$\min_{x,y} \left\| K_k \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - \begin{pmatrix} 0 \\ \delta_1 e_1 \end{pmatrix} \right\| \quad \text{s.t.} \quad x = U_k \bar{x}, \;\; y = V_k \bar{y}. \tag{17}$$

3.1. Construction of Short Recurrences. We now make some observations about the subproblem for generating $\bar{x}_k$ and $\bar{y}_k$. In order to solve subproblem (17) we use the QR decomposition of K_k (defined in (15)). Note that if we permute the blocks of K_k to

$$\begin{pmatrix} L_k^T & J_k \\ 0 & C_k \end{pmatrix},$$

the above matrix is almost upper-triangular, except that we need to form the QR decomposition of C_k. Therefore, we can solve for x_k first, and recover y_k afterwards, so that an equivalent subproblem to (17) is

$$\min_{x} \| C_k \bar{x} - \delta_1 e_1 \| \quad \text{s.t.} \quad x = U_k \bar{x}. \tag{18}$$

Subproblem (18) is similar to the LSQR subproblem, which is solved by taking the QR factorization of a bidiagonal system. Many of the following recurrence relations for recovering x_k can be found in [24].

3.2. Recurrence for x_k. We begin computing the QR decomposition of C_k using the 2×2 reflector

$$\begin{pmatrix} c_1 & s_1 \\ s_1 & -c_1 \end{pmatrix} \begin{pmatrix} \gamma_1 & & \delta_1 \\ \delta_2 & \gamma_2 & \end{pmatrix} = \begin{pmatrix} \rho_1 & \sigma_1 & \phi_1 \\ & \bar{\rho}_2 & \bar{\phi}_2 \end{pmatrix},$$

and further reflectors defined by

$$\begin{pmatrix} c_k & s_k \\ s_k & -c_k \end{pmatrix} \begin{pmatrix} \bar{\rho}_k & & \bar{\phi}_k \\ \delta_{k+1} & \gamma_{k+1} & \end{pmatrix} = \begin{pmatrix} \rho_k & \sigma_{k+1} & \phi_k \\ & \bar{\rho}_{k+1} & \bar{\phi}_{k+1} \end{pmatrix}.$$

From this we obtain the QR decomposition

$$[M_{k+1} \;\; \delta_1 e_1] = Q_k \begin{pmatrix} \rho_1 & \sigma_2 & & & & \phi_1 \\ & \rho_2 & \sigma_3 & & & \phi_2 \\ & & \ddots & \ddots & & \vdots \\ & & & \rho_k & \sigma_{k+1} & \phi_k \\ & & & & \bar{\rho}_{k+1} & \bar{\phi}_{k+1} \end{pmatrix} = Q_k \begin{pmatrix} R_k & \sigma_{k+1} e_k & \varphi_k \\ & \bar{\rho}_{k+1} & \bar{\phi}_{k+1} \end{pmatrix}.$$

We define $\varphi_k = (\phi_1, \ldots, \phi_k)^T$ and $\bar{Q}_k$ as the first k columns of Q_k, so that $\bar{x}_k = R_k^{-1} \bar{Q}_k^T \delta_1 e_1$. Then, if we define $D_k = U_k R_k^{-1}$, we have

$$x_k = U_k \bar{x}_k = (U_k R_k^{-1})(\bar{Q}_k^T \delta_1 e_1) = D_k \varphi_k = x_{k-1} + \phi_k d_k.$$

Computation of d_k is accomplished via forward substitution, since

$$(u_1, \ldots, u_{k-1}, u_k) = (d_1, \ldots, d_{k-1}, d_k) \begin{pmatrix} \rho_1 & \sigma_2 & & \\ & \rho_2 & \ddots & \\ & & \ddots & \sigma_k \\ & & & \rho_k \end{pmatrix},$$

so that $d_k = (u_k - \sigma_k d_{k-1})/\rho_k$. As done in LSQR, these recurrence relations can be further simplified if we define $d_k \leftarrow \rho_k d_k$.

3.3. Recurrence for y_k. We can recover y_k with a little bit of extra work every iteration, rather than recovering y at termination. Define $T_k = (t_1, \dots, t_k)$, so that

$$
\begin{aligned}
y_k = V_k \bar{y}_k &= -V_k L_k^{-T} J_k \bar{x}_k \\
&= (V_k L_k^{-T} J_k R_k^{-1})(-\bar{Q}_k^T \delta_1 e_1) \\
&= T_k(-\varphi_k) \\
&= y_{k-1} - \phi_k t_k.
\end{aligned}
$$

Since J_k and ϕ_k are already computed, we need only compute T_k. Define

$$
R_k J_k L_k^T = \begin{pmatrix} \lambda_1 & \mu_2 & \nu_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \lambda_{k-2} & \mu_{k-1} & \nu_k \\ & & & \lambda_{k-1} & \mu_k \\ & & & & \lambda_k \end{pmatrix},
$$

which is updated column by column every iteration, since R_k and L_k^T are upper bidiagonal. In particular, the recurrence relations are

$$
\begin{aligned}
\lambda_k &= \rho_k \xi_k \alpha_k, & k &\geqslant 1, \\
\mu_k &= \rho_{k-1} \xi_{k-1} \beta_k + \sigma_k \xi_k \alpha_k, & k &\geqslant 2, \\
\nu_k &= \sigma_{k-1} \xi_{k-1} \beta_k, & k &\geqslant 3.
\end{aligned}
$$

Since $V_k = T_k(R_k J_k L_k^T)$, we have

$$
\big(v_1, \dots, v_{k-2}, v_{k-1}, v_k\big) = \big(t_1, \dots, t_{k-2}, t_{k-1}, t_k\big) \begin{pmatrix} \lambda_1 & \mu_2 & \nu_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \lambda_{k-2} & \mu_{k-1} & \nu_k \\ & & & \lambda_{k-1} & \mu_k \\ & & & & \lambda_k \end{pmatrix},
$$

which means that $t_k = (v_k - \mu_k t_{k-1} - \nu_k t_{k-2})/\lambda_k$.

3.4. Estimating the Residual. We can estimate the residual at every iteration cheaply. Define $\bar{r}_k = \delta_1 e_1 - C_k \bar{x}_k$, and $r_k = Z_{k+1} \bar{r}_k$, and note that by the definition of $\bar{y}_k$,

$$
\begin{aligned}
\begin{pmatrix} 0 \\ g \end{pmatrix} - \begin{pmatrix} A & G_1^T \\ G_2 & 0 \end{pmatrix} \begin{pmatrix} x_k \\ y_k \end{pmatrix} &= \begin{pmatrix} AU_k J_k & 0 \\ 0 & Z_{k+1} \end{pmatrix} \left(\begin{pmatrix} 0 \\ \delta_1 e_1 \end{pmatrix} - \begin{pmatrix} J_k & L_k^T \\ C_k & 0 \end{pmatrix} \begin{pmatrix} \bar{x}_k \\ \bar{y}_k \end{pmatrix} \right) \\
&= \begin{pmatrix} AU_k J_k & 0 \\ 0 & Z_{k+1} \end{pmatrix} \begin{pmatrix} 0 \\ \bar{r}_k \end{pmatrix} \\
&= \begin{pmatrix} 0 \\ r_k \end{pmatrix}.
\end{aligned} \tag{19}
$$

Since Z_k is orthogonal, the norm of the full residual is equal to $\|\bar{r}_k\| = \|r_k\|$.

The immediate consequence is that since $\|\bar{r}_k\|$ decreases monotonically by the definition of subproblem (18), the full residual must decrease monotonically as well. We summarize this result in the following theorem.

THEOREM 4. *The norm of the residual given on the left hand side of* (19) *decreases monotonically every iteration of SPMR-SC.*

Since the residual norm is equal to $\|G_2 x_k - g\|$, we can estimate the residual as $\|\bar{r}_k\| = \bar{\phi}_{k+1} = \delta_1 s_1 s_2 \dots s_k$, as is done in LSQR.

Monotonicity of the residual is an attractive property for nonsymmetric problems, as it may provide a notion of robustness and predictability. There is a potential advantage here from a computational point of view: short recurrences are not given up as in GMRES [26] to acquire this monotonicity, nor do the short recurrences give up the monotonicity as in biconjugate based methods.

3.5. Relationship Between SPMR-SC and USYMQR.

In subsection 2.5, we showed the mathematical equivalence between SIMBA and orthogonal tridiagonalization. Using this, we can now show that SPMR is equivalent to USYMQR applied to the Schur complement system $-Sy = g$.

Recall that both SIMBA and orthogonal tridiagonalization generate the same basis (in exact arithmetic) such that

$$SV_k = Z_{k+1}\bar{T}_k = Z_{k+1} C_k J_k L_k^T,$$

where $\bar{T}_k = C_k J_k L_k^T \in \mathbb{R}^{(k+1)\times k}$ is tridiagonal. USYMQR solves the subproblem

$$y_k^Q = \arg\min_y \| - C_k J_k L_k^T \bar{y} - \delta_1 e_1\| \text{ s.t. } y = V_k \bar{y}.$$

Recall that $\bar{x}_k = -J_k L_k^T \bar{y}_k$ in SPMR-SC, and recall that (from (18)) SPMR-SC solves

$$y_k = \arg\min_y \|C_k \bar{x} - \delta_1 e_1\| \text{ s.t. } \bar{x} = -J_k L_k^T \bar{y},\ y = V_k \bar{y}.$$

These are the same subproblems, and so we obtain that $y_k = y_k^Q$ every iteration, meaning that SPMR-SC and USYMQR generate the same iterates in exact arithmetic.

This result is analogous to the equivalence between LSQR and CG on the normal equations [24], or LSMR and MINRES on the normal equations [11]. However, numerically we may have the upper hand. As in the cases just mentioned, we observe that SPMR-SC can be more numerically stable than USYMQR applied an ill-conditioned Schur complement, which we demonstrate in section 8.

4. SPMR-NS: a Null-Space Based Version of SPMR.

SPMR-SC as it has been introduced so far, requires the inversion of the matrix A. This matrix may not always be invertible, and even when it is, the inversion may be computationally prohibitive. We now introduce a sub-family of SPMR which avoids inverting A, and instead opts for using the null spaces of G_1 and G_2. 'NS' stands for null-space, since we are projecting onto the null spaces of G_1 and G_2.

