SIAM J. Matrix Anal. Appl.
Vol. 35, No. 4, pp. 1329–1343

PROJECTED KRYLOV METHODS FOR SADDLE-POINT SYSTEMS[*]

NICK GOULD[†], DOMINIQUE ORBAN[‡], AND TYRONE REES[†]

Abstract. Projected Krylov methods are full-space formulations of Krylov methods that take place in a nullspace. Provided projections into the nullspace can be computed accurately, those methods only require products between an operator and vectors lying in the nullspace. We provide systematic principles for obtaining the projected form of any well-defined Krylov method. Projected Krylov methods are mathematically equivalent to constraint-preconditioned Krylov methods provided the initial guess is well chosen, but require less memory. As a consequence, there are situations where certain known methods such as MINRES and SYMMLQ are well defined in the presence of an indefinite preconditioner.

Key words. saddle-point system, projected Krylov method, constraint preconditioner

AMS subject classifications. 15A06, 65F08, 65F10, 65F50, 65H10, 90C20

DOI. 10.1137/130916394

1. Introduction.

We consider the solution of the saddle-point problem

$$\begin{bmatrix} \boldsymbol{Q} & \boldsymbol{A}^{\mathrm{T}} \\ \boldsymbol{A} & \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_* + \boldsymbol{x}_{\mathrm{F}} \\ \boldsymbol{y}_* \end{bmatrix} = \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{b} \end{bmatrix} \tag{1.1}$$

for some fixed $\boldsymbol{x}_{\mathrm{F}}$, where all data are real and $\boldsymbol{Q}$ may be unsymmetric. Such saddle-point systems arise throughout computational science (Benzi, Golub, and Liesen, 2005). Typical applications arise in conservative fluid flow, structural engineering, and constrained optimization. The scale of the problem often precludes a direct factorization of the matrix $\boldsymbol{K}_Q$ in (1.1). Certain applications give rise to a symmetric saddle-point system, including optimization and the study of laminar fluid flow. A common approach in such cases is to employ a Krylov method for symmetric indefinite systems, e.g., MINRES or SYMMLQ (Paige and Saunders, 1975), combined with an appropriate preconditioner (Elman, Silvester, and Wathen, 2005).

In this paper we build upon the work of Gould, Hribar, and Nocedal (2001) and highlight families of iterative methods working implicitly in the nullspace of $\boldsymbol{A}$, requiring only operator-vector products with $\boldsymbol{Q}$ (and possibly its transpose) as well as one or more projections of a vector into the nullspace of $\boldsymbol{A}$ per iteration. We require that $\boldsymbol{Q}$ be available as an operator, and need accurate projections into the nullspace of $\boldsymbol{A}$. Gould, Hribar, and Nocedal (2001) describe how to compute sufficiently accurate projections at moderate cost by way of iterative refinement or residual updates.

Our main contribution is to provide systematic principles to derive a projected variant from any well-defined Krylov method. This is made possible by working at the level of the basis construction processes upon which those methods are built. An additional contribution is to provide equivalence relations in the general case between

[*] Received by the editors April 10, 2013; accepted for publication (in revised form) by M. Benzi August 20, 2014; published electronically November 4, 2014.
http://www.siam.org/journals/simax/35-4/91639.html

[†] Scientific Computing Department, Rutherford Appleton Laboratory, Chilton, Oxfordshire OX11 0QX, England (nick.gould@stfc.ac.uk, tyrone.rees@stfc.ac.uk). The research of these authors was supported by EPSRC grant EP/I013067/1.

[‡] GERAD and Department of Mathematics and Industrial Engineering, École Polytechnique, Montréal, QC, Canada (dominique.orban@gerad.ca). The research of this author was partially supported by an NSERC Discovery grant.

projected Krylov methods and traditional Krylov methods applied directly to (1.1) with a so-called *constraint* preconditioner.

