CONSTRAINT-PRECONDITIONED KRYLOV SOLVERS FOR REGULARIZED SADDLE-POINT SYSTEMS

DANIELA DI SERAFINO* AND DOMINIQUE ORBAN†

Abstract. We consider the iterative solution of regularized saddle-point systems. When the leading block is symmetric and positive semi-definite on an appropriate subspace, Dollar, Gould, Schilders, and Wathen (2006) describe how to apply the conjugate gradient (CG) method coupled with a constraint preconditioner, a choice that has proved to be effective in optimization applications. We investigate the design of constraint-preconditioned variants of other Krylov methods for regularized systems by focusing on the underlying basis-generation process. We build upon principles laid out by Gould, Orban, and Rees (2014) to provide general guidelines that allow us to specialize any Krylov method to regularized saddle-point systems. In particular, we obtain constraint-preconditioned variants of Lanczos and Arnoldi-based methods, including the Lanczos version of CG, MINRES, SYMMLQ, GMRES(ℓ) and DQGMRES. We also provide MATLAB implementations in hopes that they are useful as a basis for the development of more sophisticated software. Finally, we illustrate the numerical behavior of constraint-preconditioned Krylov solvers using symmetric and nonsymmetric systems arising from constrained optimization.

Key words. Regularized saddle-point systems, constraint preconditioners, Lanczos and Arnoldi procedures, Krylov solvers.

AMS subject classifications. 65F08, 65F10, 65F50, 90C20.

1. Introduction.

We consider the iterative solution of the regularized saddle-point system

$$\begin{bmatrix} A & B^T \\ B & -C \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}, \tag{1}$$

where $A \in \mathbb{R}^{n\times n}$ may be nonsymmetric, $C \in \mathbb{R}^{m\times m}$ is nonzero and symmetric, and $B \in \mathbb{R}^{m\times n}$. We denote K the matrix of (1). There is no loss of generality in assuming that the last m entries of the right-hand side of (1) are zero, as discussed later.

A constraint preconditioner for (1) has the form

$$P = \begin{bmatrix} G & B^T \\ B & -C \end{bmatrix}, \tag{2}$$

where G is an approximation to A such that (2) is nonsingular. When A is symmetric and has appropriate additional properties, a constraint preconditioner allows the application of CG even though K and P are indefinite (Dollar et al., 2006).

We are interested in the design of constraint-preconditioned versions of additional Krylov methods for (1), including methods that can be used when A is nonsymmetric. We extend the work of Gould et al. (2014) on projected and constraint-preconditioned Krylov methods for saddle-point systems with $C = 0$ by exploiting a suitable reformulation of (1) suggested by Dollar et al. (2006). We develop constraint-preconditioned

*Department of Mathematics and Applications "R. Caccioppoli", University of Naples Federico II, Naples, Italy. E-mail: daniela.diserafino@unina.it. Research partially supported by GERAD during a visit of this author in 2017, and by Gruppo Nazionale per il Calcolo Scientifico – Istituto Nazionale di Alta Matematica (GNCS–INdAM), Italy.

†GERAD and Department of Mathematics and Industrial Engineering, École Polytechnique, Montréal, QC, Canada. E-mail: dominique.orban@gerad.ca. Research partially supported by an NSERC Discovery Grant.

variants of the Lanczos and Arnoldi basis-generation processes, and use them to derive variants of Krylov solvers based on those processes. More generally, we provide guidelines that can be also exploited to obtain constraint-preconditioned versions of other Krylov methods not considered in this paper. Finally, we distribute MATLAB implementations of the constraint-preconditioned methods discussed here as templates for the development of more sophisticated numerical software.

Systems of type (1) arise in interior-point methods for constrained optimization in the presence of inequality constraints or when regularization is used (Benzi, Golub, and Liesen, 2005; D'Apuzzo, De Simone, and di Serafino, 2010; Friedlander and Orban, 2012). They also appear in Lagrangian approaches for variational problems with equality constraints when the constraints are relaxed or a penalty term is applied (Pestana and Wathen, 2015). In the above cases, A is usually symmetric, but may also be nonsymmetric—see Section 7, and often has additional properties, e.g., it accounts for local convexity of the optimization problem. Regularized saddle-point systems with nonsymmetric A arise also from the stabilized finite-element discretization of Oseen problems obtained by linearization, through Picard's method, of the steady-state Navier-Stokes equations governing the flow of a Newtonian incompressible viscous fluid (Benzi et al., 2005).

Constraint preconditioners have widely demonstrated their effectiveness on saddle-point systems, especially when the leading block is symmetric and enjoys additional properties, such as being positive definite; much work has been carried out to develop, analyze and approximate constraint preconditioners in this case, see, e.g., (Benzi et al., 2005; D'Apuzzo et al., 2010; Gould et al., 2014; De Simone, di Serafino, and Morini, 2018) and the references therein.

The rest of this paper is organized as follows. Section 2 provides preliminary results used in the sequel. In Section 3, we describe the constraint-preconditioned Lanczos process and, in Section 4, we present variants of Krylov solvers based on it. In Section 5, we describe the constraint-preconditioned Arnoldi process and associated Krylov methods. In Section 6, we discuss implementation issues and provide details on the MATLAB codes. In Section 7, we illustrate the numerical behavior of some constraint-preconditioned solvers on regularized saddle-point systems, with symmetric and nonsymmetric matrices, from constrained optimization. We conclude in Section 8.

Notation. Uppercase Latin letters (A, B, ...), lowercase Latin letters (a, b, ...), and lowercase Greek letters (α, β, ...) denote matrices, vectors and scalars, respectively. The Euclidean norm is denoted $\|\cdot\|$. If $S = S^T$ is a positive definite matrix, the S-norm is defined as $\|u\|_S^2 = u^T S\, u$. All vectors are column vectors. For any vector v, $\mathrm{diag}(v)$ is the diagonal matrix with diagonal entries equal to the entries of v. For brevity, we use the MATLAB-like notation $[v\,;\,w]$ to represent the vector $[v^T\ w^T]^T$.

2. Preliminaries. We assume throughout that K is nonsingular, which implies

$$\mathrm{Null}(A) \cap \mathrm{Null}(B) = \{0\} \quad \text{and} \quad \mathrm{Null}(B^T) \cap \mathrm{Null}(C) = \{0\}. \tag{3}$$

In general the converse is not true. A counterexample consists in taking

$$A = \begin{bmatrix} 1 & -1 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Benzi et al. (2005) and D'Apuzzo et al. (2010) give additional conditions that guarantee nonsingularity of K. Note however that we do not require B to have full rank or C to be positive (semi-)definite.

In order to develop constraint-preconditioned Krylov methods for (1), we specialize the basis-generation processes underlying those methods. We focus on the Lanczos (1950) and Arnoldi (1951) processes, which compute orthonormal bases of Krylov spaces associated with symmetric and general matrices, respectively. For reference, the preconditioned Lanczos process is stated as Algorithm 4 in Appendix A. The standard Lanczos process follows by setting the preconditioner to the identity. It is straighforward to apply our arguments to the Lanczos (1950) biorthogonalization process and its transpose-free variants (Brezinski and Redivo-Zaglia, 1998; Chan, de Pillis, and van der Vorst, 1998). We implicitly assume that $A = A^T$ when considering the Lanczos process.

Following Dollar et al. (2006), we reformulate (1) as follows. Assume that $\text{rank}(C) = p$ and C has been decomposed as[1]

$$C = EFE^T, \tag{4}$$

where $F \in \mathbb{R}^{p\times p}$ is symmetric and nonsingular and $E \in \mathbb{R}^{m\times p}$. Then, by using the auxiliary variable

$$w = -FE^T y, \tag{5}$$

equation (1) may be written

$$\begin{bmatrix} A & & B^T \\ & F^{-1} & E^T \\ B & E & \end{bmatrix} \begin{bmatrix} x \\ w \\ y \end{bmatrix} = \begin{bmatrix} b \\ 0 \\ 0 \end{bmatrix}, \tag{6}$$

which has a standard symmetric saddle-point form

$$\begin{bmatrix} M & N^T \\ N & \end{bmatrix} \begin{bmatrix} g \\ y \end{bmatrix} = \begin{bmatrix} b_0 \\ 0 \end{bmatrix}, \quad M = \begin{bmatrix} A & \\ & F^{-1} \end{bmatrix}, \; N = \begin{bmatrix} B & E \end{bmatrix}, \; g = \begin{bmatrix} x \\ w \end{bmatrix}, \; b_0 = \begin{bmatrix} b \\ 0 \end{bmatrix}. \tag{7}$$

The principles laid out by Gould et al. (2014) may now be applied to (6).

Note that (6) is nonsingular if and only if (1) is nonsingular, and therefore N must have full rank. Because $g \in \text{Null}(N)$, there exists $\widehat{d} \in \mathbb{R}^{n+p-m}$ such that

$$g = \begin{bmatrix} x \\ w \end{bmatrix} = Z\,\widehat{d} = \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} \widehat{d}, \tag{8}$$

where the columns of Z form a basis of $\text{Null}(N)$. The restriction of (6) to $\text{Null}(N)$ is

$$\widehat{M}\,\widehat{x} = \widehat{b}, \tag{9}$$

where

$$\widehat{M} = Z^T M Z = Z_1^T A Z_1 + Z_2^T F^{-1} Z_2, \tag{10a}$$

$$\begin{bmatrix} x \\ w \end{bmatrix} = \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} \widehat{x}, \quad \widehat{b} = \begin{bmatrix} Z_1^T & Z_2^T \end{bmatrix} \begin{bmatrix} b \\ 0 \end{bmatrix} = Z_1^T b. \tag{10b}$$

In a Krylov method for (9), it is appropriate to use a preconditioner of the form

$$\widehat{P} = Z_1^T G Z_1 + Z_2^T F^{-1} Z_2. \tag{11}$$

[1] Note that (4) will be only used for the purpose of deriving computational processes and need not be computed in practice.

If G is suitable, the preconditioned method can be reformulated entirely in terms of full space quantities (Gould, Hribar, and Nocedal, 2001; Dollar et al., 2006; Gould et al., 2014). Following (Gould et al., 2014, Assumption 2.2), we require the following assumption.

ASSUMPTION 2.1. *The matrix*

$$\begin{bmatrix} G & \\ & F^{-1} \end{bmatrix}$$

is symmetric and positive definite on Null(N).

A consequence of Assumption 2.1 is that (11) is symmetric and positive definite.

We enforce Assumption 2.1 throughout this paper to guarantee that Krylov methods for (9) give rise to corresponding full-space methods for (1). However, at least in principle, Assumption 2.1 is not always necessary, e.g., in Krylov methods based on the Arnoldi process.

The application of the preconditioner $\widehat{P}$, i.e., $\breve{u} = \widehat{P}^{-1}\hat{u}$, can be written as

$$\begin{bmatrix} \bar{u}_x \\ \bar{u}_w \end{bmatrix} = P_G \begin{bmatrix} u_x \\ u_w \end{bmatrix}, \qquad P_G = Z\widehat{P}^{-1}Z^T, \quad \begin{bmatrix} \bar{u}_x \\ \bar{u}_w \end{bmatrix} = Z\,\breve{u}, \quad Z^T \begin{bmatrix} u_x \\ u_w \end{bmatrix} = \hat{u}. \tag{12}$$

Furthermore,

$$P_G \begin{bmatrix} G & \\ & F^{-1} \end{bmatrix}$$

is an oblique projector into Null(N). Let $\widehat{L}$ be the lower triangular Cholesky factor of $\widehat{P}$ and let

$$\widehat{\mathcal{K}} = \mathcal{K}\left(\widehat{L}^{-1}\widehat{M}\,\widehat{L}^{-T},\ \widehat{L}^{-1}(\hat{b} - \widehat{M}\,\hat{x}_0)\right) \tag{13}$$

be the Krylov space generated by the preconditioned reduced operator $\widehat{L}^{-1}\widehat{M}\,\widehat{L}^{-T}$ and initial vector $\widehat{L}^{-1}(\hat{b} - \widehat{M}\,\hat{x}_0)$, where $\hat{b}$ is given in (10) and $x_0 = Z_1\hat{x}_0$, with Z_1 defined in (8).