SPMR-NS is basically SPMR-SC applied to the dual system (8). What makes it interesting is the fact that by using the dual system we are able to eliminate dependence on the inversion of A, and instead rely on the null spaces of G_1 and G_2.

We can define the same subproblem on the dual saddle-point system to minimize the residual (of the dual system), and use the same recurrences to obtain approximations p_k and q_k at each iteration.

It should be noted that this method will only obtain approximations to $x_k = -p_k$ at every iteration, but y needs to be recovered after convergence by solving a least-squares problem with G_1^T. This is consistent with the situation in PPCG and other projected methods [16, 17].

SPMR-NS is thus equivalent to USYMQR applied to the generalized reduced Hessian defined in (3), for the same reasons that SPMR-SC is equivalent to USYMQR applied to the Schur complement. We note that in [1, 4], iterative procedures for symmetric systems are proposed, which apply the conjugate gradient method to various constructions of the reduced Hessian. This is related to SPMR-NS, which in the symmetric case is equivalent to applying MINRES to the reduced Hessian.

4.1. Estimating the Residual. Just as in SPMR-SC, the residual norm in the dual saddle-point system can be estimated cheaply. Define

$$(20)\quad \begin{pmatrix} 0 \\ r_k^N \end{pmatrix} = \begin{pmatrix} 0 \\ -H_1^T f \end{pmatrix} - \begin{pmatrix} A & AH_2 \\ H_1^T A & 0 \end{pmatrix} \begin{pmatrix} p_k \\ q_k \end{pmatrix} \text{ and } \begin{pmatrix} r_k \\ 0 \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix} - \begin{pmatrix} A & G_1^T \\ G_2 & 0 \end{pmatrix} \begin{pmatrix} x_k \\ y_k \end{pmatrix},$$

as the dual and original residuals respectively. The zero block in the dual residual follows from a derivation almost identical to (19). The zero block in the original residual follows from the fact that $x_k \in \ker(G_2)$ for all k.

We can relate $\|r_k^N\|$ to an energy semi-norm of r_k, where the semi-norm is in fact a norm on the null-space of G_1. We'll see that $r_k \in \ker(G_1)$, and therefore if $r_k^N \to 0$, this will imply that $r_k \to 0$. This is captured in the following theorem.

THEOREM 5. *Let p_k and q_k be generated by SPMR-NS. Suppose $x_k = -p_k$ and let y_k solve the least-squares problem $G_1^T y = f - Ax_k$. Define the residuals as in (20). Then*

$$\|r_k^N\| = |r_k|_{H_1 H_1^T},$$

where $|\cdot|_{H_1 H_1^T}$ is a semi-norm defined by $|u|_{H_1 H_1^T} = \left(u^T (H_1 H_1^T) u\right)^{\frac{1}{2}}$. In particular, $r_k \in \ker(G_1)$, and so this energy semi-norm induces a valid norm on the residuals.

Proof. We have

$$\begin{aligned} \|r_k^N\| &= \| - H_1^T f - H_1^T A p_k\| \\ &= \left\|H_1^T (f - Ax_k)\right\| \\ &= \left\|H_1^T \left(f - Ax_k - G_1^T y_k\right)\right\| \\ &= \|H_1^T r_k\| \\ &= |r_k|_{H_1 H_1^T}, \end{aligned}$$

where we used that $G_1 H_1 = 0$. Now, since y_k is defined by the least-squares solution to $G_1^T y = f - Ax_k$, the residual must be orthogonal to the range space of G_1^T, which means that $r_k \in \ker(G_1)$. Since $r_k \in \ker(G_1)$, then $r_k^N \to 0$ implies $r_k \to 0$, which means that the semi-norm is in fact a valid norm on the residual. ∎

Thus, even though we do not have access to the ℓ_2-norm of the original residual, we can obtain a measure of convergence using the residual norm of the dual system. Furthermore, as discussed in the following section, many of the approaches for computing projections (matrix vector products with H_i and H_i^T) result in $H_1 H_1^T$ being an orthogonal projector onto the null space of G_1. In such cases, we will have the desired property that $\|r_k^N\| = \|r_k\|$.

4.2. Computing Projections onto the Null-Space. SPMR-NS has the attractive feature that it does not require A inversion. On the other hand, it does require some knowledge of the null spaces of the off-diagonal blocks, G_1 and G_2. In this section we discuss strategies for dealing with matrix-vector products with these null-spaces.

The simplest approach is to have a null-space bases H_i available for each off-diagonal block G_i, $i = 1, 2$. Then products of the form $H_i c$, and $H_i^T c$ can be computed explicitly, and SPMR-NS can be carried out exactly as SPMR-SC would be applied to the dual saddle-point system. Although this would be the simplest approach to implementing SPMR-NS, it may be expensive to compute a null-space basis, and this basis would likely be dense.

Another possibility is to use the method outlined in [16], by computing an orthogonal projection. That is, matrix-vector products of the form $H_i c$ and $H_i^T c$ are replaced by $(I - G_i^T (G_i G_i^T)^{-1} G_i)c$. This requires one solve against $G_i G_i^T$ per application, which is only of size $m \times m$, and is therefore manageable in many applications.

An equivalent approach to computing the same orthogonal projector is to instead solve a system involving a constraint preconditioner [22]. In order to compute products of the form $d = (I - G_i^T (G_i G_i^T)^{-1} G_i)c$, we can instead solve the system

$$\begin{pmatrix} I & G_i^T \\ G_i & 0 \end{pmatrix} \begin{pmatrix} d \\ * \end{pmatrix} = \begin{pmatrix} c \\ 0 \end{pmatrix}, \tag{21}$$

where we take only the first component of the solution. Although this computes the same vector, there may be more flexibility in the solution methods applied to this saddle-point system.

Since the two previous approaches to computing $H_i x$ are effectively computing the residual to the least-squares problem $G_i^T d = c$, other techniques may be employed, such as using LSQR directly as described in [28]. This may avoid conditioning issues which may occur from solving the normal equations.

It should be noted that all of the null-space basis-free approaches mentioned above which are effectively based on solving least-squares problems, implicitly produce an orthogonal projector onto the null-space of G_i. Due to this, the seminorm $|\cdot|_{H_i H_i^T}$ becomes equivalent to the ℓ_2-norm on the null-space of G_i since $H_i H_i^T$ is an orthogonal projector onto said null-space. Therefore, estimating the norm of the dual system for SPMR-NS becomes equivalent to estimating the residual norm of the original system.

5. Properties of the SPMR Solvers.

Having derived SPMR-SC and SPMR-NS, we now discuss a few useful properties of these methods. Specifically, we provide details on the circumstances of breakdowns, and discuss the issue of convergence under spectrum clustering.

5.1. Breakdowns.

As in other biconjugate methods, we have the possibility of lucky and unlucky breakdowns. Let us again use the notation T to denote either the Schur complement S if SPMR-SC is considered, or the generalized reduced Hessian R if SPMR-NS is considered. That is,

$$T \equiv \begin{cases} S, & \text{defined in (2), if SPMR-SC is considered} \\ R, & \text{defined in (3), if SPMR-NS is considered} \end{cases} \tag{22}$$

If $z_{k+1} = 0$ for some k, we can consider this as a lucky breakdown as it implies that we can reconstruct the solution to $Ty = c$ using $v_1, \ldots, v_k$. This is because

$$\begin{aligned} 0 &= c + TT^T c + \cdots + (TT^T)^{\lfloor k/2 \rfloor} c + Tb + TT^T Tb + \cdots + (TT^T)^{\lfloor (k-2)/2 \rfloor} Tb \\ &= c + T \left(T^T c + \cdots + (T^T T)^{\lfloor k/2 \rfloor - 1} T^T c + b + T^T Tb + \cdots + (T^T T)^{\lfloor (k-2)/2 \rfloor - 1} b \right) \\ &= c + T \cdot \operatorname{span}\{v_1, \ldots, v_k\}. \end{aligned}$$

If $v_{k+1} = 0$ for some k, this is a form of an unlucky breakdown since as it means that we have found a solution to the transposed system $T^T y = b$. If such a breakdown occurs, it may be possible to restart with a different v_1 to avoid this breakdown in future iterations.

Other unlucky breakdowns occur when $w_k^T A u_k \approx 0$, in the spirit of unlucky breakdowns for methods such as BiCG and QMR [10, 13, 32]. It is likely that we will be able to employ look-ahead strategies as discussed in [12, 25], although we will not further pursue this here.

5.2. Convergence Under Spectrum Clustering. The speed of convergence of SPMR-SC or SPMR-NS is related to the distribution of singular values of T. Specifically, when the singular values are clustered we may expect fast convergence that depends on the number of *distinct* singular values.

THEOREM 6. *Denote the dimension of T by t. If T has ℓ distinct singular values, Algorithm 1 or Algorithm 2 will terminate in*

$$\bar{\ell} \leqslant \min(2\ell, t)$$

steps in exact arithmetic, that is, $z_{\bar{\ell}+1} = 0$.

Proof. T is m-by-m if SPMR-SC is considered, and $(n-m)$-by-$(n-m)$ if SPMR-NS is considered. SIMBA-SC (Algorithm 1) must terminate in at most m steps and SIMBA-NS (Algorithm 2) must terminate in at most $n - m$ steps, since $z_i \in \mathbb{R}^m$ and so any $m + 1$ vectors must be linearly dependent. Suppose then that $2\ell \leqslant t$, where t is determined according to the method used.