As we mention in section 4, the principles highlighted below are applicable beyond the strict context of Krylov methods. In particular, they may be applied to derive projected variants of iterative methods based on Krylov-type basis-generation processes. An example is given by Orban (2014), who derives a projected variant of the Golub and Kahan (1965) process and applies it to equality-constrained least-squares problems.

Our approach involves a sequence of key steps. We first reduce the saddle-point problem to an equivalent one in the nullspace of $\boldsymbol{A}$ and then transform (precondition) the result. We next apply an appropriate Krylov method to the resulting preconditioned, reduced system. The effects of the preconditioner on the method are then considered in the unpreconditioned reduced space, and finally the iteration is moved back from the nullspace via a constraint preconditioner into its original full-space setting. Our framework is closely related to the *nullspace method*—see (Benzi, Golub, and Liesen, 2005, section 6) and references therein—but differs from it in that we only use the data of (1.1). Similarly to Gould, Hribar, and Nocedal (2001), we do not require that a basis for $\mathrm{Null}(\boldsymbol{A})$ be computed.

An interesting consequence of our framework is to establish that well-known methods such as MINRES and SYMMLQ applied to (1.1) may be well defined in the presence of an indefinite preconditioner. Specifically, MINRES minimizes the residual in the seminorm defined by the constraint preconditioner which, due to the structure of the residuals, results in a well-defined method—we give more detail in section 3. Similar considerations hold for SYMMLQ, and the equivalent result was shown by Rozložník and Simoncini (2002) to be true for definite systems. This is at variance with the commonly issued warnings (e.g., Elman, Silvester, and Wathen, 2005, p. 287; Greenbaum, 1997, p. 121; and van der Vorst, 2003, p. 85) that MINRES and SYMMLQ require definite preconditioners.

In section 2, we examine a number of well-known Krylov-subspace methods for saddle-point systems. Specifically, in section 2.1, we show how methods may be constructed to reflect the saddle-point structure; in section 2.2, we detail the bases used and how standard methods appear when moved into the subspace defined by such structure; and in section 2.3 we return these methods back into the original space. In section 2.4 we show that our construction is equivalent to applying standard methods with a constraint preconditioner. We illustrate this equivalence on the MINRES method in section 3 by showing that a positive definite preconditioner is not required when applying the method to saddle-point systems. We briefly consider other variants in section 4, and conclude in section 5.

1.1. Related research.

In optimization contexts, where $\boldsymbol{Q}$ is symmetric, (1.1) may be interpreted as the first-order optimality conditions of the quadratic program

$$\text{(1.2)}\quad \underset{\boldsymbol{x}+\boldsymbol{x}_{\mathrm{F}}}{\text{minimize}}\; -\boldsymbol{a}^{\mathsf{T}}(\boldsymbol{x}+\boldsymbol{x}_{\mathrm{F}}) + \tfrac{1}{2}(\boldsymbol{x}+\boldsymbol{x}_{\mathrm{F}})^{\mathsf{T}}\boldsymbol{Q}(\boldsymbol{x}+\boldsymbol{x}_{\mathrm{F}}) \quad \text{subject to}\; \boldsymbol{A}(\boldsymbol{x}+\boldsymbol{x}_{\mathrm{F}}) = \boldsymbol{b},$$