The computation of (12) can be obtained by solving

$$\begin{bmatrix} G & & B^T \\ & F^{-1} & E^T \\ B & E & \end{bmatrix} \begin{bmatrix} \bar{u}_x \\ \bar{u}_w \\ \bar{z} \end{bmatrix} = \begin{bmatrix} u_x \\ u_w \\ 0 \end{bmatrix}, \tag{14}$$

see, e.g., Gould et al. (2001), so that P_G could be expressed as

$$P_G = \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \end{bmatrix} \begin{bmatrix} G & & B^T \\ & F^{-1} & E^T \\ B & E & \end{bmatrix}^{-1} \begin{bmatrix} I & 0 \\ 0 & I \\ 0 & 0 \end{bmatrix}.$$

We now apply Principles 2.1 and 2.2 of Gould et al. (2014) to the standard Lanczos basis-generation process for $\widehat{\mathcal{K}}$, and obtain the projected Lanczos process outlined in Algorithm 1.

In Algorithm 1, the notation $\|u\|_{[P]}$ represents a measure of the deviation of $u = [u_x\,;\,u_w]$ from Null(N) (Gould et al., 2014, Section 3). More precisely

$$\|u\|^2_{[P]} := u_x^T\bar{u}_x + u_w^T\bar{u}_w, \tag{15}$$

Algorithm 1 Projected Lanczos Process

choose $[x_0\,;\,w_0]$ such that $Bx_0 + Ew_0 = 0$ — *initial guess*
$v_{0,x} = 0,\;\; v_{0,w} = -w_0$ — *initial Lanczos vector*
$u_{0,x} = b - Ax_0,\;\; u_{0,w} = -F^{-1}w_0$ — $u_0 = b_0 - Mg_0$
$[\bar{u}_{1,x}\,;\,\bar{u}_{1,w}\,;\,\bar{z}_1] \leftarrow$ solution of (14) with right-hand side $[u_{0,x}\,;\,u_{0,w}\,;\,0]$
$v_{1,x} = \bar{u}_{1,x},\;\; v_{1,w} = \bar{u}_{1,w}$ — $v_1 = P_G\,u_0$
$\beta_1 = (v_{1,x}^T u_{0,x} + v_{1,w}^T u_{0,w})^{\frac{1}{2}}$ — $\beta_1 = (v_1^T u_0)^{\frac{1}{2}}$
if $\beta_1 \neq 0$ **then**
    $v_{1,x} = v_{1,x}/\beta_1,\;\; v_{1,w} = v_{1,w}/\beta_1$ — $\|v_1\|_{[P]} = 1$
end if
$k = 1$
while $\beta_k \neq 0$ **do**
    $u_{k,x} = Av_{k,x},\;\; u_{k,w} = F^{-1}v_{k,w}$ — $u_k = Mv_k$
    $\alpha_k = v_{k,x}^T u_{k,x} + v_{k,w}^T u_{k,w}$ — $\alpha_k = v_k^T u_k$
    $[\bar{u}_{k+1,x}\,;\,\bar{u}_{k+1,w}\,;\,\bar{z}_{k+1}] \leftarrow$ solution of (14) with right-hand side $[u_{k,x}\,;\,u_{k,w}\,;\,0]$
    $v_{k+1,x} = \bar{u}_{k+1,x} - \alpha_k v_{k,x} - \beta_k v_{k-1,x}$ — $v_{k+1} = \bar{u}_{k+1} - \alpha_k v_k - \beta_k v_{k-1}$
    $v_{k+1,w} = \bar{u}_{k+1,w} - \alpha_k v_{k,w} - \beta_k v_{k-1,w}$
    $\beta_{k+1} = (v_{k+1,x}^T u_{k,x} + v_{k+1,w}^T u_{k,w})^{\frac{1}{2}}$ — $\beta_{k+1} = (v_{k+1}^T u_k)^{\frac{1}{2}}$
    if $\beta_{k+1} \neq 0$ **then**
        $v_{k+1,x} = v_{k+1,x}/\beta_{k+1},\;\; v_{k+1,w} = v_{k+1,w}/\beta_{k+1}$ — $\|v_{k+1}\|_{[P]} = 1$
    end if
    $k = k + 1$
end while

where $\bar{u} = [\bar{u}_x\,;\,\bar{u}_w]$ is defined by (14). Note that $\|u\|_{[P]}$ is actually a seminorm and vanishes if and only if $[u_x\,;\,u_w]$ is orthogonal to Null(N).

Conceptually, the Lanczos process corresponding to Algorithm 1 can be summarized as

$$\begin{bmatrix} A & & B^T \\ & F^{-1} & E^T \\ B & E & \end{bmatrix} \begin{bmatrix} V_{k,x} \\ V_{k,w} \\ \bar{Z}_k \end{bmatrix} = \begin{bmatrix} G & & B^T \\ & F^{-1} & E^T \\ B & E & \end{bmatrix} \left(\begin{bmatrix} V_{k,x} \\ V_{k,w} \\ \bar{Z}_k \end{bmatrix} T_k + \beta_{k+1} \begin{bmatrix} v_{k+1,x} \\ v_{k+1,w} \\ \bar{z}_{k+1} \end{bmatrix} e_k^T \right),$$

where

$$V_{k,x} = \begin{bmatrix} v_{1,x} & \ldots & v_{k,x} \end{bmatrix}, \quad V_{k,w} = \begin{bmatrix} v_{1,w} & \ldots & v_{k,w} \end{bmatrix}, \quad \bar{Z}_k = \begin{bmatrix} \bar{z}_1 & \ldots & \bar{z}_k \end{bmatrix},$$

and T_k is the usual Lanczos tridiagonal matrix. Provided that $[x_0\,;\,w_0] \in$ Null(N), (Gould et al., 2014, Theorem 2.2) guarantees that Algorithm 1 is well defined and equivalent to Algorithm 4 in Appendix A applied to (9)–(10) with preconditioner (11). In Algorithm 1 and subsequent algorithms, we use the symbol "$\leftarrow$" to assign to the vector on the left of the arrow the result of the external procedure on the right of the arrow.

In the next sections we show how the projected basis-generation procedures can be further reformulated by referring to the original system (1), thus avoiding the use of E and F and the factorization (4).

3. Constraint-Preconditioned Lanczos Process. If we define $\bar{p}_k = \bar{u}_{k,x}$ for all $k \geqslant 1$, and

$$t_k = EFu_{k,w}, \quad k = 0, 1, \ldots \tag{16}$$

then (14) at line 14 of Algorithm 1 can be written as

$$\begin{bmatrix} G & B^T \\ B & -C \end{bmatrix} \begin{bmatrix} \bar{p}_{k+1} \\ \bar{z}_{k+1} \end{bmatrix} = \begin{bmatrix} u_{k,x} \\ -t_k \end{bmatrix}. \tag{17}$$

Assumption 2.1 occurs when the sum of the number of negative eigenvalues of the matrix of (17) and C is m (Dollar et al., 2006, Theorem 2.1), which may be verified if an inertia-revealing symmetric indefinite factorization is used to solve (17), such as that of Duff (2004).

Unfortunately, (17) still appears to depend on F via (16). We now reformulate Algorithm 1 in terms of full-space quantities. Define the initial guess

$$w_0 = -FE^T q_0, \tag{18}$$

where $q_0 \in \mathbb{R}^m$ is arbitrary (e.g., $q_0 = 0$). Line 3 of Algorithm 1 and (16) yield

$$u_{0,x} = r_0 = b - Ax_0, \quad u_{0,w} = E^T q_0, \quad t_0 = Cq_0.$$

From here on, let us denote $p_k = v_{k,x}$. At lines 4-5 of Algorithm 1, we compute $p_1 = \bar{p}_1$ and $\bar{z}_1$ from (14), which yields, in particular, $\bar{u}_{1,w} = FE^T(q_0 - \bar{z}_1)$. If we define

$$s_1 = q_0 - \bar{z}_1, \quad q_1 = s_1,$$

lines 5-6 of Algorithm 1 take the form

$$p_1 = \bar{p}_1 \tag{19a}$$

$$v_{1,w} = FE^T q_1, \tag{19b}$$

$$\beta_1 = (p_1^T u_{0,x} + q_1^T C q_1)^{\frac{1}{2}} = (p_1^T u_{0,x} + q_1^T t_0)^{\frac{1}{2}}. \tag{19c}$$

We then normalize by dividing p_1 and q_1 by β_1. Lines 12–13 of Algorithm 1 and (16) give

$$u_{1,w} = E^T q_1, \tag{20a}$$

$$t_1 = Cq_1, \tag{20b}$$

$$\alpha_1 = p_1^T u_{1,x} + q_1^T t_1 = p_1^T A p_1 + q_1^T C q_1. \tag{20c}$$

We now compute p_2 and $v_{2,w}$ from lines 15–16 of Algorithm 1 with $k = 1$, i.e., we compute $\bar{p}_2$ and $\bar{z}_2$ from (17), and note that $\bar{u}_{2,w} = FE^T(q_1 - \bar{z}_2)$. Thus, by setting

$$s_2 = q_1 - \bar{z}_2, \quad q_2 = s_2 - \alpha_1 q_1 - \beta_1 q_0,$$

we obtain from lines 2 and 15–16 of Algorithm 1 together with (19b), (20a) and (20b):

$$\begin{aligned} p_2 &= \bar{p}_2 - \alpha_1 p_1 - \beta_1 p_0 \\ v_{2,w} &= FE^T s_2 - \alpha_1 FE^T q_1 - \beta_1 FE^T q_0 = FE^T q_2 \\ \beta_2 &= (p_2^T A p_1 + q_2^T C q_1)^{\frac{1}{2}} = (p_2^T u_{1,x} + q_2^T t_1)^{\frac{1}{2}}. \end{aligned}$$

Then, according to line 19, p_2 must be divided by β_2, and we do the same with q_2. An induction argument shows that for all $k \geqslant 1$

$$\begin{aligned} u_{k,w} &= E^T q_k, \\ t_k &= C q_k, \\ \alpha_k &= p_k^T u_{k,x} + q_k^T t_k = p_k^T A p_k + q_k^T C q_k, \end{aligned}$$

where q_k has been normalized by β_k. Furthermore, letting

$$s_{k+1} = q_k - \bar{z}_{k+1}, \quad q_{k+1} = s_{k+1} - \alpha_k q_k - \beta_k q_{k-1},$$

we obtain

$$\begin{aligned} p_{k+1} &= \bar{p}_{k+1} - \alpha_k p_k - \beta_k p_{k-1} \\ v_{k+1,w} &= F E^T q_{k+1}, \\ \beta_{k+1} &= (p_{k+1}^T A p_k + q_{k+1}^T C q_k)^{\frac{1}{2}} = (p_{k+1}^T u_{k,x} + q_{k+1}^T t_k)^{\frac{1}{2}}. \end{aligned}$$

We divide p_{k+1} and q_{k+1} by β_{k+1} to obtain the vectors to be used at the next iteration. Thus, if we rename $u_{k,x}$ as u_k, we obtain Algorithm 2.