Let the left singular vectors of T be p_i, and the right singular vectors be q_i with corresponding singular values σ_i. Then $\sigma_i p_i = T q_i$ and $\sigma_i q_i = T^T p_i$. Thus if $b = \sum_{i=1}^{\ell} \eta_i q_i$ and $c = \sum_{i=1}^{\ell} \theta_i q_i$, then

$$(T^T T)^k b = \sum_{i=1}^{\ell} \eta_i \sigma_i^{2k} q_i, \qquad (T^T T)^k T b = \sum_{i=1}^{\ell} \eta_i \sigma_i^{2k+1} p_i,$$

$$(T T^T)^k c = \sum_{i=1}^{\ell} \theta_i \sigma_i^{2k} p_i, \qquad (T^T T)^k T^T c = \sum_{i=1}^{\ell} \theta_i \sigma_i^{2k+1} q_i.$$

Thus vectors generated by applications of T and T^T, always live in the span of $\{p_1, \ldots, p_\ell, q_1, \ldots, q_\ell\}$ which has dimension at most 2ℓ. Then this means that the number of linearly independent z_i cannot grow beyond 2ℓ and therefore SIMBA-SC or SIMBA-NS must terminate in at most 2ℓ iterations. ∎

The dependence of SPMR-SC and SPMR-NS on singular values of the Schur complement or the generalized reduced Hessian, as highlighted in Theorem 6, will affect preconditioning strategies (discussed in section 7), and may make the method attractive over other Krylov methods in some instances. One potential situation where this may be beneficial is for highly non-normal T, where it is significantly easier to characterize the convergence based on singular values rather than eigenvalues [19].

6. SPQMR. As we have shown in Theorem 6, the performance of the SPMR solvers SPMR-SC and SPMR-NS depends primarily on the distribution of the singular values of the Schur complement, S, or the generalized reduced Hessian, R, respectively. In many situations the distribution of eigenvalues is better understood than the distribution of the singular values, and eigenvalue clustering may be easier to accomplish. We now introduce a variant to SPMR which we call SPQMR, whose convergence properties rely on eigenvalue distribution of either S or R. This variant requires sacrificing the monotonicity of the residual norm, but this may be a price worth paying. Like we did for SPMR, we will have two main variants: SPMR-SC and SPMR-NS. As we will show, SPQMR-SC is mathematically equivalent to QMR applied to the Schur complement, but it is numerically more stable in the sense that there is no effect akin to squaring the condition number. Similarly, SPQMR-NS is mathematically equivalent to QMR applied to the generalized reduced Hessian.

6.1. SIMBO: Simultaneous Bidiagonalization via Bi-Orthogonality.

The main difference between SPMR and SPQMR is in the bidiagonalization procedure, which replaces orthogonality of V_k and Z_k with biorthogonality. We start with the 'SC' version of SIMBO, which requires A inversion.

6.1.1. SIMBO-SC: Using A Inversion.

Suppose A is invertible, and inverting it is computationally viable. Instead of the procedure laid out for SIMBA-SC, let us construct bases U_k, V_k, W_k, and Z_k which satisfy the relations

$$
\begin{aligned}
G_1^T V_k &= A U_k J_k L_k^T, & W_k^T A U_k &= J_k,\\
G_1 W_k &= Z_{k+1} B_k, & Z_k^T V_k &= I,\\
G_2^T Z_k &= A^T W_k J_k M_k^T, &&\\
G_2 U_k &= V_{k+1} C_k, &&
\end{aligned}
\tag{23}
$$

where again, J_k is diagonal such that $(J_k)_{j,j} = \xi_j = \pm 1$. We have marked in red the quantities that have changed, compared to the original bidiagonalization procedure SIMBA-SC described in Algorithm 1 (see also (12)). Specifically, V_{k+1} and Z_{k+1} have been swapped, and the requirement that V_k and Z_k be orthogonal has been replaced by a bi-orthogonality requirement.

This modified simultaneous bidiagonalization results in Algorithm 3. Analogously to Theorem 2, it can be shown that Algorithm 3 produces the desired relations in (23). We call this procedure SIMBO-SC.

6.1.2. SIMBO-NS: Using Null Spaces of G_1 and G_2.

Suppose now that instead of inverting A, computing the null spaces of G_1 and G_2 is necessary, or preferred. As usual, let H_1 and H_2 be such that $G_1 H_1 = G_2 H_2 = 0$. Instead of the requirements for SIMBA-NS, we require:

$$
\begin{aligned}
H_2^T V_k &= U_k J_k L_k^T, & W_k^T A U_k &= J_k,\\
H_2 A W_k &= Z_{k+1} B_k, & Z_k^T V_k &= I,\\
H_1^T Z_k &= A^T W_k J_k M_k^T, &&\\
H_1 A U_k &= V_{k+1} C_k, &&
\end{aligned}
\tag{24}
$$

The changes have been marked in red, compared to Algorithm 2 and (13).

6.2. Search Subspace.

We can classify the spaces in which the bases live in Theorem 7 in a result analogous to Theorem 3.

THEOREM 7. *Define T as in (22), and let $\beta_1 v_1 = b$, $\delta_1 z_1 = c$. Then*

$$
v_k \in \operatorname{span}\left\{b, Tb, T^2 b, \ldots, T^{k-1} b\right\},
$$

$$
z_k \in \operatorname{span}\left\{c, T^T c, (T^T)^2 c, \ldots, (T^T)^{k-1} c\right\}.
$$

For SPQMR-SC we have $u_k \in \operatorname{span}\left\{A^{-1} G_1^T V_k\right\}$ and $w_k \in \operatorname{span}\left\{A^{-T} G_2^T Z_k\right\}$, whereas for SPQMR-NS we have $u_k \in \operatorname{span}\left\{H_2^T V_k\right\}$ and $w_k \in \operatorname{span}\left\{H_1^T Z_k\right\}$.

Algorithm 3 SIMBO-SC: Simultaneous Bidiagonalization via Bi-Orthogonality, Using A Inversion

INPUT: A, G_1, G_2, b, c

$v_1 \leftarrow b$	$z_1 \leftarrow c$
$\delta_1 \leftarrow \text{sgn}(v_1^T z_1)\left(\lvert v_1^T z_1\rvert\right)^{1/2}$	$\beta_1 \leftarrow \left(\lvert v_1^T z_1\rvert\right)^{1/2}$
$v_1 \leftarrow v_1/\delta_1$	$z_1 \leftarrow z_1/\beta_1$
$\hat{u}_1 \leftarrow G_1^T v_1$	$\hat{w}_1 \leftarrow G_2^T z_1$
$u_1 \leftarrow A^{-1}\hat{u}_1$	$w_1 \leftarrow A^{-T}\hat{w}_1$
$\xi_1 \leftarrow \text{sgn}(w_1^T \hat{u}_1)$	
$\alpha_1 \leftarrow \lvert w_1^T \hat{u}_1\rvert^{1/2}$	$\gamma_1 \leftarrow \alpha_1$
$u_1 \leftarrow \xi_1 u_1/\alpha_1$	$w_1 \leftarrow \xi_1 w_1/\gamma_1$
for $k = 1, 2, \ldots$ **do**	
$v_{k+1} \leftarrow G_2 u_k - \gamma_k v_k$	$z_{k+1} \leftarrow G_1 w_k - \alpha_k z_k$
$\delta_{k+1} \leftarrow \text{sgn}(v_{k+1}^T z_{k+1})\left(\lvert v_{k+1}^T z_{k+1}\rvert\right)^{1/2}$	$\beta_{k+1} \leftarrow \left(\lvert v_{k+1}^T z_{k+1}\rvert\right)^{1/2}$
$v_{k+1} \leftarrow v_{k+1}/\delta_{k+1},$	$z_{k+1} \leftarrow z_{k+1}/\beta_{k+1}$
$\hat{u}_{k+1} \leftarrow G_1^T v_{k+1}/\beta_{k+1}$	$\hat{w}_{k+1} \leftarrow G_2^T z_{k+1}/\delta_{k+1}$
$u_{k+1} \leftarrow A^{-1}\hat{u}_{k+1} - \xi_k \beta_{k+1} u_k$	$w_{k+1} \leftarrow A^{-T}\hat{w}_{k+1} - \xi_k \delta_{k+1} w_k$
$\xi_{k+1} \leftarrow \text{sgn}(w_{k+1}^T \hat{u}_{k+1})$	
$\alpha_{k+1} \leftarrow \lvert w_{k+1}^T \hat{u}_{k+1}\rvert^{1/2}$	$\gamma_{k+1} \leftarrow \alpha_{k+1}$
$u_{k+1} \leftarrow \xi_{k+1} u_{k+1}/\alpha_{k+1}$	$w_{k+1} \leftarrow \xi_{k+1} w_{k+1}/\gamma_{k+1}$
end for	

Algorithm 4 SIMBO-NS: Simultaneous Bidiagonalization via Bi-Orthogonality, using the null spaces of G_1 and G_2, namely H_1 and H_2 such that $G_1 H_1 = 0$ and $G_2 H_2 = 0$.

INPUT: A, H_1, H_2, b, c

$v_1 \leftarrow b$	$z_1 \leftarrow c$
$\delta_1 \leftarrow \text{sgn}(v_1^T z_1)\left(\lvert v_1^T z_1\rvert\right)^{1/2}$	$\beta_1 \leftarrow \left(\lvert v_1^T z_1\rvert\right)^{1/2}$
$v_1 \leftarrow v_1/\delta_1$	$z_1 \leftarrow z_1/\beta_1$
$u_1 \leftarrow H_2 v_1$	$w_1 \leftarrow H_1 z_1$
$\hat{u}_1 \leftarrow A u_1$	$\hat{w}_1 \leftarrow A^T w_1$
$\xi_1 \leftarrow \text{sgn}(w_1^T \hat{u}_1)$	
$\alpha_1 \leftarrow \lvert w_1^T \hat{u}_1\rvert^{1/2}$	$\gamma_1 \leftarrow \alpha_1$
$u_1 \leftarrow \xi_1 u_1/\alpha_1$	$w_1 \leftarrow \xi_1 w_1/\gamma_1$
for $k = 1, 2, \ldots$ **do**	
$v_{k+1} \leftarrow H_1^T \hat{u}_k - \gamma_k v_k$	$z_{k+1} \leftarrow H_2 A^T \hat{w}_1 - \alpha_k z_k$
$\delta_{k+1} \leftarrow \text{sgn}(v_{k+1}^T z_{k+1})\left(\lvert v_{k+1}^T z_{k+1}\rvert\right)^{1/2}$	$\beta_{k+1} \leftarrow \left(\lvert v_{k+1}^T z_{k+1}\rvert\right)^{1/2}$
$v_{k+1} \leftarrow v_{k+1}/\delta_{k+1}$	$z_{k+1} \leftarrow z_{k+1}/\beta_{k+1}$
$u_{k+1} \leftarrow H_2 v_{k+1}/\beta_{k+1} - \xi_k \beta_{k+1} u_k$	$w_{k+1} \leftarrow H_1 z_{k+1}/\delta_{k+1} - \xi_k \delta_{k+1} w_k$
$\hat{u}_{k+1} \leftarrow A u_{k+1}$	$\hat{w}_{k+1} \leftarrow A^T w_{k+1}$
$\xi_{k+1} \leftarrow \text{sgn}(w_{k+1}^T \hat{u}_{k+1})$	
$\alpha_{k+1} \leftarrow \lvert w_{k+1}^T \hat{u}_{k+1}\rvert^{1/2}$	$\gamma_{k+1} \leftarrow \alpha_{k+1}$
$u_{k+1} \leftarrow \xi_{k+1} u_{k+1}/\alpha_{k+1}$	$w_{k+1} \leftarrow \xi_{k+1} w_{k+1}/\gamma_{k+1}$
end for	