where $\boldsymbol{y}$ are the Lagrange multipliers associated with the equality constraints. It is well known (see, e.g., Gould, 1985), that (1.2) possesses a unique solution if and only if $\boldsymbol{Q}$ is positive definite on the nullspace of $\boldsymbol{A}$. Based on this, Gould, Hribar, and Nocedal (2001) devised the projected conjugate gradient method, a variant of the standard conjugate gradient algorithm applied to $\boldsymbol{Q}$ and restricted to exploration of the nullspace of $\boldsymbol{A}$—see also Polyak (1969), Coleman (1994), Lukšan and Vlček (1998), and Perugia and Simoncini (2000). Benzi, Golub, and Liesen (2005, section 6) describe and provide numerous references on the nullspace method, which requires the computation of a basis for Null($\boldsymbol{A}$). Numerical challenges quickly accumulate as desirable properties for this basis, such as sparsity and good conditioning, typically come at high expense. As it turns out, Gould, Hribar, and Nocedal (2001) show that it is possible to avoid computing a basis for the nullspace of $\boldsymbol{A}$ altogether and formulate the entire algorithm in terms of projections into this nullspace and operator-vector products with $\boldsymbol{Q}$. One way to compute projections efficiently when $\boldsymbol{A}$ is available as an explicit matrix is to perform a one-time factorization of a symmetric indefinite matrix of the form (1.1) where $\boldsymbol{Q}$ is replaced by a simpler operator such as (but not restricted to) the identity. In many applications, performing this factorization is realistic and cost effective; for example, one may choose the (1,1) block so that a convenient Schilders' factorization (Dollar and Wathen, 2006) may be employed.

Orban (2008) applies the same principles as those provided by Gould, Hribar, and Nocedal (2001) to specific Krylov methods for unsymmetric systems with the solution of fluid flow problems in mind. He focuses on methods not requiring products with the transpose operator.

1.2. Terminology and notation.

We refer to the quantities $\boldsymbol{Q}$, $\boldsymbol{A}$, $\boldsymbol{x}$, etc., of (1.1) as *full-space* quantities. Throughout the paper, they are typeset in italicized bold Roman font. Whenever we work explicitly in the nullspace of $\boldsymbol{A}$, we refer to corresponding quantities as *reduced-space* quantities and typeset them in italicized Roman light face font, e.g., Q, A, x. We precondition reduced-space quantities and refer to them as *preconditioned-space* quantities. They are typeset in an italicized bold sans-serif font, e.g., $\boldsymbol{\mathsf{Q}}$, $\boldsymbol{\mathsf{A}}$, $\boldsymbol{\mathsf{x}}$. The Euclidean norm and its associated inner product are denoted $\|\cdot\|$ and $\langle\cdot,\cdot\rangle$, respectively, throughout.

2. Saddle-point problems.

Throughout the paper, our working assumption is the following.

Assumption 2.1. The coefficient matrix of (1.1) is nonsingular.

Benzi, Golub, and Liesen (2005, Theorem 3.4) provide necessary and sufficient conditions for Assumption 2.1 to hold. In particular, if $\boldsymbol{H} = \frac{1}{2}(\boldsymbol{Q} + \boldsymbol{Q}^{\mathsf{T}})$ denotes the symmetric part of $\boldsymbol{Q}$, and if we assume that $\boldsymbol{H}$ is positive semidefinite and $\boldsymbol{A}$ has full row rank, then

- (1.1) is nonsingular if $\text{Null}(\boldsymbol{H}) \cap \text{Null}(\boldsymbol{A}) = \{\boldsymbol{0}\}$, and
- $\text{Null}(\boldsymbol{Q}) \cap \text{Null}(\boldsymbol{A}) = \{\boldsymbol{0}\}$ if (1.1) is nonsingular,

but the reverse implications do not hold in general. The above conditions are sufficiently general to encompass numerous applications of interest; we refer the interested reader to (Benzi, Golub, and Liesen, 2005) for more details. At least in theory, it is possible to ensure that $\boldsymbol{A}$ has full row rank by way of preprocessing. In some cases, regularization is preferred and consists in giving a nonzero value to the $(2,2)$ block of (1.1). In section 4, we explain how the methods presented in the next sections apply to such a regularized system.

Suppose that $\boldsymbol{x}_{\text{F}}$ satisfies $\boldsymbol{A}\boldsymbol{x}_{\text{F}} = \boldsymbol{b}$; as we will see, finding such a value is easy in the framework we develop. Substituting into (1.1), $\boldsymbol{x}_*$ and $\boldsymbol{y}_*$ satisfy

$$\begin{bmatrix} \boldsymbol{Q} & \boldsymbol{A}^{\mathsf{T}} \\ \boldsymbol{A} & \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_* \\ \boldsymbol{y}_* \end{bmatrix} = \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{0} \end{bmatrix}, \tag{2.1}$$

where $\boldsymbol{c} := \boldsymbol{a} - \boldsymbol{Q}\boldsymbol{x}_{\text{F}}$. Our aim is thus to solve (2.1).