Algorithm 2 Constraint-Preconditioned Lanczos Process

1: choose $[x_0\,;\,q_0]$ such that $Bx_0 - Cq_0 = 0$ *initial guess*
2: $p_0 = 0$ *initial Lanczos vector*
3: $u_0 = b - Ax_0, \;\; t_0 = Cq_0$
4: $[\bar{p}_1\,;\,\bar{z}_1] \leftarrow$ solution of (17) with right-hand side $[u_0\,;\,-t_0]$
5: $p_1 = \bar{p}_1$
6: $s_1 = q_0 - \bar{z}_1, \;\; q_1 = s_1$
7: $\beta_1 = (p_1^T u_0 + q_1^T t_0)^{\frac{1}{2}}$
8: **if** $\beta_1 \neq 0$ **then**
9: $\quad p_1 = p_1/\beta_1, \;\; q_1 = q_1/\beta_1$
10: **end if**
11: $k = 1$
12: **while** $\beta_k \neq 0$ **do**
13: $\quad u_k = Ap_k, \;\; t_k = Cq_k$
14: $\quad \alpha_k = p_k^T u_k + q_k^T t_k$ $= p_k^T A p_k + q_k^T C q_k$
15: $\quad [\bar{p}_{k+1}\,;\,\bar{z}_{k+1}] \leftarrow$ solution of (17) with right-hand side $[u_k\,;\,-t_k]$
16: $\quad p_{k+1} = \bar{p}_{k+1} - \alpha_k p_k - \beta_k p_{k-1}$
17: $\quad s_{k+1} = q_k - \bar{z}_{k+1}, \;\; q_{k+1} = s_{k+1} - \alpha_k q_k - \beta_k q_{k-1}$
18: $\quad \beta_{k+1} = (p_{k+1}^T u_k + q_{k+1}^T t_k)^{\frac{1}{2}}$ $= (p_{k+1}^T A p_k + q_{k+1}^T C q_k)^{\frac{1}{2}}$
19: $\quad$ **if** $\beta_{k+1} \neq 0$ **then**
20: $\quad\quad p_{k+1} = p_{k+1}/\beta_{k+1}, \;\; q_{k+1} = q_{k+1}/\beta_{k+1}$
21: $\quad$ **end if**
22: $\quad k = k + 1$
23: **end while**

The above transformations can be condensed in the following principle, which summarizes the conversion a of projected process into a constraint-preconditioned process.

PRINCIPLE 1.

1. *Basis vectors* $v_{k+1,x}$ *are unchanged;*
2. *Basis vectors* $v_{k+1,w}$ *have the form* $FE^T q_{k+1}$*, where* q_{k+1} *is defined by*

$$\begin{aligned} s_{k+1} &= q_k - \bar{z}_{k+1}, \\ q_1 &= s_1, \\ q_{k+1} &= s_{k+1} - \alpha_k q_k - \beta_k q_{k-1}, \quad (k \geqslant 1), \end{aligned}$$

   and where $\bar{z}_{k+1}$ *results from the solution of* (17)*;*
3. *Inner products of the form* $v_{i,w}^T u_{j,w}$ *become* $q_i^T C q_j = q_i^T t_j$.

Theorem 1 summarizes the equivalence between the two formulations.

THEOREM 1. *Let* E *and* F *be as defined in* (4) *and* G *chosen to satisfy Assumption 2.1. Let* $q_0 \in \mathbb{R}^m$ *be arbitrary. Then, Algorithm 1 with starting guesses* $x_0 \in \mathbb{R}^n$ *and* $w_0 = -FE^T q_0$*, such that* $Bx_0 + Ew_0$*, is equivalent to Algorithm 2 with starting guesses* x_0 *and* q_0*. In particular, for all* k*, the vectors* $v_{k,x}$ *and* $v_{k,w}$*, and the scalars* α_k *and* β_k *in Algorithm 1 are equal to the vectors* p_k *and* $FE^T q_k$*, and to the scalars* α_k *and* β_k *in Algorithm 2, respectively.*

Note that Algorithm 2 does not contain references to E and F. The variable s_k is used only to improve readability. Assumption 2.1 guarantees that Algorithm 2 is well posed because it is equivalent to Algorithm 1, which, in turn, is equivalent to the standard Lanczos process for building an orthonormal basis of (13). The main advantages of Algorithm 2 are that it works directly with the formulation (1) and it only requires storage for three vectors of size $n + m$ ($[p_k\,;\,q_k]$, $[u_k\,;\,t_k]$, and $[\bar{p}_k\,;\,\bar{z}_k]$), as opposed to the same number of vectors of size $n + p + m$ for Algorithm 1.

We call Algorithm 2 the Constraint-Preconditioned Lanczos (CP-Lanczos) process because of its similarity to a Lanczos process for building an orthonormal basis of a Krylov space associated with the preconditioned operator $P^{-1}M$, even though the latter appears nonsymmetric.

4. Constraint-Preconditioned Lanczos-Based Krylov Solvers. We may exploit Theorem 1 and use Algorithm 2 to derive a constraint-preconditioned version of any Krylov method based on the Lanczos process. To this aim, we must understand how the update of the k-th iterate $[x_k\,;\,w_k]$ in a Krylov method based on Algorithm 1 translates into the update of the k-th iterate $[x_k\,;\,y_k]$ in the version of that Krylov method based on Algorithm 2. In the following, the former and the latter version of the Krylov method are referred to as projected-Krylov (P-Krylov) and constraint-preconditioned-Krylov (CP-Krylov), respectively.

Because the initial guess $g_0 = [x_0\,;\,w_0]$ of P-Krylov applied to (6) must lie in $\text{Null}(N)$, CP-Krylov must be initialized with $[x_0\,;\,y_0]$ such that

$$Bx_0 - Cy_0 = 0. \tag{21}$$

Our first result states a property of Algorithm 2 that follows from a specific q_0.

LEMMA 1. *Let Algorithm 2 be initialized with* $x_0 \in \mathbb{R}^n$ *and* $q_0 \in \text{Null}(C)$*. Then, for all* $k \geqslant 0$*,*

$$Bp_k + Cq_k = 0. \tag{22}$$

Proof. We proceed by induction. For $k = 0$, (22) holds because $p_0 = 0$ and $q_0 \in \text{Null}(C)$. For $k = 1$, $p_1 = \bar{p}_1$, $q_1 = q_0 - \bar{z}_1 = -\bar{z}_1$, and (17) and our assumption

that $q_0 \in \text{Null}(C)$ yield

$$Bp_1 + Cq_1 = B\bar{p}_1 - C\bar{z}_1 = -t_0 = -Cq_0 = 0.$$

Assume (22) holds for any index $j \leqslant k$. Lines 16–17 of Algorithm 2, (16), (17), and our induction assumption imply that

$$\begin{aligned} Bp_{k+1} + Cq_{k+1} &= B\bar{p}_{k+1} + C(q_k - \bar{z}_{k+1}) - \alpha_k(Bp_k + Cq_k) - \beta_k(Bp_{k-1} + Cq_{k-1}) \\ &= B\bar{p}_{k+1} + C(q_k - \bar{z}_{k+1}) \\ &= B\bar{p}_{k+1} - C\bar{z}_{k+1} + t_k = 0, \end{aligned}$$

which establishes (22). ∎

An interesting property of the CP-Lanczos process is that it is equivalent to formally applying the standard Lanczos process to system (1) with preconditioner (2), where by "formal application", we mean that the Lanczos process is applied blindly as if P were positive definite. Such formal application is stated as Algorithm 5 in Appendix A. The equivalence with Algorithm 2 is stated in the next result, which parallels (Gould et al., 2014, Theorem 2.2).

THEOREM 2. *Let Algorithm 2 be initialized with $x_0 \in \mathbb{R}^n$ such that $Bx_0 = 0$, $q_0 = 0 \in \mathbb{R}^m$, and Algorithm 5 be initialized with the same x_0 and $y_0 \in \mathbb{R}^m$ such that (21) is satisfied. Then, for all $k \geqslant 0$, $v_{k,x} = p_k$ and $v_{k,y} = -q_k$, where $[v_{k,x}\,;\, v_{k,y}]$ is the k-th Lanczos vector generated in Algorithm 5, and p_k and q_k are the k-th Lanczos vectors generated in Algorithm 2. In addition, the scalars α_k and β_k computed at each iteration are the same in both algorithms.*

Proof. We proceed by induction. The result holds for $k = 0$ because $[v_{0,x}\,;\, v_{0,y}] = [0\,;\, 0] = [p_0\,;\, -q_0]$. With $q_0 = 0$, Algorithm 2 initializes $u_0 = b - Ax_0$ and $t_0 = 0$. Because (21) is satisfied, Algorithm 5 initializes $r_{0,x} = u_0 - B^T y_0$ and $r_{0,y} = 0$. Thus, $[v_{1,x}\,;\, v_{1,y}]$ solves (17) with right-hand side $[u_0 - B^T y_0\,;\, 0]$. By (Gould et al., 2014, Theorem 2.1, item 2), $[v_{1,x}\,;\, v_{1,y}]$ equivalently solves (17) with right-hand side $[u_0\,;\, 0]$, and therefore, $[v_{1,x}\,;\, v_{1,y}]$ at line 4 of Algorithm 5 is equal to $[\bar{p}_1\,;\, \bar{z}_1]$. Lines 5–6 of Algorithm 2 subsequently set $p_1 = \bar{p}_1 = v_{1,x}$ and $q_1 = s_1 = q_0 - \bar{z}_1 = -v_{1,y}$.

With $q_0 = 0$, line 7 of Algorithm 2 computes $\beta_1 = (p_1^T u_0)^{\frac{1}{2}}$. We take the inner product of the second row of (17) with $\bar{z}_1 = -q_1$ and note that $t_0 = 0$, and obtain $\bar{z}_1^T B p_1 = \bar{z}_1^T C \bar{z}_1 = q_1^T C q_1$. Similarly, we take the inner product of the first row of (17) with p_1 and substitute $\bar{z}_1^T B p_1$ to obtain $p_1^T u_0 = p_1^T G p_1 + q_1^T C q_1$, so that β_1 is the same as that computed at line 5 of Algorithm 5. We have established that the result also holds for $k = 1$.

At a general iteration k, Algorithm 2 sets $u_k = Ap_k$, $t_k = Cq_k$ and computes $\alpha_k = p_k^T u_k + q_k^T t_k = p_k^T A p_k + q_k^T C q_k$. By Lemma 1, $q_k^T B p_k + q_k^T C q_k = 0$, so that $\alpha_k = p_k^T A p_k - 2q_k^T B p_k - q_k^T C q_k$. Under the recurrence assumption that $v_{k,x} = p_k$ and $v_{k,y} = -q_k$, this expression of α_k is the same as that computed at line 12 of Algorithm 5.