6.3. SPQMR-SC and SPQMR-NS. Similar to SPMR-SC, if we choose $\delta_1 v_1 = g$, Algorithm 3 produces bases which satisfy

$$\begin{pmatrix} A & G_1^T \\ G_2 & 0 \end{pmatrix}\begin{pmatrix} x_k \\ y_k \end{pmatrix} - \begin{pmatrix} 0 \\ g \end{pmatrix} = \begin{pmatrix} A & G_1^T \\ G_2 & 0 \end{pmatrix}\begin{pmatrix} U_k & 0 \\ 0 & V_k \end{pmatrix}\begin{pmatrix} \bar{x}_k \\ \bar{y}_k \end{pmatrix} - \begin{pmatrix} 0 \\ g \end{pmatrix}$$

$$= \begin{pmatrix} AU_kJ_k & 0 \\ 0 & V_{k+1} \end{pmatrix}\left(\begin{pmatrix} J_k & L_k^T \\ C_k & 0 \end{pmatrix}\begin{pmatrix} \bar{x}_k \\ \bar{y}_k \end{pmatrix} - \begin{pmatrix} 0 \\ \delta_1 e_1 \end{pmatrix}\right).$$

We can again solve the QMR subproblem

$$\min_{x,y} \left\| K_k \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} - \begin{pmatrix} 0 \\ \delta_1 e_1 \end{pmatrix} \right\| \text{ s.t. } x = U_k\bar{x},\ y = V_k\bar{y}. \tag{25}$$

which is equivalent to the subproblem

$$\min_{x} \|C_k\bar{x} - \delta_1 e_1\| \text{ s.t. } x = U_k\bar{x}. \tag{26}$$

From this point the recurrence relations for constructing x_k and y_k are the same as in subsection 3.1, as the structure of suproblem (25) has not changed.

As in (19), the residual here has a zero block, i.e., the same structure. But we can only obtain an upper bound as done in [13], because V_k is not orthogonal. This means that at the kth iteration,

$$\|r_k\| \leqslant \sqrt{k+1}\,\delta_1\, s_1 \ldots s_k.$$

For SPQMR-NS we can derive analogous results, using the dual saddle-point system and a different right hand side; details are omitted.

6.4. Comparison of SPMR to SPQMR and Relations to Other Methods. An immediate difference between SPMR and SPQMR is that Z_k and V_k are not orthogonal in SPQMR, and therefore the residual does not decrease monotonically with every iteration. Furthermore, the lack of orthogonality in the bases means that residual estimation requires an upper bound rather than an exact estimate.

The other major difference is that SPMR has convergence that depends on the clustering of singular values of the Schur complement or the generalized reduced Hessian, compared to SPQMR whose convergence depends the eigenvalues when the Schur complement or the generalized reduced Hessian are diagonalizable. This difference affects preconditioning strategies, as there can be saddle-point matrices with Schur complements whose eigenvalues are clustered (e.g., triangular matrices with constant diagonal), but with unclustered singular values. The converse is also possible (e.g., orthogonal matrices).

Similar to how SPMR-SC is equivalent to USYMQR applied to the Schur complement, SPQMR-SC can be viewed as being equivalent to QMR being applied to the Schur complement. As the relationship between orthogonal tridiagonalization and SIMBA is explored in subsection 2.5, a similar analysis can be made to show that SIMBO is unsymmetric Lanczos applied to the Schur complement. SPQMR-SC is equivalent to QMR applied to the Schur complement by an argument similar to subsection 3.5.

We also comment on the case where $\mathcal{K}$ is symmetric, with particular attention to A being symmetric positive definite. If $\mathcal{K}$ is symmetric, then both SPMR-SC and SPQMR-SC become the same method. Furthermore, if A is SPD, then it becomes a form of Generalized LSQR [3]. If A is indefinite, then our method differs from other generalized LSQR methods, which handle only the positive definite case.

	SPMR	SPQMR
monotonic residual	✓	×
short recurrence	✓	✓
bidiagonalization procedure	SIMBA	SIMBO
depends on	singular values of T	eigenvalues of T
mathematically equivalent to	USYMQR on T	QMR on T

Table 1: Comparison of properties of SPMR vs. SPQMR. The matrix T denotes either the Schur complement or the generalized reduced Hessian; see (22).

Similar observations can be made for SPQMR-NS, where the Schur complement is replaced by the generalized reduced Hessian. We note, however, that fewer analogies are available in the symmetric case, because solvers based on reduced Hessians have been explored less comprehensively than solvers associated with the Schur complement.

We summarize these observations in Table 1.

7. Preconditioning. To develop a preconditioned version of SPMR, we will need to maintain the saddle-point structure of the matrix, and this presents a few challenges. If the preconditioner is symmetric positive definite, then weighted inner products are well defined and we will directly modify the bidiagonalization procedures SIMBA and SIMBO; otherwise we will modify the operator directly and apply our methods to the preconditioned matrix.

In general, the approach will be to use right preconditioners of the form

$$\mathcal{P} = \begin{pmatrix} I & 0 \\ 0 & \mathcal{M} \end{pmatrix}. \tag{27}$$

This leads to the relationship (for the 'SC' sub-family of methods)

$$\mathcal{K}\mathcal{P}^{-1} \begin{pmatrix} U_k & 0 \\ 0 & V_k \end{pmatrix} = \begin{pmatrix} AU_kJ_k & 0 \\ 0 & Z_{k+1} \end{pmatrix} \begin{pmatrix} J_k & L_k^T \\ C_k & 0 \end{pmatrix},$$

which is achieved in two different ways, depending on whether $\mathcal{M}$ is an SPD preconditioner or not. If $\mathcal{M}$ is SPD, we modify SIMBA and SIMBO to use $\mathcal{M}^{-1}$-orthogonality in V_k and Z_k; if $\mathcal{M}$ is not SPD, then we can practically run unpreconditioned SIMBA or SIMBO on $\mathcal{K}\mathcal{P}^{-1}$. For the 'NS' sub-family, this discussion also applies, but to the dual system.

7.1. Preconditioned SIMBA. For symmetric problems with SPD preconditioners, symmetry can be retained by modifying the bidiagonalization procedure. To that end, assume that $\mathcal{M}$ is a positive definite matrix of size $m \times m$. We will describe the (right-)preconditioned SIMBA process, noting that preconditioned SIMBO is quite similar and for the sake of brevity will not be explicitly described.

Preconditioned SIMBA compared to the unpreconditioned version trades orthogonality of V_k and Z_k for $\mathcal{M}^{-1}$-orthogonality. For SIMBA-SC, the following relations are satisfied:

$$G_1^T \mathcal{M}^{-1} V_k = A U_k J_k L_k^T, \qquad W_k^T A U_k = J_k,$$
$$G_1 W_k = V_{k+1} B_k, \qquad V_k^T \mathcal{M}^{-1} V_k = I, \tag{28}$$
$$G_2^T \mathcal{M}^{-1} Z_k = A^T W_k J_k M_k^T, \qquad Z_k^T \mathcal{M}^{-1} Z_k = I,$$
$$G_2 U_k = Z_{k+1} C_k.$$

Changes from the unpreconditioned relations, (Algorithm 1 and Equation (12)), are marked in red. The resulting procedure is summarized in Algorithm 5.

Algorithm 5 Preconditioned SIMBA-SC

INPUT: A, G_1, G_2, b, c, $\mathcal{M}$

$\hat{v}_1 = b$ $\qquad$ $\hat{z}_1 = c$

$v_1 = \mathcal{M}^{-1}\hat{v}_1$ $\qquad$ $z_1 = \mathcal{M}^{-1}\hat{z}_1$

$\beta_1 = \left(\hat{v}_1^T v_1\right)^{1/2}$ $\qquad$ $\delta_1 = \left(\hat{z}_1^T z_1\right)^{1/2}$

$v_1 = v_1/\beta_1$ $\qquad$ $z_1 = z_1/\delta_1$

$\hat{u}_1 = G_1^T \mathcal{M}^{-1} v_1$ $\qquad$ $\hat{w}_1 = G_2^T \mathcal{M}^{-1} z_1$

$u_1 = A^{-1}\hat{u}_1$ $\qquad$ $w_1 = A^{-T}\hat{w}_1$

$\xi_1 = \operatorname{sgn}(w_1^T \hat{u}_1)$

$\alpha_1 = |w_1^T \hat{u}_1|^{1/2}$ $\qquad$ $\gamma_1 = \alpha_1$

$u_1 = \xi_1 u_1/\alpha_1$ $\qquad$ $w_1 = \xi_1 w_1/\gamma_1$

for $k = 1, 2, \ldots$ **do**

$\quad v_{k+1} = G_1 w_k - \alpha_k v_k$ $\qquad$ $z_{k+1} = G_2 u_k - \gamma_k z_k$

$\quad \hat{v}_{k+1} = \mathcal{M}^{-1} v_{k+1}$ $\qquad$ $\hat{z}_{k+1} = \mathcal{M}^{-1} z_{k+1}$