Let $\boldsymbol{Z}$ be any full-rank matrix whose columns span the nullspace of $\boldsymbol{A}$, i.e., such that

$$\boldsymbol{A}\boldsymbol{Z} = \mathbf{0}. \tag{2.2}$$

Then necessarily from the second block of (2.1), $\boldsymbol{x}_* = \boldsymbol{Z}x$ for some x, in which case the first block gives $\boldsymbol{Q}\boldsymbol{Z}x + \boldsymbol{A}^{\mathsf{T}}\boldsymbol{y}_* = \boldsymbol{c}$ and, hence, from (2.2),

$$Qx_* = c, \qquad \text{where} \quad Q := \boldsymbol{Z}^{\mathsf{T}}\boldsymbol{Q}\boldsymbol{Z} \quad \text{and} \quad c := \boldsymbol{Z}^{\mathsf{T}}\boldsymbol{c}. \tag{2.3}$$

Notice that if we define

$$\boldsymbol{r} := \boldsymbol{c} - \boldsymbol{Q}\boldsymbol{x} - \boldsymbol{A}^{\mathsf{T}}\boldsymbol{y} \quad \text{and} \quad r := c - Qx,$$

it follows immediately from (2.2) that residuals are transformed according to

$$r = \boldsymbol{Z}^{\mathsf{T}}\boldsymbol{r} \tag{2.4}$$

regardless of the value of $\boldsymbol{y}$.

Let G be a symmetric positive-definite approximation to Q, and suppose that we may write $G = LL^{\mathsf{T}}$ as necessary. We will consider the central-preconditioned variant

$$L^{-1}QL^{-\mathsf{T}}(L^{\mathsf{T}}x_*) = L^{-1}c \tag{2.5}$$

of (2.3). If we define

$$\boldsymbol{\mathsf{Q}} := L^{-1}QL^{-\mathsf{T}}, \quad \boldsymbol{\mathsf{c}} := L^{-1}c, \quad \text{and} \quad \boldsymbol{\mathsf{x}}_* = L^{-\mathsf{T}}x_*, \tag{2.6}$$

the system (2.5) may be expressed compactly as

$$\boldsymbol{\mathsf{Q}}\boldsymbol{\mathsf{x}}_* = \boldsymbol{\mathsf{c}}.$$

Here, residuals are transformed according to

$$\boldsymbol{\mathsf{r}} = \boldsymbol{\mathsf{c}} - \boldsymbol{\mathsf{Q}}\boldsymbol{\mathsf{x}} = L^{-1}(c - Qx) = L^{-1}r.$$

2.1. Krylov-subspace methods. The kth approximations $\boldsymbol{\mathsf{x}}_k \approx \boldsymbol{\mathsf{x}}_*$ generated by Krylov methods lie in either of the k-dimensional Krylov spaces

$$\mathcal{K}_k := \operatorname{Span}\left\{\boldsymbol{\mathsf{Q}}^i\boldsymbol{\mathsf{c}}\right\}_{i=0}^{k-1} \qquad \text{or} \qquad \mathcal{K}_k^{\mathsf{T}} := \operatorname{Span}\left\{(\boldsymbol{\mathsf{Q}}^{\mathsf{T}})^i\boldsymbol{\mathsf{d}}\right\}_{i=0}^{k-1} \tag{2.7}$$