At line 15 of Algorithm 2, we compute $[\bar{p}_{k+1}\,;\, \bar{z}_{k+1}]$ from (17), or, equivalently, as the solution to

$$\begin{bmatrix} G & B^T \\ B & -C \end{bmatrix} \begin{bmatrix} \bar{p}_{k+1} \\ \bar{z}_{k+1} - q_k \end{bmatrix} = \begin{bmatrix} Ap_k \\ 0 \end{bmatrix}.$$

In view of Lemma 1, our recurrence assumption, and (Gould et al., 2014, Theorem 2.1, item 2), line 13 of Algorithm 5 computes $[v_{k+1,x}\,;\, v_{k+1,y}]$ as the solution to the

same system as above. Therefore, at that point in each algorithm $v_{k+1,x} = \bar{p}_{k+1}$ and $v_{k+1,y} = \bar{z}_{k+1} - q_k = -s_{k+1}$. The vector updates at lines 16–17 of Algorithm 2 together with those at line 14 of Algorithm 5 show that $v_{k+1,x} = p_{k+1}$ and $v_{k+1,y} = -q_{k+1}$. Our recurrence assumption and Lemma 1 yield $Bv_{k,x} - Cv_{k,y} = 0$ and $Bv_{k+1,x} - Cv_{k+1,y} = 0$. Finally, Algorithm 5 sets

$$\begin{aligned}\beta_{k+1}^2 &= v_{k+1,x}^T u_{k,x} + v_{k+1,y}^T u_{k,y} \\ &= v_{k+1,x}^T A v_{k,x} + (Bv_{k+1,x} - Cv_{k+1,y})^T v_{k,y} + v_{k+1,y}^T B v_{k,x} \\ &= v_{k+1,x}^T A v_{k,x} + v_{k+1,y}^T C v_{k,x},\end{aligned}$$

which is the same value computed in Algorithm 2. ∎

Theorem 2 shows that Algorithm 2 may be summarized as

$$\begin{bmatrix} A & B^T \\ B & -C \end{bmatrix} \begin{bmatrix} P_k \\ -Q_k \end{bmatrix} = \begin{bmatrix} G & B^T \\ B & -C \end{bmatrix} \left(\begin{bmatrix} P_k \\ -Q_k \end{bmatrix} T_k + \beta_{k+1} \begin{bmatrix} p_{k+1} \\ -q_{k+1} \end{bmatrix} e_k^T, \right)$$

provided that $Bx_0 = 0$ and $q_0 = 0$, where T_k is the same as in Algorithm 1, and

$$P_k = \begin{bmatrix} p_1 & \cdots & p_k \end{bmatrix}, \quad Q_k = \begin{bmatrix} q_1 & \cdots & q_k \end{bmatrix}.$$

A consequence of Theorem 2 is that any CP-Krylov method is formally equivalent to the corresponding standard Krylov method applied to system (1) with preconditioner (2).

COROLLARY 1. *Let Algorithm 2 be initialized with $x_0 \in \mathbb{R}^n$ such that $Bx_0 = 0$, $q_0 = 0 \in \mathbb{R}^m$, and Algorithm 5 be initialized with the same x_0 and $y_0 \in \mathbb{R}^m$ such that (21) is satisfied. The k-th approximate solution of (1) computed by any Lanczos-based CP-Krylov method coincides with the k-th approximate solution obtained by formally applying the standard version of the same method to (1) with preconditioner (2).*

Although Corollary 1 states that standard Lanczos-based methods can be safely applied to (1) with preconditioner (2) and an appropriate starting point, Algorithm 2 reduces the computational effort by never requiring products with B or B^T. Only products with A and C are necessary. On the other hand, thanks to Theorem 2, specialized implementations of the standard Lanczos-based methods can be developed by exploiting the equalities $Bp_k + Cq_k = 0$ and $Bx_k - Cy_k = 0$, thus saving matrix-vector products. The computation involving s_{k+1} can be carried out, for example, as the update $q_{k-1} = q_k - \bar{z}_{k+1} - \beta_k q_{k-1}$ followed by $q_{k+1} = q_{k-1} - \alpha_k q_k$, or s_{k+1} can overwrite $\bar{z}_{k+1}$. Finally, once (2) has been factorized, storing B is no longer necessary, and this can be used to free memory if needed.

A consequence of Theorem 2 is a formal equivalence between the iterates generated by Lanczos-based methods applied by way of Algorithm 2 and Algorithm 5. This equivalence requires a re-interpretation of the optimality conditions associated with the Krylov method.

Consider, e.g., MINRES (Paige and Saunders, 1975). The residual associated with iterate $[x_k\,;\, w_k\,;\, y_k]$ generated by P-MINRES, with $w_k = -FE^T y_k$, is

$$r_{\mathrm{P},k} = \begin{bmatrix} r_{\mathrm{P},k,x} \\ r_{\mathrm{P},k,w} \\ r_{\mathrm{P},k,y} \end{bmatrix} = \begin{bmatrix} b \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} A & & B^T \\ & F^{-1} & E^T \\ B & E & \end{bmatrix} \begin{bmatrix} x_k \\ w_k \\ y_k \end{bmatrix} = \begin{bmatrix} b - Ax_k - B^T y_k \\ 0 \\ 0 \end{bmatrix},$$

where we used the fact that $Bx_k + Ew_k = 0$ for all k. This residual corresponds to the residual at iterate $[x_k\,;\,y_k]$ generated by CP-MINRES:

$$r_{\text{CP},k} = \begin{bmatrix} b \\ 0 \end{bmatrix} - \begin{bmatrix} A & B^T \\ B & -C \end{bmatrix} \begin{bmatrix} x_k \\ y_k \end{bmatrix} = \begin{bmatrix} b - Ax_k - B^T y_k \\ 0 \end{bmatrix},$$

where we exploited the fact that $Bx_k - Cy_k = 0$ for all k, which comes from $Bx_k + Ew_k = 0$ and $w_k = -FE^T y_k$.

We may apply the arguments of Gould et al. (2014, Section 3) to conclude that P-MINRES, and hence CP-MINRES, minimizes the deviation of $[r_{\text{P},k,x}\,;\,0]$ from the range space of N, i.e., as in (15),

$$\|r_{\text{P},k}\|^2_{[P]} = (b - Ax_k - B^T y_k)^T h_k, \tag{23}$$

where

$$\begin{bmatrix} G & & B^T \\ & F^{-1} & E^T \\ B & E & \end{bmatrix} \begin{bmatrix} h_k \\ f_k \\ l_k \end{bmatrix} = \begin{bmatrix} b - Ax_k - B^T y_k \\ 0 \\ 0 \end{bmatrix}.$$

Because $h_k \in \text{Null}(B)$, we also have

$$\|r_{\text{P},k}\|^2_{[P]} = (b - Ax_k)^T h_k. \tag{24}$$

Equivalently, h_k may be computed from

$$\begin{bmatrix} G & B^T \\ B & -C \end{bmatrix} \begin{bmatrix} h_k \\ l_k \end{bmatrix} = \begin{bmatrix} b - Ax_k \\ 0 \end{bmatrix}.$$

Because of its residual norm minimization property, CP-MINRES is appropriate to solve saddle-point systems in a linesearch inexact-Newton context, where we seek to reduce the residual of the Newton-like equations (1) in an appropriate space.

The same reasoning applies to the constraint-preconditioned version of any Lanczos-based Krylov method. For example, Paige and Saunders (1975) derive the conjugate gradient method of Hestenes and Stiefel (1952) directly from the Lanczos process. The nullspace variant of the constraint-preconditioned version, Lanczos CP-CG, generates iterates $\widehat{x}_k$ so as to minimize the energy norm of the error, i.e.,

$$\|\widehat{e}_k\|^2_{\widehat{M}} = \widehat{e}_k^T \widehat{M} \widehat{e}_k,$$

where $\widehat{e}_k = \widehat{x}_k - \widehat{x}_*$, and $\widehat{x}_*$ is the exact solution of (9). The definitions (10) yield

$$\begin{aligned} \|\widehat{e}_k\|^2_{\widehat{M}} &= (\widehat{x}_k - \widehat{x}_*)^T Z^T M Z (\widehat{x}_k - \widehat{x}_*) \\ &= (x_k - x_*)^T A (x_k - x_*) + (w_k - w_*)^T F^{-1} (w_k - w^*) \\ &= (x_k - x_*)^T A (x_k - x_*) + (y_k - y_*)^T C (y_k - y_*), \end{aligned}$$

where we used again the relationship $w_k = -FE^T y_k$ between iterates of P-CG and CP-CG. For Lanczos CP-CG to be applicable, $\widehat{M}$ must be positive definite, which occurs when the sum of the number of negative eigenvalues of K and C is m (Dollar et al., 2006, Theorem 2.1).

We can derive a "traditional" CP-CG implementation by applying the usual transformations to the Lanczos CP-CG. The result coincides with the implementation

of Dollar et al. (2006), although the latter authors assume that B has full row rank for specific purposes. It is also equivalent to that of Cafieri, D'Apuzzo, De Simone, and di Serafino (2007a) for (1) with positive definite C. The above suggests that CP-CG is appropriate to solve saddle-point systems in constrained optimization where (1) is used to minimize a quadratic model of a penalty function and sufficient decrease of this quadratic model is sought, such as in trust-region methods.

Our last example considers SYMMLQ (Paige and Saunders, 1975), which does not require $\widehat{M}$ to be positive definite but, like CG, requires (1) to be consistent. Its constraint-preconditioned version, CP-SYMMLQ, computes $[x_k\,;\,y_k]$ so as to minimize the error in a norm defined by the preconditioner, i.e.,

$$\begin{aligned}\widehat{e}_k^T \widehat{P}^{-1} \widehat{e}_k &= (\widehat{x}_k - \widehat{x}_*)^T (Z^T P Z)^{-1} (\widehat{x}_k - \widehat{x}_*) \\ &= (\widehat{x}_k - \widehat{x}_*)^T Z^T Z (Z^T P Z)^{-1} Z^T Z (\widehat{x}_k - \widehat{x}_*) \\ &= \begin{bmatrix} x_k - x_* \\ w_k - w_* \end{bmatrix}^T P_G \begin{bmatrix} x_k - x_* \\ w_k - w_* \end{bmatrix}.\end{aligned}$$

where we used similar identifications as above and assumed, without loss of generality, that Z has orthonormal columns. In other words, if we define

$$\begin{bmatrix} G & & B^T \\ & F^{-1} & E^T \\ B & E & \end{bmatrix} \begin{bmatrix} e_x \\ e_w \\ \bar{e} \end{bmatrix} = \begin{bmatrix} x_k - x_* \\ w_k - w_* \\ 0 \end{bmatrix}, \tag{25}$$

then

$$\widehat{e}_k^T \widehat{P}^{-1} \widehat{e}_k = (x_k - x_*)^T e_x + (w_k - w_*)^T e_w = e_x^T G e_x + e_w F^{-1} e_w.$$

By (5) and (25), there exists a vector e_y such that $e_w = -FE^T e_y$, and thus $e_w^T F^{-1} e_w = e_y^T C e_y$. The second block row of (25) premultiplied by E yields $EF(w_k - w_*) - C\bar{e} = -Ce_y$, so that (25) can be written as

$$\begin{bmatrix} G & B^T \\ B & -C \end{bmatrix} \begin{bmatrix} e_x \\ e_y \end{bmatrix} = \begin{bmatrix} x_k - x_* \\ 0 \end{bmatrix}.$$

Finally, CP-SYMMLQ minimizes

$$\widehat{e}_k^T \widehat{P}^{-1} \widehat{e}_k = e_x^T G e_x + e_y^T C e_y.$$

5. Constraint-Preconditioned Arnoldi Process and Associated Krylov Solvers. A constraint-preconditioned version of the Arnoldi process can be derived by reasoning as in Section 3, obtaining Algorithm 3. The equivalence between the projected version (Algorithm 6 in Appendix A) and the constraint-preconditioned version is stated in Theorem 3, which is akin to Theorem 1.