$\quad \beta_{k+1} = \left(v_{k+1}^T \hat{v}_{k+1}\right)^{1/2}$ $\qquad$ $\delta_{k+1} = \left(z_{k+1}^T \hat{z}_{k+1}\right)^{1/2}$

$\quad v_{k+1} = v_{k+1}/\beta_{k+1}$ $\qquad$ $z_{k+1} = z_{k+1}/\delta_{k+1}$

$\quad \hat{u}_{k+1} = G_1^T \hat{v}_{k+1}/\beta_{k+1}$ $\qquad$ $\hat{w}_{k+1} = G_2^T \hat{z}_{k+1}/\delta_{k+1}$

$\quad u_{k+1} = A^{-1}\hat{u}_{k+1} - \xi_k \beta_{k+1} u_k$ $\qquad$ $w_{k+1} = A^{-T}\hat{w}_{k+1} - \xi_k \delta_{k+1} w_k$

$\quad \xi_{k+1} = \operatorname{sgn}(w_{k+1}^T \hat{u}_{k+1})$

$\quad \alpha_{k+1} = |w_{k+1}^T \hat{u}_{k+1}|^{1/2}$ $\qquad$ $\gamma_{k+1} = \alpha_{k+1}$

$\quad u_{k+1} = \xi_{k+1} u_{k+1}/\alpha_{k+1}$ $\qquad$ $w_{k+1} = \xi_{k+1} w_{k+1}/\gamma_{k+1}$

end for

The exact same procedure is applied to SIMBA-NS, and as before, this is done for the dual system, (8); see Algorithm 6.

All recurrences applied to the resulting bidiagonal matrices carry through as described in section 3. As this is equivalent to right-preconditioning, at the end y needs to be recovered via an additional $\mathcal{M}$-solve, that is, $y \leftarrow \mathcal{M}^{-1}y$.

7.2. Preconditioned SPMR-SC and SPQMR-SC. If the preconditioner is not symmetric positive definite, then it is impractical to precondition the bidiagonalization procedures SIMBA and SIMBO directly; instead we modify the saddle-point system directly. Theorem 6 and Krylov subspace theory may be used to show that if the Schur complement has clustered singular values then SPMR-SC will converge quickly, and if it has clustered eigenvalues then SPQMR-SC will converge quickly. Furthermore, preconditioners must be block diagonal in order to maintain the saddle-point structure of the operator. Therefore, if $\tilde{S} \approx S$ is an approximation to the Schur

Algorithm 6 Preconditioned SIMBA-NS

INPUT: A, H_1, H_2, b, c, $\mathcal{M}$

$$
\begin{array}{ll}
\hat{v}_1 = b & \hat{z}_1 = c \\
v_1 = \mathcal{M}^{-1}\hat{v}_1 & z_1 = \mathcal{M}^{-1}\hat{z}_1 \\
\beta_1 = \left(\hat{v}_1^T v_1\right)^{1/2} & \delta_1 = \left(\hat{z}_1^T z_1\right)^{1/2} \\
v_1 = v_1/\beta_1 & z_1 = z_1/\delta_1 \\
u_1 = H_2\mathcal{M}^{-1}v_1 & w_1 = H_1\mathcal{M}^{-1}z_1 \\
\hat{u}_1 = Au_1 & \hat{w}_1 = A^T w_1 \\
\xi_1 = \operatorname{sgn}(w_1^T\hat{u}_1) & \\
\alpha_1 = |w_1^T\hat{u}_1|^{1/2} & \gamma_1 = \alpha_1 \\
u_1 = \xi_1 u_1/\alpha_1 & w_1 = \xi_1 w_1/\gamma_1
\end{array}
$$

for $k = 1, 2, \ldots$ **do**

$$
\begin{array}{ll}
v_{k+1} = H_2^T\hat{w}_k - \alpha_k v_k & z_{k+1} = H_1^T\hat{u}_k - \gamma_k z_k \\
\hat{v}_{k+1} = \mathcal{M}^{-1}v_{k+1} & \hat{z}_{k+1} = \mathcal{M}^{-1}z_{k+1} \\
\beta_{k+1} = \left(v_{k+1}^T\hat{v}_{k+1}\right)^{1/2} & \delta_{k+1} = \left(z_{k+1}^T\hat{z}_{k+1}\right)^{1/2} \\
v_{k+1} = v_{k+1}/\beta_{k+1} & z_{k+1} = z_{k+1}/\delta_{k+1} \\
u_{k+1} = H_2\hat{v}_{k+1}/\beta_{k+1} - \xi_k\beta_{k+1}u_k & w_{k+1} = H_1\hat{z}_{k+1}/\delta_{k+1} - \xi_k\delta_{k+1}w_k \\
\hat{u}_{k+1} = Au_{k+1} & \hat{w}_{k+1} = A^T w_{k+1} \\
\xi_{k+1} = \operatorname{sgn}(w_{k+1}^T\hat{u}_{k+1}) & \\
\alpha_{k+1} = |w_{k+1}^T\hat{u}_{k+1}|^{1/2} & \gamma_{k+1} = \alpha_{k+1} \\
u_{k+1} = \xi_{k+1}u_{k+1}/\alpha_{k+1} & w_{k+1} = \xi_{k+1}w_{k+1}/\gamma_{k+1}
\end{array}
$$

end for

complement, then we seek left- or right-preconditioners of the form

$$\mathcal{P} = \begin{pmatrix} I & 0 \\ 0 & \tilde{S} \end{pmatrix}.$$

For right-preconditioning, this will be equivalent to using the right-preconditioned operator

$$\mathcal{K}\mathcal{P}^{-1} = \begin{pmatrix} A & G_1^T\tilde{S}^{-1} \\ G_2 & 0 \end{pmatrix}. \tag{29}$$

Computing solutions to linear systems of the form $\tilde{S}d = c$ can be performed in an alternative fashion as well using a constraint preconditioner. Using an approximation to the leading block $\tilde{A} \approx A$, we can instead compute the solution to the linear system

$$\begin{pmatrix} \tilde{A} & G_1^T \\ G_2 & 0 \end{pmatrix}\begin{pmatrix} * \\ d \end{pmatrix} = \begin{pmatrix} 0 \\ -c \end{pmatrix},$$

keeping only the second component d. We note that the key requirement here is preserving the block structure, therefore it is possible to also approximate the off-diagonal blocks G_1 and G_2. That is, it is not necessarily the case that a constraint preconditioner must be used.

7.3. Preconditioning SPMR-NS and SPQMR-NS. Since the 'NS' methods are effectively SPMR-SC and SPQMR-SC methods applied to the dual saddle-point system (8), the strategy for preconditioning is analogous to the previous section in that we want to approximate $R = H_1^T A H_2$, but instead of working with the preconditioned (primal) saddle-point system, we will work with the *preconditioned* dual saddle-point system,

$$\mathcal{K}_D \mathcal{P}^{-1} = \begin{pmatrix} A & AH_2\tilde{R}^{-1} \\ H_1^T A & 0 \end{pmatrix}. \tag{30}$$

If null-space bases H_1 and H_2 are given, then it is feasible to construct such an approximation, but such an approach would be difficult if H_1 and H_2 are implicit operators or if they are not easily available.

We start our quest for designing a preconditioner for the NS sub-family by assuming that H_1 and H_2 are available and have full rank. This requirement will be eliminated later on. Consider the *ideal* preconditioner $\tilde{R} = H_1^T A H_2$, so that the preconditioned dual saddle-point matrix (30) can now be written as follows:

$$\begin{pmatrix} A & AH_2(H_1^T A H_2)^{-1} \\ H_1^T A & 0 \end{pmatrix}. \tag{31}$$

We say that this choice of $\tilde{R}$ gives an ideal preconditioner because the Schur complement of the above matrix is the identity. Since we are interested in a strongly clustered spectrum for the Schur complement, this observation is useful as a starting point for designing a preconditioner. Of course, the (1,2)-block cannot be easily computed and we need to find ways to alleviate this difficulty. First, if $\tilde{A} \approx A$ is an approximation for the leading block, we can make the representation more practical. Next, we can instead consider computing matrix vector products of the form

$$d = H_2(H_1^T \tilde{A} H_2)^{-1} H_1^T c. \tag{32}$$

If we compare (32) to the (1,2)-block of (31), we observe that main difference is in a pre-multiplication by H_1^T and the post-multiplication of A which is trivial to apply. Systems such as in (32) can be relatively easily computed by solving the constraint preconditioner system

$$\begin{pmatrix} \tilde{A} & G_1^T \\ G_2 & 0 \end{pmatrix} \begin{pmatrix} d \\ * \end{pmatrix} = \begin{pmatrix} c \\ 0 \end{pmatrix}. \tag{33}$$

To see this, notice that the matrix in (32) is precisely equal to the leading block of the inverse of the matrix in (33) [5, 9]. Thus it is no longer necessary to have H_1 and H_2 available explicitly; we can accomplish computation of d by solving a constraint preconditioner.

8. Applications and Numerical Experiments.

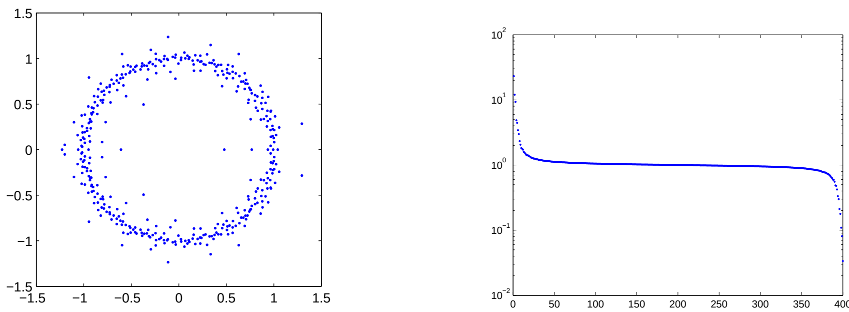
In this section we numerically illustrate the features of SPMR and its variants.