for specified $\boldsymbol{\mathsf{d}}$ for which $\langle\boldsymbol{\mathsf{c}}, \boldsymbol{\mathsf{d}}\rangle \neq 0$. Let $\boldsymbol{\mathsf{V}}_k$ be a matrix whose columns form a basis of $\mathcal{K}_k$. The choice of $\boldsymbol{x}_k$ is characterized by the basis of $\mathcal{K}_k$ that is chosen and an optimality condition. *(Ritz–)Galerkin* methods seek solutions of the form $\boldsymbol{\mathsf{x}}_k := \boldsymbol{\mathsf{V}}_k\boldsymbol{z}$ and impose that the residual $\boldsymbol{\mathsf{r}}_k := \boldsymbol{\mathsf{c}} - \boldsymbol{\mathsf{Q}}\boldsymbol{\mathsf{x}}_k$ be orthogonal to $\mathcal{K}_k$. *Minimum-residual* methods also seek $\boldsymbol{\mathsf{x}}_k := \boldsymbol{\mathsf{V}}_k\boldsymbol{z}$, where $\boldsymbol{z} \in \arg\min_z \|\boldsymbol{\mathsf{Q}}\boldsymbol{\mathsf{V}}_k z - \boldsymbol{\mathsf{c}}\|$ in some specified norm. *Minimum-error* methods seek approximations $\boldsymbol{\mathsf{x}}_k := \boldsymbol{\mathsf{Q}}^{\mathsf{T}}\boldsymbol{\mathsf{V}}_k\boldsymbol{z}$, where $\boldsymbol{z} \in \arg\min_z \|\boldsymbol{\mathsf{Q}}^{\mathsf{T}}\boldsymbol{\mathsf{V}}_k z - \boldsymbol{\mathsf{x}}_*\|$.

The Arnoldi (1951) process may be used to generate orthogonal basis matrices $\boldsymbol{\mathsf{V}}_k^{\circ}$ and $\boldsymbol{\mathsf{W}}_k^{\circ}$ of $\mathcal{K}_k$ and $\mathcal{K}_k^{\mathsf{T}}$, respectively. When $\boldsymbol{\mathsf{Q}}$ is symmetric, the Arnoldi process simplifies into the symmetric Lanczos (1950) process. An alternative is to use a biorthogonal pair of basis matrices $\boldsymbol{\mathsf{V}}_k$ and $\boldsymbol{\mathsf{W}}_k$ for $\mathcal{K}_k$ and $\mathcal{K}_k^{\mathsf{T}}$. This basis-generation process is known as the Lanczos (1950) biorthogonalization process. It is a riskier alternative and may break down, but when $\boldsymbol{\mathsf{Q}}$ is symmetric, breakdown will not occur if $\boldsymbol{\mathsf{c}} = \boldsymbol{\mathsf{d}}$ (van der Vorst, 2003, section 7.1). A good general overview is provided by Saad (2003).

Each of the possibilities depends entirely on its basis matrix $\boldsymbol{V}_k^\circ$ or $\boldsymbol{V}_k$ and the approximation approach. We concentrate on general Krylov methods for which

$$\boldsymbol{x}_k = \boldsymbol{V}_k^\circ z_k \quad \text{or} \quad \boldsymbol{x}_k = \boldsymbol{V}_k z_k,$$

or, in terms of the unpreconditioned variables,

$$x_k = V_k^\circ z_k \quad \text{or} \quad x_k = V_k z_k,$$

where

$$V_k^\circ = L^{-\mathsf{T}} \boldsymbol{V}_k^\circ \quad \text{and} \quad V_k = L^{-\mathsf{T}} \boldsymbol{V}_k \tag{2.8}$$

and z_k is computed by any means from appropriate data $\mathcal{Z}_k$; we shall formally denote the algorithm used to compute z_k as

$$z_k = \Theta_k(\mathcal{Z}_k), \tag{2.9}$$

as appropriate, for some vector valued function Θ_k of dimension k.

Note that the vectors in V_k° are not orthogonal in the ℓ_2 norm, but will be in the G^{-1} norm. We inherit the notation from the orthogonal matrix $\boldsymbol{V}_k^\circ$ with this understanding. We denote the columns of $\boldsymbol{V}_k^\circ$ by $\boldsymbol{v}_i^\circ$, $i = 0, \dots, k-1$.