THEOREM 3. *Let E and F be as defined in (4) and G chosen to satisfy Assumption 2.1. Let $q_0 \in \mathbb{R}^m$ be arbitrary. Then, Algorithm 6 in Appendix A with starting guesses $x_0 \in \mathbb{R}^m$ and $w_0 = -FE^T q_0$, such that $Bx_0 + Ew_0 = 0$, is equivalent to Algorithm 3 with starting guesses x_0 and q_0. In particular, for all k, the vectors $v_{k,x}$ and $v_{k,w}$ and the scalars $h_{i,k}$ in Algorithm 6 are equal to the vectors p_k and $FE^T q_k$, and to the scalars $h_{i,k}$ in Algorithm 3, respectively.*

As in the case of the Lanczos process, the CP-Arnoldi process is equivalent to applying the corresponding standard Arnoldi process to system (1) with preconditioner (2) (see Algorithm 7 in Appendix A), as stated in the next theorem.

Algorithm 3 Constraint-Preconditioned Arnoldi Process

choose $[x_0\,;\,q_0]$ such that $Bx_0 - Cq_0 = 0$ *initial guess*
$p_0 = 0$ *initial Arnoldi vector*
$u_0 = b - Ax_0, \;\; t_0 = Cq_0$
$[\bar{p}_1\,;\,\bar{z}_1] \leftarrow$ solution of (17) with right-hand side $[u_0\,;\,-t_0]$
$p_1 = \bar{p}_1$
$q_1 = q_0 - \bar{z}_1$
$h_{1,0} = (p_1^T u_0 + q_1^T t_0)^{\frac{1}{2}}$
if $h_{1,0} \neq 0$ **then**
    $p_1 = p_1/h_{1,0}, \;\; q_1 = q_1/h_{1,0}$
end if
$k = 1$
while $h_{k,k-1} \neq 0$ **do**
    $u_k = Ap_k, \;\; t_k = Cq_k$
    $[\bar{p}_{k+1}\,;\,\bar{z}_{k+1}] \leftarrow$ solution of (17) with right-hand side $[u_k\,;\,-t_k]$
    $p_{k+1} = \bar{p}_{k+1}$
    $q_{k+1} = q_k - \bar{z}_{k+1}$
    for $i = 1, \ldots, k$ **do**
        $h_{i,k} = p_i^T u_k + q_i^T t_k$ $= p_i^T A p_k + q_i^T C q_k$
        $p_{k+1} = p_{k+1} - h_{i,k} p_i$
        $q_{k+1} = q_{k+1} - h_{i,k} q_i$
    end for
    $h_{k+1,k} = (p_{k+1}^T u_k + q_{k+1}^T t_k)^{\frac{1}{2}}$ $= (p_{k+1}^T A p_k + q_{k+1}^T C q_k)^{\frac{1}{2}}$
    if $h_{k+1,k} \neq 0$ **then**
        $p_{k+1} = p_{k+1}/h_{k+1,k}, \;\; q_{k+1} = q_{k+1}/h_{k+1,k}$
    end if
    $k = k + 1$
end while

THEOREM 4. *Let Algorithm 3 be initialized with $x_0 \in \mathbb{R}^n$ such that $Bx_0 = 0$, $q_0 = 0 \in \mathbb{R}^m$, and Algorithm 7 be initialized with the same x_0 and $y_0 \in \mathbb{R}^m$ such that (21) is satisfied. Then, for all $k \geqslant 0$, $v_{k,x} = p_k$ and $v_{k,y} = -q_k$, where $[v_{k,x}\,;\,v_{k,y}]$ is the k-th Arnoldi vector generated in Algorithm 7, and p_k and q_k are the k-th Arnoldi vectors generated in Algorithm 3. In addition, the scalars $h_{i,k}$ computed at each iteration are the same in both algorithms.*

Theorem 4 allows us to develop a constraint-preconditioned variant of any Krylov method based on the Arnoldi process, using a starting guess satisfying (21). Such variants are equivalent to their standard counterparts preconditioned with (2), but are computationally cheaper, as in the case of Lanczos-based methods. Furthermore, CP-Krylov versions of optimal Arnoldi-based Krylov methods preserve the minimization properties of these methods in the sense explained in Section 4. For example, the constraint-preconditioned version of GMRES (Saad and Schultz, 1986) minimizes the norm of the deviation of the residual from Range(N) similarly to MINRES. Below, we denote GMRES(ℓ) the variant of GMRES that is restarted every ℓ iterations.

Obtaining constraint-preconditioned versions of GMRES(ℓ) and DQGMRES is straighforward, by restarting and truncating the CP-Arnoldi basis generation process, respectively, as in the standard case (Saad, 2003). Note that DQGMRES with memory 2, i.e., with orthogonalization of each Arnoldi vector against the two previous

vectors only, is equivalent to CP-MINRES in exact arithmetic when A is symmetric. In finite precision arithmetic, DQGMRES with a larger memory may dampen the loss of orthogonality among the Lanczos vectors and act as a local reorthogonalization procedure, although we did not observe significant differences in Section 7.

Dollar (2007, Theorems 4.1 and 4.3) establishes that if C is positive semi-definite of rank p, $P^{-1}K$ has an eigenvalue at 1 of multiplicity $2m-p$, while the remaining $n-m+p$ eigenvalues are defined by a generalized eigenvalue problem. A remark after (Dollar, 2007, Theorem 4.1) states that Assumption 2.1 ensures that all eigenvalues are real. In addition, the dimension of the Krylov space is at most $\min(n-m+p+2,\, n+m)$. Inspection reveals that Dollar's proofs of those results do not use the fact that A is symmetric; the results hold for general A. Loghin (2017) establishes similar results on the eigenvalues of non-regularized saddle-point matrices for general A and general G. Clustering eigenvalues accelerates convergence of nonsymmetric Krylov solvers in many practical cases, although the convergence behavior of such solvers is not fully characterized by the eigenvalues (Greenbaum, Pták, and Strakoš, 1996).

6. Implementation Issues. We implemented the constraint-preconditioned variants of the Lanczos-CG, MINRES, SYMMLQ, GMRES(ℓ) and DQGMRES methods for (1) in a MATLAB library named `cpkrylov`. For completeness, we also included in the library an implementation of the CP-CG method in the form given by Dollar et al. (2006). We think that `cpkrylov` can be useful as a basis for the development of more sophisticated numerical software.

All solvers are accessed via a common interface exposed by the main driver `reg_cpkrylov()`, which performs pre-processing operations, calls the requested solver, performs post-processing operations, and returns solutions and statistics to the user. `cpkrylov` is freely available from `github.com/optimizers/cpkrylov`.

Because A is never required as an explicit matrix, we allow the user to supply it as an abstract linear operator as implemented in the Spot linear operator toolbox[2]. Spot allows us to use the familiar matrix notation with operators for which a representation as an explicit matrix is unavailable or inefficient. This affords the user flexibility in defining A while keeping the implementation of the various Krylov methods as readable as if A were a matrix.

Gould et al. (2001, 2014) observe that the numerical stability of projected Krylov solvers depends on keeping $[x_k\,;\, w_k]$ in Null(N). While the iterates lie in the nullspace in exact arithmetic, $[x_k\,;\, w_k]$ may have a non-negligible component in Range(N^T) because of roundoff error. In turn, the stability of CP-Krylov solvers depends on how accurately $[x_k\,;\, y_k]$ satisfies

$$Bx_k - Cy_k = 0.$$

Gould et al. (2001) suggest to increase the accuracy by applying iterative refinement after solving (17) with a direct method. In `cpkrylov`, the constraint preconditioner P is implemented as a Spot operator `P` such that writing `z = P*r`, where `z = [z1 ; z2]` and `r = [r1 ; r2]`, corresponds to solving

$$\begin{bmatrix} G & B^T \\ B & -C \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix}, \tag{26}$$

and automatically performing iterative refinement if requested by the user or if the residual norm of (26) exceeds a given tolerance.

[2] `www.cs.ubc.ca/labs/scl/spot`

An alternative approach to minimizing the size of the component of $[x_k\,;w_k]$ in Range(N^T) suggested by Gould et al. (2001) is to perform *iterative semi-refinement*. The latter consists in noting that the solution of (14) is not affected (in exact arithmetic) if we add a vector lying in Range(N^T) to $[u_x\,;u_w]$ in the right-hand side. Such a vector is available cheaply in the form of $[B^T\bar{z}\,;E^T\bar{z}]$ where $\bar{z}$ is the trailing segment of the solution of the most recent projection step (14), and $\bar{z}=0$ at the first projection step. The net result is that instead of (17), we solve

$$\begin{bmatrix} G & B^T \\ B & -C \end{bmatrix} \begin{bmatrix} \bar{p}_{k+1} \\ \bar{z}_{k+1} \end{bmatrix} = \begin{bmatrix} u_{k,x} - B^T\bar{z}_k \\ -(t_k - C\bar{z}_k) \end{bmatrix}, \quad \bar{z}_0 := 0. \tag{27}$$

By default, the matrix of (26) is factorized by way of MATLAB's `ldl()`. Spot allows us to separate the implementation of the preconditioner from that of other phases of solvers, so that future extensions to the former (e.g., the case where applying P results from a different factorization) will not require changes to the latter. Our implementation of P is transparent to the user, who must only pass the matrices G, B and C to `reg_cpkrylov()`.

All CP-Krylov solvers stop when

$$\|r_{\mathrm{P},k}\|_{[P]} \leqslant \epsilon_a + \|r_{\mathrm{P},0}\|_{[P]}\,\epsilon_r, \tag{28}$$

where $\|r_{\mathrm{P},k}\|_{[P]}$ is defined in (23) (or, equivalently, in (24)), and ϵ_a and ϵ_r are tolerances given by the user (default values are also set in our implementations). Note that, for all the CP-Krylov solvers except CP-DQGMRES, $\|r_{\mathrm{P},k}\|_{[P]}$ is obtained as a byproduct of other computations performed in algorithm. CP-DQGMRES computes an estimate of the residual norm only. A computationally cheap overestimate of the residual norm could be used in the stopping criterion, but this may unnecessarily increase the number of iterations (Saad and Wu, 1996, Section 3.1). A maximum number of iterations can be also specified for all solvers.

So far, we have considered the case where the last m entries of the right-hand side of (1) are zero. When the right-hand side has the general form $[b_1\,;b_2]$ with $b_2 \neq 0$, we can compute Δx and Δy such that

$$B\Delta x - C\Delta y = b_2, \tag{29}$$

by applying P to $[0\,;b_2]$, and subsequently solve (1) with $b = b_1 - A\Delta x - B^T\Delta y$. The solution of the original system is $[x+\Delta x\,;y+\Delta y]$. These pre- and post-processing steps are implemented in `reg_cpkrylov()`.