8.1. Nearly-Orthogonal Schur Complement.

We begin with an example of the performance of members of the 'SC' family, highlighting the distinction between having well clustered singular values and well clustered eigenvalues for the Schur complement. We generate the system

$$\mathcal{K}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} A & G_1^T \\ QG_2 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ g \end{pmatrix}, \tag{34}$$

where $n = 700$, $m = 400$, g is random, A is a nonsymmetric diagonally dominant sparse random matrix, G_1, G_2 are sparse random matrices, and Q is a random orthogonal matrix. The sparse matrices were generated via MATLAB's *sprand*, with a density of 0.1, and Q was generated via the QR factorization of a random matrix. A is made diagonally dominant by adding a multiple of the identity.



(a) Eigenvalues in the complex plane of the preconditioned Schur complement of problem (34). For convenient visualization purposes, a small number of the larger eigenvalues are excluded from the figure.

(b) Singular values of the preconditioned Schur complement of problem (34).

Fig. 2: Spectrum of preconditioned Schur complement of problem (34) in Subsection 8.1.

Since A is diagonally dominant, a reasonable approximation to the Schur complement is

$$\tilde{S} = G_2 D^{-1} G_1^T$$

where D is the diagonal of A. We can thus write $QG_2A^{-1}G_1^T\tilde{S}^{-1} \approx Q$, which means that the Schur complement would have a well distributed spectrum of singular values, while the eigenvalues would be spread around the unit circle in the complex plane. Recall that SPMR-SC rapidly converges when the singular values of the Schur complement are strongly clustered. Solvers whose convergence rate depends on eigenvalues may not perform as well in this case.

We plot the eigenvalues in the complex plane in Figure 2a, and the singular values on a semilog plot in Figure 2b, which confirm our claim for this example.

Consider the right preconditioners

$$\mathcal{P}_1 = \begin{pmatrix} I & 0 \\ 0 & \tilde{S} \end{pmatrix} \quad \text{and} \quad \mathcal{P}_2 = \begin{pmatrix} A & 0 \\ 0 & \tilde{S} \end{pmatrix}. \tag{35}$$

We compare the performance of SPMR-SC and SPQMR-SC, where we use the preconditioner $\mathcal{P}_1$, and GMRES where we use the preconditioner $\mathcal{P}_2$. The results are presented in Figure 3, where we track the residual norm per iteration.

As expected, SPMR-SC converges quickly due to well clustered singular values. On the other hand SPQMR-SC and GMRES are not competitive since the eigenvalues of the Schur complement are spread around the complex unit circle. GMRES takes exactly $2m + 1$ iterations, since it's applied to the operator

$$\mathcal{K}\mathcal{P}_2^{-1} = \begin{pmatrix} I & G_1^T\tilde{S} \\ QG_2A^{-1} & 0 \end{pmatrix},$$

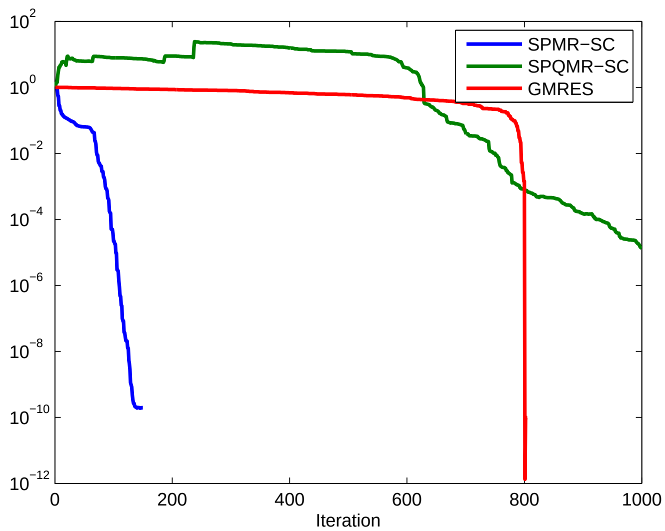


Fig. 3: $\|r_k\|$ for problem (34) of Subsection 8.1.

whose eigenvalues are 1 (with algebraic multiplicity $n-m$) and the other $2m$ eigenvalues are $\pm\lambda$ where λ is an eigenvalue of the Schur complement of the above operator, $QG_2A^{-1}G_1^T\tilde{S}$, which are not clustered.

8.2. Highly Non-Normal Generalized Reduced Hessian. We show an example where SPMR-NS outperforms typical Krylov methods in terms of convergence behavior of the residual norm. In this case we take a saddle-point matrix such that the leading block A is an $n \times n$ Grcar matrix [30, Ch. 7], and the off-diagonal blocks $G_1 = G_2 = \begin{pmatrix} F_1 & F_2 \end{pmatrix}$, with $F_1, F_2 \in \mathbb{R}^{\frac{n}{2} \times \frac{n}{2}}$, and $\|F_1\|_2 \gg \|F_2\|$. We choose $n = 1000$, and take the right-hand side to be of the form $(f^T, 0^T)^T$ with f random.

We run unpreconditioned SPMR-SC and SPQMR-SC, where we use the null-space matrices

$$H_1 = H_2 = \begin{pmatrix} F_1^{-1}F_2 \\ -I \end{pmatrix}.$$

For the purpose of comparison, we run GMRES and LSQR preconditioned with

$$\mathcal{P} = \begin{pmatrix} I & G_1^T \\ G_2 & 0 \end{pmatrix}.$$

We use the constraint preconditioner due to its relationship to projections onto the null-space of the off-diagonal blocks. Thus, we can now talk about comparable iterates in terms of projections onto the null-space. The norm of the residual is plotted in Figure 4.

It is known that nonsymmetric Krylov subspace methods may suffer on highly non-normal matrices such as the Grcar matrix [30]. Since $\|F_1\| \gg \|F_2\|$, most of the mass of the null-space basis is in the identity block. This means that the generalized reduced Hessian exhibits spectral behaviour similar to A. We can see in Figure 4 that LSQR has trouble converging, and GMRES and SPQMR-NS which depend eigenvalues do not converge too quickly. On the other hand, we see that SPMR-NS has

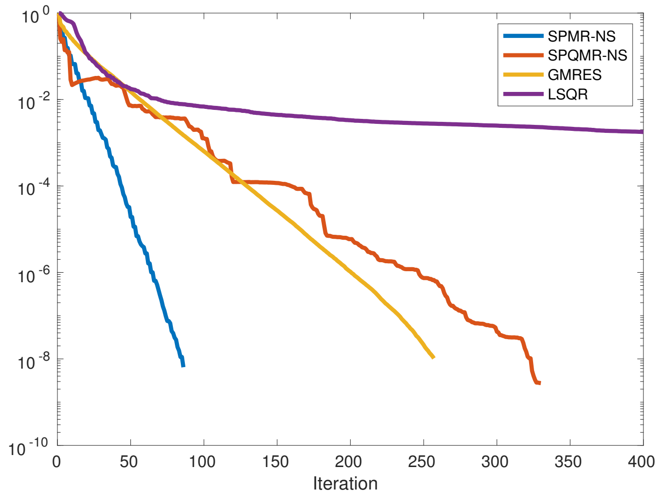


Fig. 4: $\|r_k\|$ for the problem of Subsection 8.2.

fast convergence, since it depends on the singular values of the generalized reduced Hessian.

8.3. Effect of Conditioning on SPMR-SC. We next demonstrate the strong performance of SPMR-SC in comparison with solvers that work directly on the Schur complement. As we have shown in subsection 3.5, SPMR-SC works on the entire saddle-point system but is mathematically equivalent to USYMQR applied to the Schur complement system $Sy = -g$.

Consider the saddle-point system

$$\mathcal{K}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} A & G_1^T \\ G_2 & 0 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ g \end{pmatrix}, \tag{36}$$

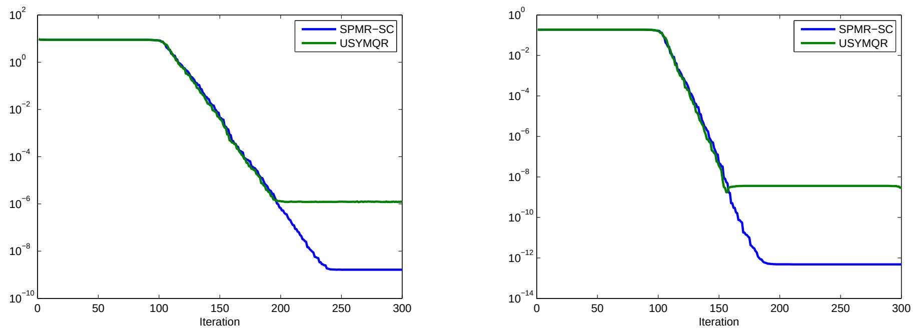
where in this case, $n = 600$, $m = 300$, g is random, and A is a block tridiagonal matrix of the form

$$A = \begin{pmatrix} B & -I & & & \\ -I & B & -I & & \\ & \ddots & \ddots & \ddots & \\ & & -I & B & -I \\ & & & -I & B \end{pmatrix},$$

with

$$B = \begin{pmatrix} 4 & -1+\delta & & & \\ -1-\delta & 4 & -1+\delta & & \\ & \ddots & \ddots & \ddots & \\ & & -1-\delta & 4 & -1+\delta \\ & & & -1-\delta & 4 \end{pmatrix},$$

where $\delta = 0.1$. The matrix A is a finite difference discretization of a simple 2D convection-diffusion equation with constant coefficients on the unit square. G_1 is a



(a) $\|r_k\|$ for problem (36).   (b) $\|y_* - y_k\|$ for problem (36).

Fig. 5: Performance of SPMR versus USYMQR on problem (36).

random matrix whose condition number has been set to be $\kappa(G_1) = 10^5$, while G_2 is a random perturbation of G_1 so that it has a similar condition number. This results in $\kappa(S) \approx 10^8$. The exact solution x_* and y_* is obtained via MATLAB's backslash operator.

In Figure 5a and Figure 5b we see the residual and error norms at every iteration respectively. It is clear that even though in exact arithmetic the two would produce the same iterates, we we obtain 4 digits of accuracy more using SPMR-SC on the entire saddle-point system as compared to USYMQR on the Schur complement. This result is similar in spirit to the improved stability in LSQR over running CG on the normal equations [24].