2.2. Computing suitable bases.

2.2. Computing suitable bases. The other main ingredient in our development is to recall how the bases we mentioned above are computed. Beginning with standard methods for $\boldsymbol{Q}$ as given by (2.6) we derive the corresponding iteration for the data Q and its preconditioner G. In the next few sections, we focus on the Arnoldi process, but similar principles may be applied to other basis-generation processes. We leave until section 2.3 a detailed discussion on how these computed bases may be applied in the full space to solve (2.1).

For a given orthogonalization method, in order to formulate corresponding algorithm involving unpreconditioned quantities we apply changes of variables suggested by (2.6) and (2.8). All transformations in the next few sections will follow the same principle.

PRINCIPLE 2.1.

1. *Basis vectors transform according to* $v_k^\circ = L^{-\mathsf{T}} \boldsymbol{v}_k^\circ$. *Because of* (2.6) *and* (2.7), *each generated column* $\boldsymbol{v}_k^\circ$ *of* $\boldsymbol{V}_k^\circ$ *in preconditioned space has the form* $\boldsymbol{v}_k^\circ = L^{-1}u$ *for some vector* u. *Thus, we obtain* v_k° *by solving the preconditioning system* $G v_k^\circ = u$ *for* v_k°.
2. *Inner products of the form* $\langle \boldsymbol{v}_k^\circ, \boldsymbol{v}_i^\circ \rangle$ *become* $\langle u, v_i^\circ \rangle$, *again because of* (2.6), *where* u *is the vector defined as in the first part of the principle.*

The Arnoldi process applied to (2.6) ensures that

$$\boldsymbol{Q}\boldsymbol{V}_k = \boldsymbol{V}_{k+1} \boldsymbol{H}_{k+1,k} \tag{2.10}$$

$$= \boldsymbol{V}_k \boldsymbol{H}_k + \boldsymbol{h}_{k+1,k} \boldsymbol{v}_{k+1} e_k^{\mathsf{T}}, \tag{2.11}$$

where $\boldsymbol{H}_k$ is k-by-k upper Hessenberg and $\boldsymbol{H}_{k+1,k}$ is $\boldsymbol{H}_k$ with the extra row $\boldsymbol{h}_{k+1,k} e_k^{\mathsf{T}}$. Applying Principle 2.1 to this we obtain Algorithm 2.1 in which everything is expressed in the reduced (unpreconditioned) space. Note also that there is now no explicit mention of L, but rather that solves with G, using any appropriate mechanism, are required.

Algorithm 2.1. Preconditioned Arnoldi process for V_k° and $H_{k,k-1}$.

```
Require: $Q$, $G = G^\mathsf{T} \succ 0$, $c$ and $x_0$
u = c − Qx_0
$u = c - Qx_0$
Solve $Gv_1^\circ = u$ for $v_1^\circ$                       // Initial Krylov vector
$h_{1,0} = \sqrt{\langle u, v_1^\circ \rangle}$                // Initial preconditioned residual norm
if $h_{1,0} \neq 0$ then
    $v_1^\circ = v_1^\circ / h_{1,0}$
$k = 1$
while $h_{k,k-1} \neq 0$ do
    $u = Qv_k^\circ$ and Solve $Gv_{k+1}^\circ = u$            // Compute next Krylov vector
    for $i = 1, \dots, k$ do                                 // Modified Gram–Schmidt
        $h_{i,k} = \langle u, v_i^\circ \rangle$
        $v_{k+1}^\circ = v_{k+1}^\circ - h_{i,k} v_i^\circ$
    $h_{k+1,k} = \sqrt{\langle u, v_{k+1}^\circ \rangle}$
    if $h_{k+1,k} \neq 0$ then
        $v_{k+1}^\circ = v_{k+1}^\circ / h_{k+1,k}$
    $k = k + 1$
```

Principle 2.1 is directly applicable to other basis-generation processes. For instance, it may be applied verbatim to the Lanczos (1950) biorthogonalization process to obtain a corresponding process formulated in terms of reduced-space quantities and the preconditioner G. It may also be applied to the variant given by Chan, de Pillis, and van der Vorst (1998) and Freund, Gutknecht, and Nachtigal (1993), or to the transpose-free variant (Brezinski and Redivo-Zaglia, 1998; Chan, de Pillis, and van der Vorst, 1998).