7. Numerical Experiments. We report results obtained by applying some solvers from the `cpkrylov` library to regularized saddle-point systems arising in the application of the primal-dual interior point solver PDCO to convex quadratic programming problems (see `web.stanford.edu/group/SOL/software/pdco/`). PDCO solves linearly constrained optimization problems with a smooth convex objective function in the form

$$\begin{array}{ll} \underset{x\in\mathbb{R}^n,\, r\in\mathbb{R}^m}{\text{minimize}} & f(x) + \frac{1}{2}\|D_1 x\|^2 + \frac{1}{2}\|r\|^2 \\ \text{subject to} & Bx + D_2 r = c \\ & l \leqslant x \leqslant u, \end{array} \tag{30}$$

where $f:\mathbb{R}^n \to \mathbb{R}$ is smooth and convex, $B \in \mathbb{R}^{m\times n}$, and D_1 and D_2 are positive-definite diagonal matrices that provide primal and dual regularization. In particular,

D_2 determines whether $Bx = c$ should be satisfied accurately or in the least-squares sense.

At each iteration of PDCO, a Newton step is applied to suitably perturbed KKT conditions associated with (30). The Newton step requires the solution of a linear system, which can be cast into the form (1) by a combination of permutation operations and/or inexpensive block eliminations. Possibly the most common saddle-point formulation is

$$K_2 = \begin{bmatrix} A & B^T \\ B & -C \end{bmatrix} = \begin{bmatrix} H + D_1^2 + X_1^{-1}Z_1 + X_2^{-1}Z_2 & B^T \\ B & -D_2^2 \end{bmatrix}$$

where H is the Hessian of the objective function at the current approximation of the optimal solution, $X_1 = \text{diag}(x_1)$, $X_2 = \text{diag}(x_2)$, $Z_1 = \text{diag}(z_1)$, $Z_2 = \text{diag}(z_2)$, $x_1 = x - l > 0$, $x_2 = u - x > 0$, and $z_1 > 0$ and $z_2 > 0$ are the corresponding dual variable estimates. In our experiments, H is constant because f is quadratic. More details are available from `web.stanford.edu/group/SOL/software/pdco/pdco.pdf`.

Recently, unreduced KKT systems have attracted the interest of researchers because of their better spectral properties, especially as the interior point iterates approach the solution of the optimization problem (Greif, Moulding, and Orban, 2014; Morini, Simoncini, and Tani, 2016). Other symmetric and unsymmetric saddle-point formulations are obtained with simple operations. In particular, within PDCO we used the unreduced symmetric saddle-point formulation

$$K_{3.5} = \begin{bmatrix} A & B^T \\ B & -C \end{bmatrix} = \left[\begin{array}{c|cc} H + D_1^2 & B^T & Z^{\frac{1}{2}} \\ \hline B & -D_2^2 & 0 \\ Z^{\frac{1}{2}} & 0 & -X \end{array}\right],$$

where $X = \text{diag}([x_1\,;\, x_2])$ and $Z = \text{diag}([z_1\,;\, z_2])$. We also considered the unsymmetric saddle-point formulation

$$K_{3p} = \begin{bmatrix} A & B^T \\ B & -C \end{bmatrix} = \left[\begin{array}{cc|c} H + D_1^2 & I & B^T \\ -Z & X & 0 \\ \hline B & 0 & -D_2^2 \end{array}\right],$$

which has the same structure as the saddle-point matrix in equation (2.5a) of (Greif et al., 2014) up to a permutation.

For all the saddle-point formulations, the constraint preconditioner P in (2) was defined by choosing G equal to the diagonal of the leading block A. This is a common choice in interior point methods—see, e.g., (D'Apuzzo et al., 2010). In our experiments, iterative refinement never needed to be performed.

The CP-Krylov solvers were stopped using an adaptive criterion that relates the accuracy in the solution of the KKT linear system to the duality measure at the current interior point iteration, as suggested by Cafieri, D'Apuzzo, De Simone, and di Serafino (2007b). Thus, criterion (28) was applied by setting $\epsilon_r = 0$ and

$$\epsilon_a = \max\left\{\min\left\{10^{-2}\mu, 10^{-2}\right\}, 10^{-6}\right\},$$

where μ is the the barrier parameter in PDCO.

We run PDCO on the CUTEst (Gould, Orban, and Toint, 2015) problems reported in Table 1. We use the models translated[3] into the AMPL modeling language (Fourer,

[3] `github.com/mpf/Optimization-Test-Problems`

Problem	K_2 size	$K_{3.5}$ (K_{3p}) size
cvxqp1_s	200	400
cvxqp1_m	2000	4000
cvxqp1_l	20000	40000
cvxqp2_s	150	350
cvxqp2_m	1500	3500
cvxqp2_l	15000	35000
cvxqp3_s	250	450
cvxqp3_m	2500	4500
cvxqp3_l	25000	45000
gouldqp3	1397	2795
gouldqp2	1397	2795
mosarqp1	3900	7100
mosarqp2	3900	3600
stcqp1	8201	16395
stcqp2	8201	16395

TABLE 1
CUTEst problems used in the experiments.

Gay, and Kernighan, 2002). Our version of PDCO has been modified to take an optimization problem in the form of an instance of the `nlpmodel` class as argument, which is defined in the `model` Matlab package.[4] The `amplmodel` subclass of `nlpmodel` reads an AMPL `nl` file by way of the `AmplMEXInterface` package[5] and conforms to the `model` interface expected by our version of PDCO. The rest of PDCO is identical to the original version. For the problem received by PDCO to have the form (30), it is necessary to introduce slack variables. In our implementation, linear inequalities $\ell \leqslant Ax \leqslant u$ are transformed to $Ax - s = 0$ and $\ell \leqslant s \leqslant u$. The transformation is performed by the `slackmodel` class, which receives an arbitrary instance of `nlpmodel`, including arbitrary instances of `amplmodel`, and adds slack variables as just described. The options passed to PDCO, including scaling parameters, are the same as those described by Orban (2015).

The problems are chosen so that H is non-diagonal; otherwise, for K_2 and $K_{3.5}$ the constraint preconditioner would be equal to the saddle-point matrix. The table also shows the sizes of K_2 and $K_{3.5}$ (or K_{3p}).

We ran PDCO with the saddle-point matrices K_2 and $K_{3.5}$, using CP-CG, CP-MINRES, CP-DQGMRES(ℓ) and CP-GMRES(ℓ) as Krylov solvers. By CP-DQGMRES(ℓ) we denote CP-DQGMRES with memory parameter ℓ, i.e., the number of Arnoldi vectors to be stored in the truncated CP-Arnoldi process. We set $\ell = 2$; in this case CP-DQGMRES is equivalent to CP-MINRES in exact arithmetic. We also ran PDCO with K_{3p} using CP-DQGMRES(ℓ) and CP-GMRES(ℓ) with various values of ℓ. The goal of the experiments is to illustrate the behavior of CP-Krylov solvers inside an interior-point method.

PDCO was run on a 2.5 GHz Intel Core i7 processor with 16 GB of RAM, 4 MB of L3 cache and the macOS 10.13.6 operating system, using MATLAB R2018b. Execution times were measured in seconds, by using the MATLAB function `timeit`, which removes some of the noise inherent to time measurements by calling a specified function multiple times and returning the median of the measurements.

Tables 2 to 5 summarize the results obtained with K_2 and $K_{3.5}$ using CP-CG and CP-MINRES. For each problem, "outer it" is the number of outer interior-point iterations, "inner it" is the cumulative number of inner Krylov iterations, "PDCO time" is the total run time reported by PDCO, and "prec time" and "solve time" are the

[4] github.com/optimizers/model
[5] github.com/optimizers/AmplMexInterface

name	outer it	inner it	PDCO time	prec time	solve time
cvxqp1_s	17	80	1.025e−01	4.137e−02	4.276e−02
cvxqp1_m	19	103	2.233e−01	7.196e−02	1.190e−01
cvxqp1_l	20	138	1.367e+00	4.195e−01	6.623e−01
cvxqp2_s	17	80	6.875e−02	3.357e−02	2.102e−02
cvxqp2_m	19	118	1.775e−01	4.506e−02	1.032e−01
cvxqp2_l	20	140	9.295e−01	1.468e−01	5.485e−01
cvxqp3_s	20	72	8.294e−02	4.222e−02	2.336e−02
cvxqp3_m	19	99	2.731e−01	1.098e−01	1.309e−01
cvxqp3_l	20	137	1.236e+00	5.764e−01	3.727e−01
gouldqp3	10	20	5.697e−02	2.671e−02	1.619e−02
gouldqp2	11	26	7.421e−02	3.219e−02	2.546e−02
mosarqp1	17	54	2.962e−01	9.205e−02	1.595e−01
mosarqp2	16	73	2.344e−01	8.506e−02	1.212e−01
stcqp1	15	124	4.959e+00	3.559e+00	1.219e+00
stcqp2	16	199	7.196e−01	1.006e−01	5.132e−01

TABLE 2
Results for K_2 with solver CP-CG. Times are in seconds.

name	outer it	inner it	PDCO time	prec time	solve time
cvxqp1_s	17	80	1.134e−01	4.936e−02	3.212e−02
cvxqp1_m	19	103	2.231e−01	7.145e−02	1.182e−01
cvxqp1_l	20	137	1.312e+00	4.185e−01	6.098e−01
cvxqp2_s	17	80	6.928e−02	3.350e−02	2.168e−02
cvxqp2_m	19	118	1.870e−01	4.766e−02	1.073e−01
cvxqp2_l	20	140	9.056e−01	1.506e−01	5.127e−01
cvxqp3_s	20	72	8.724e−02	4.401e−02	2.511e−02
cvxqp3_m	19	99	2.869e−01	1.164e−01	1.356e−01
cvxqp3_l	20	136	1.236e+00	5.739e−01	3.767e−01
gouldqp3	10	20	5.260e−02	2.453e−02	1.531e−02
gouldqp2	11	26	6.434e−02	2.757e−02	2.210e−02
mosarqp1	17	54	3.383e−01	9.735e−02	1.671e−01
mosarqp2	16	73	2.340e−01	8.128e−02	1.246e−01
stcqp1	15	124	5.115e+00	3.624e+00	1.307e+00
stcqp2	16	196	7.310e−01	1.017e−01	5.215e−01

TABLE 3
Results for K_2 with solver CP-MINRES. Times are in seconds.

cumulative times to assemble and factorize the constraint preconditioner and to solve the linear systems, respectively. We see that CP-MINRES performs a slightly smaller number of iterations than CP-CG on some problems, which may be beneficial if very large systems are solved. CP-MINRES is adequate in the context of a linesearch inexact Newton method such as PDCO because it reduces the residual norm monotonically by design. Fong and Saunders (2012) observe that MINRES possesses other desirable properties that are generally attributed to CG.