We note that this property may not always manifest itself as it would in the symmetric case where A is positive definite. Since these are nonsymmetric problems, there could exist cases where it may be beneficial to form the Schur complement over working with the full saddle-point system. That being said, in cases when the Schur complement has a large condition number which is nearly the product of the condition numbers of G_1 and G_2, we would expect SPMR-SC to outperform methods that work directly on the Schur complement.

8.4. Interior-Point Methods. Constrained optimization problems provide a rich source of saddle-point systems in various forms. Consider quadratic programs and their corresponding duals, of the form

$$\min_{x} \; c^T x + \tfrac{1}{2} x^T H x \quad \text{subject to} \quad Jx = b, \; x \geqslant 0, \tag{37}$$

$$\max_{x,y,z} \; b^T y - \tfrac{1}{2} x^T H x \quad \text{subject to} \quad J^T y + z - Hx = c, \; z \geqslant 0. \tag{38}$$

One of the most popular classes of techniques for solving this problem are *interior-point methods*. They are based on relaxing the complementarity conditions by introducing a small parameter-dependent perturbation. The Newton step is 'corrected' by steering the iterate towards the so called 'central path' [23]. The extent by which this is done depends on the proximity to the solution and other considerations.

The perturbed optimality conditions are

$$\begin{pmatrix} c + Hx - J^T y - z \\ Jx - b \\ \tau e - XZe \end{pmatrix} = 0, \qquad (x, z) > 0. \tag{39}$$

The parameter τ is initially set as a small positive number and is gradually decreased towards zero as we approach the optimal solution. There are various strategies for selecting the value of τ. Solving the mildly nonlinear system (39) using Newton's method results in the linear system

$$\begin{pmatrix} H & -I & J^T \\ -Z & -X & 0 \\ J & 0 & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta z \\ -\Delta y \end{pmatrix} = \begin{pmatrix} -c - Hx + J^T y + z \\ b - Jx \\ XZe - \tau e \end{pmatrix}. \tag{40}$$

The linear system (40) is nonsymmetric. The matrices X and Z are diagonal, but they grow increasingly ill-conditioned as the solution of the optimization problem is approached, due to driving τ to zero. It is possible to symmetrize (40), but doing so requires inverting Z, and this may affect the numerical stability of the solution procedure, although the effect is subject for debate. Issues related to conditioning of the matrices involved in the interior-point linear system have been subject to extensive exploration; see, for example, [34].

We may opt to solve the linear system by forming the Schur complement, and there is more than one alternative here. In [20] a comprehensive study was conducted on the condition number (40) and reduced versions based on block Gaussian elimination. It was shown that from a conditioning point of view, the unreduced 3-by-3 form is more robust near the optimal solution, compared to reduced versions.

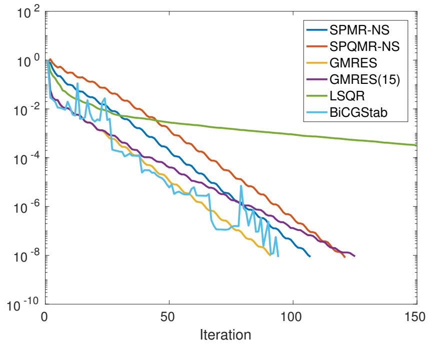
Forming the Schur complement may yield a highly ill-conditioned matrix, and the inversion of the leading block in this case may be computationally prohibitive, especially if the Hessian H is hard to deal with computationally (note that it may often be indefinite). We thus resort to using null spaces. Since null-space methods are a popular approach to solving problems with linear constraints, it is reasonable to have a linear mapping to the null-space of J, which we will call C. In this case, we will use the orthogonal projector $C = I - J^T(JJ^T)^{-1}J$. We also modify the right-hand side by finding a particular solution Δx_0 such that $J\Delta x_0 = XZe - \tau e$, so that we instead solve the system

$$\begin{pmatrix} H & -I & J^T \\ -Z & -X & 0 \\ J & 0 & 0 \end{pmatrix} \begin{pmatrix} \Delta x - \Delta x_0 \\ \Delta z \\ -\Delta y \end{pmatrix} = \begin{pmatrix} -c - Hx + J^T y + z - J\Delta x_0 \\ b - Jx \\ 0 \end{pmatrix}.$$

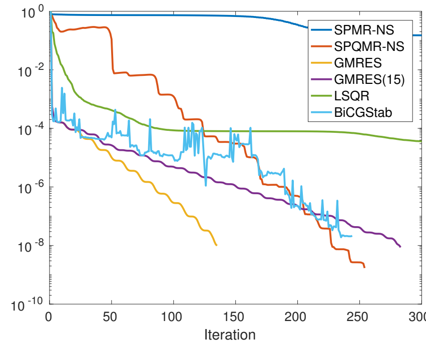
Thus we can apply SPMR-NS and SPQMR-NS with

$$H_1 = H_2 = \begin{pmatrix} C & \\ & I \end{pmatrix}.$$

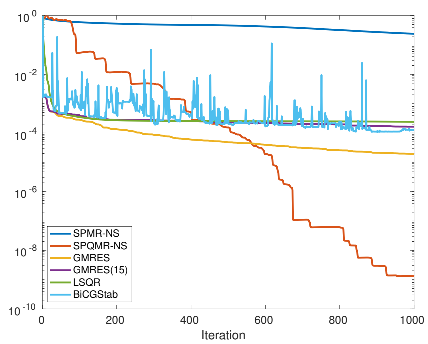
We compare SPMR-NS and SPQMR-NS against GMRES (both full and restarted with a restart of 20), LSQR and BiCGSTAB. We take the polygon100 problem from COPS [6] (in its nonnegative slack formulation), where $n = 16347$ and $m = 10700$, and construct a quadratic approximation to the nonlinear program at the initial point plus a small offset to move it off of the boundary. We can control how ill-conditioned the problem is by moving x and z close to the boundary of the bound constraints.



(a) $x = x_0 + 10^{-1}\mathbf{1}, z = 10^{-1}\mathbf{1}$.



(b) $x = x_0 + 10^{-1}\mathbf{1}, z = 10^{-2}\mathbf{1}$.



(c) $x = x_0 + 5 \cdot 10^{-3}\mathbf{1}, z = 10^{-2}\mathbf{1}$.

Fig. 6: $\|r_k\|_2$ using various values for x and z. x_0 is provided as part of the polygon100 problem. $\mathbf{1}$ denotes a vector of all ones.

We first run the iterative methods for various values of x and z which progressively make the problem more ill-conditioned. We also precondition GMRES, BiCGSTAB and LSQR with the constraint preconditioner

$$\mathcal{P} = \begin{pmatrix} I & 0 & J^T \\ 0 & I & 0 \\ J & 0 & 0 \end{pmatrix}.$$

We plot the residual norm per iteration in Figure 6 with various values of x and z. In Figure 6a, all of the methods other than LSQR are comparable in performance, as they tend to decrease the residual geometrically. SPMR-NS, SPQMR-NS, BiCGSTAB and GMRES appear to have roughly the same rate (although BiCGSTAB is highly irregular), while restarted GMRES decreases more slowly. Since SPMR-NS, SPQMR-NS and BiCGSTAB are the fastest converging short-recurrence methods, they appear appropriate for this problem.

As we make the problem more ill-conditioned in Figure 6b, we see that SPMR-NS no longer converges, and although GMRES converges the most quickly, it begins to become more expensive per iteration to do the reorthogonalization. We see SPQMR-NS converges most quickly among the short-recurrence methods, while BiCGSTAB and restarted GMRES lag a little bit behind.

In the most ill-conditioned case, we see that SPQMR-NS converges first by far, while GMRES takes significantly longer. Restarted GMRES, BiCGSTAB and LSQR stall out around $\|r_k\| \approx 10^{-4}$, while SPMR-NS has trouble converging at all. Thus we see that SPQMR-NS is the most practical method in this case.

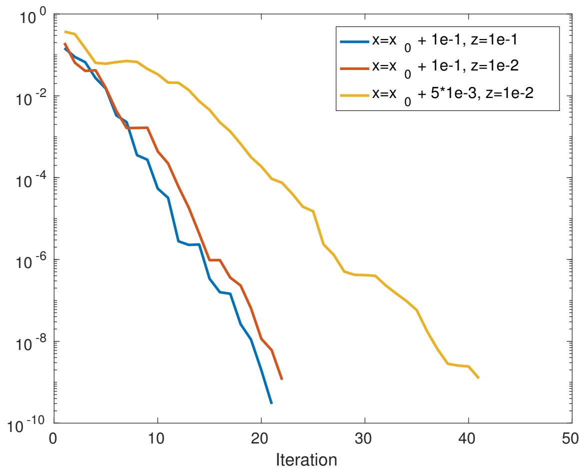


Fig. 7: $\|r_k\|$ for SPQMR-NS on the polygon100 problems from Figure 6 with preconditioning.

We now precondition SPQMR-NS by approximating the generalized reduced Hessian, to see how the convergence behaviour changes. The generalized reduced Hessian in this case is

$$R = \begin{pmatrix} C^T H C & -C^T \\ -ZC & -X \end{pmatrix}.$$

Note that with the non-negative slack formulation, H will have large zero blocks corresponding to the slack variables; therefore it is reasonable to approximate H by the identity, so that the first block is replaced by $C^T C = C^2 = C$ since C is a symmetric orthogonal projector. Therefore, we can approximate the reduced Hessian by the block triangular matrix

$$R \approx \widehat{R} = \begin{pmatrix} C + \alpha I & 0 \\ -ZC & -X \end{pmatrix},$$

where α is a small value to make $\widehat{R}$ nonsingular (we take $\alpha = 10^{-3}$). Since X is diagonal and C is an orthogonal projector, solving against this preconditioner can be done efficiently. Thus we now use the null-space operators

$$H_1 = \begin{pmatrix} C & \\ & I \end{pmatrix}, \text{ and } H_2 = H_1 \widehat{R}^{-1}.$$

The residual norm convergence history for the 3 problems is given in Figure 7. Even with a relatively simple approximation to R, we see that we can now take a fairly reasonable number of iterations to converge, which makes SPQMR-NS a potentially practical method for solving saddle-point systems arising from such optimization problems.

Problem	n	m	SPMR-NS
N_1	88	25	8
N_2	368	113	8
N_3	1504	481	8
N_4	6080	1985	8
N_5	24448	8065	8
L_1	353	98	6
L_2	634	179	6
L_3	2004	604	6
L_4	7544	2383	6

Table 2: Number of iterations for SPMR-NS for several problems to achieve relative residual norm of 10^{-10}. The N_i problems correspond to a unit square domain whereas the L_i problems correspond to L-shaped domains.

8.5. Maxwell. A simple form of time-harmonic Maxwell equations can be written as follows:

$$-\nabla \times \nabla \times u + \nabla p = f,$$
$$\nabla \cdot u = 0,$$

with appropriate boundary conditions. We point the reader to [21] for additional details. A significant challenge in solving this problem is that the discrete curl-curl operator is rank deficient, and hence the corresponding leading block of the saddle-point matrix is singular (see, for example, [8, 9] for ways to deal with a highly rank deficient leading block). For this reason SPMR-SC is not a viable candidate. On the other hand, for SPMR-NS we can exploit the fact that the null-space of the off-diagonal blocks of the matrix is explicitly known and can be expressed in a sparse fashion. We therefore examine SPMR-NS.

The computational kernels involved in using SPMR-NS and SPQMR-NS are to solve constraint preconditioners of the form

$$\begin{pmatrix} I & G^T \\ G & 0 \end{pmatrix} \begin{pmatrix} d \\ * \end{pmatrix} = \begin{pmatrix} c \\ 0 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} A+M & G^T \\ G & 0 \end{pmatrix} \begin{pmatrix} d \\ * \end{pmatrix} = \begin{pmatrix} c \\ 0 \end{pmatrix}, \tag{41}$$

where M is the vector mass-matrix.

We solve against a random right-hand side of the form $(f^T, 0)^T$, and record the number of iterations required to achieve a relative residual norm of 10^{-10}. The results are recorded in table 2.

Since this is a symmetric problem using a symmetric positive definite preconditioner, SPMR-NS and SPQMR-NS are the same method. We see that SPMR-NS shows perfect scalability with the given preconditioner.

We note that scalable solution methods based on block diagonal preconditioned MINRES do exist and perform very well [8, 21]. Here we show that SPMR is competitive with those approaches and is fully scalable too, although the preconditioner solves are slightly more computationally costly. Further connections to existing solvers such as PP-MINRES [17] may be apparent.

9. Concluding Remarks. The promise of the SPMR family is in it being a customized solver for saddle-point systems, with a monotonic and short recurrence version for the nonsymmetric case. It is significant that for the SC version, as opposed to other solvers, we effectively avoid squaring the condition number the Schur complement while implicitly forming it. It is also notable that convergence is very rapid when the singular values of the Schur complement are clustered.

SPMR on its various versions offers a novel simultaneous bidiagonalization procedure, and proves competitive with other solvers in a variety of scenarios, as we have demonstrated in our numerical experiments.

We would also like to offer some comments on inexact matrix-vector products. Considerable work has been done in the field of inexact Krylov methods, such as in [14, 18, 29, 31]. It would be beneficial to be able to use inexact A-solves (for SPMR-SC or SPQMR-SC) or inexact null-space projections (for SPMR-NS or SPQMR-NS) by using this theory. Although previous work is concerned primarily with methods based on the Arnoldi or Lanczos process [18, 29, 31], or the Golub-Kahan process [14], it should be possible to extend this work to SIMBA and SIMBO. The main disadvantage is that either short-recurrence methods become long-recurrence methods when inexact matrix-vector products are introduced as in [14], or the tolerance for how inexact the products must be made tighter [31]. Even if the methods are forced to be long-recurrence, if the iteration cost is dominated by the A-solves or null-space projects rather than reorthogonalization, investigating the use of inexactness would be advantageous, and the topic of future research.

Finally, it may be desirable to explore applying SPMR to the important class of regularized saddle-point systems.

A MATLAB version of our code is available at https://github.com/restrin/LinearSystemSolvers.

REFERENCES

[1] M. Arioli. The use of QR factorization in sparse quadratic programming and backward error issues. *SIAM J. Matrix Anal. Appl.*, 21(3):825–839, 2000.

[2] M. Arioli. Generalized Golub-Kahan bidiagonalization and stopping criteria. *SIAM J. Matrix Anal. Appl.*, 34(2):571–592, 2013.

[3] M. Arioli and D. Orban. Iterative methods for symmetric quasi-definite linear systems. Part I: Theory. Technical report, GERAD, 2013.

[4] Mario Arioli and Lucia Baldini. A backward error analysis of a null space algorithm in sparse quadratic programming. *SIAM J. Matrix Anal. Appl.*, 23(2):425–442, 2001.

[5] Michele Benzi, Gene H. Golub, and Jörg Liesen. Numerical solution of saddle point problems. *Acta Numer.*, 14:1–137, 2005.

[6] Alexander S Bondarenko, David M Bortz, and Jorge J Moré. Cops: Large-scale nonlinearly constrained optimization problems. Technical report, Argonne National Lab., IL (US), 2000.

[7] Howard C. Elman, David J. Silvester, and Andrew J. Wathen. *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics*. Numerical Mathematics and Scientific Computation. Oxford University Press, Oxford, second edition, 2014.

[8] Ron Estrin and Chen Greif. On nonsingular saddle-point systems with a maximally rank deficient leading block. *SIAM J. Matrix Anal. Appl.*, 36(2):367–384, 2015.

[9] Ron Estrin and Chen Greif. Towards an optimal condition number of certain augmented Lagrangian-type saddle-point matrices. *Numer. Linear Algebra Appl.*, 23(4):693–705, 2016.

[10] R. Fletcher. *Conjugate gradient methods for indefinite systems*, pages 73–89. Springer Berlin Heidelberg, Berlin, Heidelberg, 1976.

[11] David Chin-Lung Fong and Michael Saunders. LSMR: an iterative algorithm for sparse least-squares problems. *SIAM J. Sci. Comput.*, 33(5):2950–2971, 2011.

[12] Roland W. Freund, Martin H. Gutknecht, and Noël M. Nachtigal. An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices. *SIAM J. Sci. Comput.*, 14(1):137–158, 1993.

[13] Roland W. Freund and Noël M. Nachtigal. QMR: a quasi-minimal residual method for non-

Hermitian linear systems. *Numerische Mathematik*, 60(1):315–339, 1991.

[14] Sarah W Gaaf and Valeria Simoncini. Approximating leading singular triplets of a matrix function. *arXiv preprint arXiv:1505.03453*, 2015.

[15] G. Golub and W. Kahan. Calculating the singular values and pseudo-inverse of a matrix. *J. Soc. Indust. Appl. Math. Ser. B Numer. Anal.*, 2:205–224, 1965.

[16] Nicholas I. M. Gould, Mary E. Hribar, and Jorge Nocedal. On the solution of equality constrained quadratic programming problems arising in optimization. *SIAM J. Sci. Comput.*, 23(4):1376–1395, 2001.

[17] Nick Gould, Dominique Orban, and Tyrone Rees. Projected Krylov methods for saddle-point systems. *SIAM J. Matrix Anal. Appl.*, 35(4):1329–1343, 2014.

[18] Serge Gratton, Ph L Toint, and Jean Tshimanga. Inexact range-space krylov solvers for linear systems arising from inverse problems. *Technical Report 09/20*, 2009.

[19] Anne Greenbaum, Vlastimil Pták, and Zdeněk Strakoš. Any nonincreasing convergence curve is possible for GMRES. *SIAM J. Matrix Anal. Appl.*, 17(3):465–469, 1996.

[20] Chen Greif, Erin Moulding, and Dominique Orban. Bounds on eigenvalues of matrices arising from interior-point methods. *SIAM Journal on Optimization*, 24(1):49–83, 2014.

[21] Chen Greif and Dominik Schötzau. Preconditioners for the discretized time-harmonic Maxwell equations in mixed form. *Numer. Linear Algebra Appl.*, 14(4):281–297, 2007.

[22] Carsten Keller, Nicholas I. M. Gould, and Andrew J. Wathen. Constraint preconditioning for indefinite linear systems. *SIAM J. Matrix Anal. Appl.*, 21(4):1300–1317, 2000.

[23] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer Series in Operations Research and Financial Engineering. Springer, New York, second edition, 2006.

[24] C. C. Paige and M. A. Saunders. LSQR: An algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Softw.*, 8(1):43–71, March 1982.

[25] Beresford N. Parlett, Derek R. Taylor, and Zhishun A. Liu. A look-ahead Lánczos algorithm for unsymmetric matrices. *Math. Comp.*, 44(169):105–124, 1985.

[26] Youcef Saad and Martin H. Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 7(3):856–869, 1986.

[27] M. A. Saunders, H. D. Simon, and E. L. Yip. Two conjugate-gradient-type methods for unsymmetric linear equations. *SIAM J. Numer. Anal.*, 25(4):927–940, 1988.

[28] Michael A. Saunders. Computing projections with LSQR. *BIT*, 37(1):96–104, 1997.

[29] Valeria Simoncini and Daniel B Szyld. Theory of inexact krylov subspace methods and applications to scientific computing. *SIAM Journal on Scientific Computing*, 25(2):454–477, 2003.

[30] Lloyd N. Trefethen and Mark Embree. *Spectra and pseudospectra*. Princeton University Press, Princeton, NJ, 2005. The behavior of nonnormal matrices and operators.

[31] Jasper Van Den Eshof and Gerard LG Sleijpen. Inexact krylov subspace methods for linear systems. *SIAM Journal on Matrix Analysis and Applications*, 26(1):125–153, 2004.

[32] H. A. van der Vorst. Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 13(2):631–644, 1992.

[33] Andrew Wathen. Preconditioning. *Acta Numer.*, 24:329–376, 2015.

[34] M. H. Wright. Interior methods for constrained optimization. *Acta Numerica*, 1:341–407, 1992.