We also observe that the number of CP-Krylov iterations with $K_{3.5}$ is always smaller than with K_2, which may be due to the smaller condition number of $K_{3.5}$ (Greif et al., 2014; Morini et al., 2016). On cvxqp3_s, $K_{3.5}$ also results in a smaller number of PDCO iterations. On gouldqp2, $K_{3.5}$ results in fewer inner iterations than outer iterations because the initial guess satisfies the stopping condition of the first five subproblems, resulting in zero inner iterations for those outer iterations. This behavior does not occur with K_2, which produces different multiplier estimates. We used the MATLAB function `condest` to estimate the condition numbers of K_2 and $K_{3.5}$ encountered during the PDCO iterations for each problem. On the cvxqp problems, the largest value of `condest(`$K_{3.5}$`)` is between three and four orders of magnitude smaller than the largest value of `condest(`K_2`)`. The factor is between five and seven orders of magnitude on the gouldqp problems, one to two orders on the mosarqp problems, and two to three orders on the stcqp problems. While such

name	outer it	inner it	PDCO time	prec time	solve time
cvxqp1_s	17	66	1.162e−01	4.206e−02	3.165e−02
cvxqp1_m	19	86	3.685e−01	9.864e−02	2.275e−01
cvxqp1_l	20	120	3.761e+00	1.159e+00	2.220e+00
cvxqp2_s	17	64	7.814e−02	4.010e−02	2.098e−02
cvxqp2_m	19	101	2.769e−01	6.854e−02	1.712e−01
cvxqp2_l	20	123	2.483e+00	4.038e−01	1.753e+00
cvxqp3_s	18	50	8.543e−02	4.123e−02	2.774e−02
cvxqp3_m	19	81	4.584e−01	1.793e−01	2.384e−01
cvxqp3_l	20	119	4.253e+00	1.726e+00	2.163e+00
gouldqp3	10	12	5.470e−01	4.725e−01	5.119e−02
gouldqp2	9	5	8.842e−02	4.019e−02	3.323e−02
mosarqp1	17	39	5.908e−01	2.126e−01	3.048e−01
mosarqp2	16	60	3.951e−01	1.590e−01	2.025e−01
stcqp1	15	110	5.622e+00	3.541e+00	1.883e+00
stcqp2	16	183	1.841e+00	1.721e−01	1.527e+00

TABLE 4
Results for $K_{3.5}$ with solver CP-CG. Times are in seconds.

measurements do not tell the whole story and it would be more accurate to measure the condition number of (10), they tend to confirm that the condition number of $K_{3.5}$ is provably bounded if strict complementarity is satisfied. On our test set, the PDCO time reported for formulation K_2 is almost always slightly smaller than that for $K_{3.5}$.

name	outer it	inner it	PDCO time	prec time	solve time
cvxqp1_s	17	65	9.533e−02	4.257e−02	3.340e−02
cvxqp1_m	19	86	3.882e−01	1.030e−01	2.462e−01
cvxqp1_l	20	119	3.362e+00	9.801e−01	2.039e+00
cvxqp2_s	17	64	6.759e−02	3.480e−02	1.855e−02
cvxqp2_m	19	101	2.633e−01	6.210e−02	1.668e−01
cvxqp2_l	20	123	2.466e+00	3.748e−01	1.776e+00
cvxqp3_s	18	50	8.394e−02	4.090e−02	2.741e−02
cvxqp3_m	19	81	4.505e−01	1.737e−01	2.369e−01
cvxqp3_l	20	118	4.272e+00	1.758e+00	2.142e+00
gouldqp3	10	12	5.600e−01	4.870e−01	5.158e−02
gouldqp2	9	5	8.700e−02	4.047e−02	3.243e−02
mosarqp1	17	39	6.036e−01	2.265e−01	3.166e−01
mosarqp2	16	60	4.167e−01	1.646e−01	2.168e−01
stcqp1	15	109	5.648e+00	3.556e+00	1.895e+00
stcqp2	16	180	1.827e+00	1.730e−01	1.510e+00

TABLE 5
Results for $K_{3.5}$ with solver CP-MINRES. Times are in seconds.

We do not show the details for CP-DQGMRES(2) and CP-GMRES(2), because they do not add much to the discussion. We summarize the results as follows: CP-DQGMRES(2) results in the same number of PDCO and CP-Krylov iterations as CP-MINRES, as expected, and the corresponding times are comparable with those of MINRES. CP-GMRES(2) results in an increase in the number of CP-Krylov iterations as compared with MINRES. Whereas CP-DQGMRES with $\ell > 2$ may be viewed as CP-MINRES with a form of partial reorthogonalization, setting $\ell = 4$ did not yield any improvement on the symmetric formulations.

The results with CP-DQGMRES(ℓ) and CP-GMRES(ℓ) on K_{3p} are not favorable. In general, the unsymmetric CP-Krylov solvers on K_{3p} are much less efficient than the symmetric ones on K_2 and $K_{3.5}$. For example, with $\ell = 500$ the number of CPKrylov iterations is much larger than in the symmetric case and there are some problems where CP-DQGMRES(ℓ) and CP-GMRES(ℓ) cannot always satisfy the stopping criterion. In these cases, they halt because a maximum number of CP-Krylov iterations equal to $2n$ is achieved, thus preventing PDCO from computing the optimal solution by its maximum number of iterations, which is set as $\min\{\max\{30, n\}, 50\}$. Among the

possible reasons, we mention the non-normality of K_{3p} and the choice of the $(1,1)$ block G of the preconditioner. A similar behavior has been observed by using the MATLAB function `gmres` with the constraint preconditioner. A more efficient choice of G and a better formulation than K_{3p} are the subject of further investigation.

We ran all our tests a second time with iterative semi-refinement (27) activated, but did not observe any difference in the number of inner or outer iterations.

8. Discussion. We extended the approach of Gould et al. (2014) to saddle-point systems with regularization and provided principles from which to derive constrained-preconditioned iterative methods. The resulting methods are conceptually equivalent to standard iterative methods applied to a reduced system in a way that preserves their properties, including quantities that increase or decrease monotonically at each iteration. Specifically, we discussed constraint-preconditioned versions of the CG-Lanczos, MINRES, SYMMLQ, GMRES(ℓ) and DQGMRES(ℓ) methods, and showed that they preserve the properties of the corresponding standard methods in a suitable reduced Krylov space. We illustrated our approach on methods based on the Lanczos and Arnoldi processes, but it applies equally to other processes, including those of Golub and Kahan (1965), Saunders, Simon, and Yip (1988), and the unsymmetric Lanczos (1952) bi-orthogonalization process. We implemented our constraint-preconditioned methods in a MATLAB library named `cpkrylov` that provides a basis for the development of more sophisticated numerical software.

An open question related to constraint preconditioners concerns the best way to reduce their computational cost. Inexact constraint preconditioners have been developed and analyzed, based on approximations of the Schur complement of the leading block of the constraint preconditioner or on other approximations (Lukšan and Vlček, 1998; Perugia and Simoncini, 2000; Durazzi and Ruggiero, 2003; Bergamaschi, Gondzio, Venturin, and Zilli, 2007; Sesana and Simoncini, 2013). Preconditioner updating techniques, producing inexact and exact constraint preconditioners, have been also proposed in order to reduce the cost of solving sequences of saddle-point systems (Bellavia, De Simone, di Serafino, and Morini, 2015, 2016; Fisher, Gratton, Gürol, Trémolet, and Vasseur, 2016; Bergamaschi, De Simone, di Serafino, and Martínez, 2018). It must be noted, however, that the inexact constraint preconditioners considered so far generally do not produce preconditioned vectors lying in the nullspace of the matrix N defined in (7), which is a key issue to obtain CP-preconditioned methods for (1) equivalent to suitably preconditioned Krylov methods for (9). On the other hand, inexact preconditioners have proven effective in reducing the computational time for the solution of large-scale saddle-point systems. A further possibility for lowering the cost of constraint preconditioners is to apply them inexactly using an iterative method. Of course, preserving the property of obtaining preconditioned vectors lying in the nullspace of N is a major issue. To the best of our knowledge, this approach has not been yet addressed in the literature.

Finally, it is worth investigating the choice of the $(1,1)$ block of the constraint preconditioner when solving non-normal saddle-point systems.

Acknowledgments. We thank two anonymous reviewers for constructive comments that helped us greatly improve the numerical experiments section.

References.

W. E. Arnoldi. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Q. Appl. Math.*, 9:17–29, 1951.

S. Bellavia, V. De Simone, D. di Serafino, and B. Morini. Updating constraint preconditioners

for KKT systems in quadratic programming via low-rank corrections. *SIAM J. Optim.*, 25 (3):1787–1808, 2015.

S. Bellavia, V. De Simone, D. di Serafino, and B. Morini. On the update of constraint preconditioners for regularized KKT systems. *Comput. Optim. Appl.*, 65(2):339–360, 2016.

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

L. Bergamaschi, J. Gondzio, M. Venturin, and G. Zilli. Inexact constraint preconditioners for linear systems arising in interior point methods. *Comput. Optim. Appl.*, 36(2–3):137–147, 2007.

L. Bergamaschi, V. De Simone, D. di Serafino, and A. Martínez. BFGS-like updates of constraint preconditioners for sequences of KKT linear systems in quadratic programming. *Numer. Linear Algebra Appl.*, 25(5):e2144, 2018.

C. Brezinski and M. Redivo-Zaglia. Transpose-free Lanczos-type algorithms for nonsymmetric linear systems. *Numer. Algor.*, 17(1–2):67–103, 1998.

S. Cafieri, M. D'Apuzzo, V. De Simone, and D. di Serafino. On the iterative solution of KKT systems in potential reduction software for large-scale quadratic problems. *Comput. Optim. Appl.*, 38(1):27–45, 2007a.

S. Cafieri, M. D'Apuzzo, V. De Simone, and D. di Serafino. Stopping criteria for inner iterations in inexact potential reduction methods: a computational study. *Comput. Optim. Appl.*, 36(2–3):165–193, 2007b.

T. F. Chan, L. de Pillis, and H. van der Vorst. Transpose-free formulations of Lanczos-type methods for nonsymmetric linear systems. *Numer. Algor.*, 17(1–2):51–66, 1998.

M. D'Apuzzo, V. De Simone, and D. di Serafino. On mutual impact of numerical linear algebra and large-scale optimization with focus on interior point methods. *Comput. Optim. Appl.*, 45(2):283–310, 2010.

V. De Simone, D. di Serafino, and B. Morini. On preconditioner updates for sequences of saddle-point linear systems. *Communications in Applied and Industrial Mathematics*, 9(1): 35–41, 2018.

H. S. Dollar. Constraint-style preconditioners for regularized saddle point problems. *SIAM J. Matrix Anal. Appl.*, 29(2):672–684, 2007.

H. S. Dollar, N. I. M. Gould, W. H. A. Schilders, and A. J. Wathen. Implicit-factorization preconditioning and iterative solvers for regularized saddle-point systems. *SIAM J. Matrix Anal. Appl.*, 28(1):170–189, 2006.

I. S. Duff. MA57—a code for the solution of sparse symmetric definite and indefinite systems. *ACM Trans. Math. Software*, 30(2):118–144, 2004.

C. Durazzi and V. Ruggiero. Indefinitely preconditioned conjugate gradient method for large sparse equality and inequality constrained quadratic problems. *Numer. Linear Algebra Appl.*, 10(8):673–688, 2003.

M. Fisher, S. Gratton, S. Gürol, Y. Trémolet, and X. Vasseur. Low rank updates in preconditioning the saddle point systems arising from data assimilation problems. *Optim. Method Softw.*, 33(1):45–69, 2016.

D. C.-L. Fong and M. A. Saunders. CG versus MINRES: An empirical comparison. *SQU Journal for Science*, 17(1):44–62, 2012.

R. Fourer, D. M. Gay, and B. W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. Brooks/Cole, Pacific Grove, second edition, 2002. https://ampl.com/resources/the-ampl-book.

M. P. Friedlander and D. Orban. A primal-dual regularized interior-point method for convex quadratic problems. *Math. Program. Comp.*, 4(1):71–107, 2012.

G. H. Golub and W. Kahan. Calculating the singular values and pseudo-inverse of a matrix. *SIAM J. Numer. Anal.*, 2(2):205–224, 1965.

N. Gould, D. Orban, and Ph. Toint. CUTEst: a constrained and unconstrained testing environment with safe threads for mathematical optimization. *Comput. Optim. Appl.*, 60 (3):545–557, 2015.

N. I. M. Gould, M. E. Hribar, and J. Nocedal. On the solution of equality constrained

quadratic problems arising in optimization. *SIAM J. Sci. Comput.*, 23(4):1375–1394, 2001.
N. I. M. Gould, D. Orban, and T. Rees. Projected Krylov methods for saddle-point systems. *SIAM J. Matrix Anal. Appl.*, 35(4):1329–1343, 2014.
A. Greenbaum, V. Pták, and Z. Strakoš. Any nonincreasing convergence curve is possible for GMRES. *SIAM J. Matrix Anal. Appl.*, 17(3):465–469, 1996.
C. Greif, E. Moulding, and D. Orban. Bounds on eigenvalues of matrices arising from interior-point methods. *SIAM J. Optim.*, 24(1):49–83, 2014.
M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *J. Res. Natl. Bur. Stand.*, 49(6):409–436, 1952.
C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. Natl. Bur. Stand.*, 45:225–280, 1950.
C. Lanczos. Solution of systems of linear equations by minimized iterations. *J. Res. Natl. Bur. Stand.*, 49(1):33–53, 1952.
D. Loghin. A note on constraint preconditioning. *SIAM J. Matrix Anal. Appl.*, 38(4): 1486–1495, 2017.
L. Lukšan and J. Vlček. Indefinitely preconditioned inexact Newton method for large sparse equality constrained nonlinear programming problems. *Numer. Linear Algebra Appl.*, 5: 219–247, 1998.
B. Morini, V. Simoncini, and M. Tani. Spectral estimates for unreduced symmetric KKT systems arising from interior point methods. *Numer. Linear Algebra Appl.*, 23:776–800, 2016.
D. Orban. Limited-memory LDL^T factorization of symmetric quasi-definite matrices with application to constrained optimization. *Numer. Algor.*, 70(1):9–41, 2015.
C. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.*, 12(4):617–629, 1975.
I. Perugia and V. Simoncini. Block-diagonal and indefinite symmetric preconditioners for mixed finite element formulations. *Numer. Linear Algebra Appl.*, 7(7–8):585–616, 2000.
J. Pestana and A. J. Wathen. Natural preconditioning and iterative methods for saddle point systems. *SIAM Review*, 57(1):51–71, 2015.
Y. Saad. *Iterative methods for sparse linear systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, second edition, 2003.
Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. and Statist. Comput.*, 7(3):856–869, 1986.
Y. Saad and K. Wu. DQGMRES: a direct quasi-minimal residual algorithm based on incomplete orthogonalization. *Numer. Linear Algebra Appl.*, 3(4):329–343, 1996.
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.
D. Sesana and V. Simoncini. Spectral analysis of inexact constraint preconditioning for symmetric saddle point matrices. *Linear Algebra Appl.*, 438(6):2683–2700, 2013.

Appendix A. Standard Lanczos and Arnoldi Processes. For reference we state the preconditioned Lanczos process and the full-space Lanczos process for (1) with preconditioner (2). We also state the projected and full-space Arnoldi processes.

Algorithm 4 Lanczos Process for $Ax = b$ with Preconditioner $J = J^T \succ 0$

choose x_0
$v_0 = 0$
$r_0 = b - Ax_0$
solve $Jv_1 = r_0$
$\beta_1 = (v_1^T r_0)^{\frac{1}{2}}$
if $\beta_1 \neq 0$ **then**
    $v_1 = v_1/\beta_1$ $\quad \|v_1\|_J = 1$
end if
$k = 1$
while $\beta_k \neq 0$ **do**
    $u_k = Av_k$
    $\alpha_k = u_k^T v_k$
    solve $Jv_{k+1} = u_k$
    $v_{k+1} = v_{k+1} - \alpha_k v_k - \beta_k v_{k-1}$
    $\beta_{k+1} = (v_{k+1}^T u_k)^{\frac{1}{2}}$
    if $\beta_{k+1} \neq 0$ **then**
        $v_{k+1} = v_{k+1}/\beta_k$ $\quad \|v_{k+1}\|_J = 1$
    end if
    $k = k + 1$
end while

Algorithm 5 Full-Space Lanczos Process for (1) with Preconditioner (2)

choose $[x_0\,;\,y_0]$ such that $Bx_0 - Cy_0 = 0$

initialize

$$\begin{bmatrix} v_{0,x} \\ v_{0,y} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

set $\qquad Bx_0 - Cy_0 = 0 \;\Rightarrow\; r_{0,y} = 0$

$$\begin{bmatrix} r_{0,x} \\ r_{0,y} \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix} - \begin{bmatrix} A & B^T \\ B & -C \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \begin{bmatrix} b - Ax_0 - B^T y_0 \\ 0 \end{bmatrix}$$

obtain v_1 as the solution of

$$\begin{bmatrix} G & B^T \\ B & -C \end{bmatrix} \begin{bmatrix} v_{1,x} \\ v_{1,y} \end{bmatrix} = \begin{bmatrix} r_{0,x} \\ r_{0,y} \end{bmatrix}$$

$\beta_1 = (v_1^T r_0)^{\frac{1}{2}} = (v_{1,x}^T r_{0,x})^{\frac{1}{2}}$
if $\beta_1 \neq 0$ **then**
    $v_1 = v_1/\beta_1$
end if
$k = 1$
while $\beta_k \neq 0$ **do**
    compute

$$\begin{bmatrix} u_{k,x} \\ u_{k,y} \end{bmatrix} = \begin{bmatrix} A & B^T \\ B & -C \end{bmatrix} \begin{bmatrix} v_{k,x} \\ v_{k,y} \end{bmatrix}$$

    $\alpha_k = u_k^T v_k = v_{k,x}^T A v_{k,x} + 2 v_{k,x}^T B^T v_{k,y} - v_{k,y}^T C v_{k,y}$
    obtain v_{k+1} as the solution of

$$\begin{bmatrix} G & B^T \\ B & -C \end{bmatrix} \begin{bmatrix} v_{k+1,x} \\ v_{k+1,y} \end{bmatrix} = \begin{bmatrix} u_{k,x} \\ u_{k,y} \end{bmatrix}$$

    $v_{k+1} = v_{k+1} - \alpha_k v_k - \beta_k v_{k-1}$
    $\beta_{k+1} = (v_{k+1}^T u_k)^{\frac{1}{2}} = (v_{k+1,x}^T u_{k,x} + v_{k+1,y}^T u_{k,y})^{\frac{1}{2}}$
    if $\beta_{k+1} \neq 0$ **then**
        $v_{k+1} = v_{k+1}/\beta_{k+1}$
    end if
    $k = k + 1$
end while

Algorithm 6 Projected Arnoldi Process

choose $[x_0\,;\,w_0]$ such that $Bx_0 + Ew_0 = 0$
$v_{0,x} = 0,\;\; v_{0,w} = -w_0$
$u_{0,x} = b - Ax_0,\; u_{0,w} = -F^{-1}w_0$
$[\bar{u}_{1,x}\,;\,\bar{u}_{1,w}\,;\,\bar{z}_1] \leftarrow$ solution of (14) with right-hand side $[u_{0,x}\,;\,u_{0,w}\,;\,0]$
$v_{1,x} = \bar{u}_{1,x},\;\; v_{1,w} = \bar{u}_{1,w}$ $\qquad v_1 = P_G\, u_0$
$h_{1,0} = (v_{1,x}^T u_{0,x} + v_{1,w}^T u_{0,w})^{\frac{1}{2}}$
if $h_{1,0} \neq 0$ **then**
    $v_{1,x} = v_{1,x}/h_{1,0},\;\; v_{1,w} = v_{1,w}/h_{1,0}$
end if
$k = 1$
while $h_{k,k-1} \neq 0$ **do**
    $u_{k,x} = Av_{k,x},\;\; u_{k,w} = F^{-1}v_{k,w}$
    $[\bar{u}_{k+1,x}\,;\,\bar{u}_{k+1,w}\,;\,\bar{z}_{k+1}] \leftarrow$ solution of (14) with right-hand side $[u_{k,x}\,;\,u_{k,w}\,;\,0]$
    $v_{k+1,x} = \bar{u}_{k+1,x},\;\; v_{k+1,w} = \bar{u}_{k+1,w}$ $\qquad v_{k+1} = P_G\, u_k$
    for $i = 1, \ldots, k$ **do**
        $h_{i,k} = v_{i,x}^T u_{k,x} + v_{i,w}^T u_{k,w}$
        $v_{k+1,x} = v_{k+1,x} - h_{i,k}v_{i,x},\;\; v_{k+1,w} = v_{k+1,w} - h_{i,k}v_{i,w}$
    end for
    $h_{k+1,k} = (v_{k+1,x}^T u_{k,x} + v_{k+1,w}^T u_{k,w})^{\frac{1}{2}}$
    if $h_{k+1,k} \neq 0$ **then**
        $v_{k+1,x} = v_{k+1,x}/h_{k+1,k},\;\; v_{k+1,w} = v_{k+1,w}/h_{k+1,k}$
    end if
    $k = k + 1$
end while

Algorithm 7 Full-Space Arnoldi Process for (1) with Preconditioner (2)

choose $[x_0\,;\,y_0]$ such that $Bx_0 - Cy_0 = 0$

initialize

$$\begin{bmatrix} v_{0,x} \\ v_{0,y} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

set $\qquad Bx_0 - Cy_0 = 0 \;\Rightarrow\; r_{0,y} = 0$

$$\begin{bmatrix} r_{0,x} \\ r_{0,y} \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix} - \begin{bmatrix} A & B^T \\ B & -C \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \begin{bmatrix} b - Ax_0 - B^T y_0 \\ 0 \end{bmatrix}$$

obtain v_1 as the solution of

$$\begin{bmatrix} G & B^T \\ B & -C \end{bmatrix} \begin{bmatrix} v_{1,x} \\ v_{1,y} \end{bmatrix} = \begin{bmatrix} r_{0,x} \\ r_{0,y} \end{bmatrix}$$

$h_{1,0} = (v_1^T r_0)^{\frac{1}{2}} = (v_{1,x}^T r_{0,x})^{\frac{1}{2}}$
if $h_{1,0} \neq 0$ **then**
    $v_1 = v_1/\beta_1$
end if
$k = 1$
while $h_{k,k-1} \neq 0$ **do**
    compute

$$\begin{bmatrix} u_{k,x} \\ u_{k,y} \end{bmatrix} = \begin{bmatrix} A & B^T \\ B & -C \end{bmatrix} \begin{bmatrix} v_{k,x} \\ v_{k,y} \end{bmatrix}$$

    obtain v_{k+1} as the solution of

$$\begin{bmatrix} G & B^T \\ B & -C \end{bmatrix} \begin{bmatrix} v_{k+1,x} \\ v_{k+1,y} \end{bmatrix} = \begin{bmatrix} u_{k,x} \\ u_{k,y} \end{bmatrix}$$

    for $i = 1, \ldots, k$ **do**
        $h_{i,k} = v_i^T u_k = v_{i,x}^T A v_{k,x} + 2 v_{i,x}^T B^T v_{k,y} - v_{i,y}^T C v_{k,y}$
        $v_{k+1} = v_{k+1} - h_{i,k} v_i$
    end for
    $h_{k+1,k} = (v_{k+1}^T u_k)^{\frac{1}{2}} = (v_{k+1,x}^T u_{k,x} + v_{k+1,y}^T u_{k,y})^{\frac{1}{2}}$
    if $h_{k+1,k} \neq 0$ **then**
        $v_{k+1} = v_{k+1}/h_{k+1,k}$
    end if
    $k = k + 1$
end while