2.3. Iteration in the full space. We are now in a position to describe how the Krylov methods we have considered in the nullspace of $\boldsymbol{A}$ may actually be applied in the original (full) space. This is achieved via a systematic principle that may be applied to any basis-generation process. The outcome of the application of the principle was previously known in special cases, such as the case where $\boldsymbol{Q}$ is symmetric and positive definite, e.g., (Gould, Hribar, and Nocedal, 2001). Our description is general in the sense that it applies to any valid Krylov method. Recall that in the full space we have

$$\boldsymbol{x}_* = \boldsymbol{Z} x_*, \quad Q = \boldsymbol{Z}^\mathsf{T} \boldsymbol{Q} \boldsymbol{Z}, \quad \text{and} \quad c = \boldsymbol{Z}^\mathsf{T} \boldsymbol{c},$$

in which case we have

$$\boldsymbol{x}_k = \boldsymbol{Z} x_k = \boldsymbol{Z} V_k^\circ z_k \quad \text{or} \quad \boldsymbol{x}_k = \boldsymbol{Z} x_k = \boldsymbol{Z} V_k z_k.$$

Consider also a preconditioner of the form

$$G := \boldsymbol{Z}^\mathsf{T} \boldsymbol{G} \boldsymbol{Z},$$

where $\boldsymbol{G}$ satisfies the following requirement.

Assumption 2.2. The matrix $\boldsymbol{G}$ is symmetric and positive definite on Null($\boldsymbol{A}$).

Such a requirement on $\boldsymbol{G}$ clearly implies that G is symmetric, positive definite. But while choosing $\boldsymbol{G}$ itself to be positive definite will suffice, this is far from necessary and indeed undesirable if, as is common for constrained optimization, $\boldsymbol{Q}$ is indefinite.

Consider iterates generated according to $\boldsymbol{x}_k = \boldsymbol{Z} x_k = \boldsymbol{Z} V_k^\circ z_k$ by the Arnoldi process. We now apply the following principle to express Algorithm 2.1 in terms of full-space quantities.

PRINCIPLE 2.2.

1. *Basis vectors transform according to* $\boldsymbol{v}_k^\circ = \boldsymbol{Z} v_k^\circ$. *Because of* (2.3), *each assignment of* v_k° *in the reduced space has the form* $(\boldsymbol{Z}^\mathsf{T}\boldsymbol{G}\boldsymbol{Z})^{-1}u$ *for some* u *of the form* $u = \boldsymbol{Z}^\mathsf{T}\boldsymbol{u}$. *Thus, we obtain* $\boldsymbol{v}_k^\circ = \boldsymbol{Z}(\boldsymbol{Z}^\mathsf{T}\boldsymbol{G}\boldsymbol{Z})^{-1}\boldsymbol{Z}^\mathsf{T}\boldsymbol{u}$.
2. *Inner products of the form* $\langle u, v_k^\circ\rangle$ *become* $\langle \boldsymbol{Z}^\mathsf{T}\boldsymbol{u}, (\boldsymbol{Z}^\mathsf{T}\boldsymbol{G}\boldsymbol{Z})^{-1}\boldsymbol{Z}^\mathsf{T}\boldsymbol{u}\rangle = \langle \boldsymbol{u}, \boldsymbol{v}_k^\circ\rangle$.

At first sight, Principle 2.2 appears to suggest that full-space algorithms depend explicitly on $\boldsymbol{Z}$. However our choice of G ensures that the crucial operator

$$\boldsymbol{P}_G := \boldsymbol{Z}(\boldsymbol{Z}^\mathsf{T}\boldsymbol{G}\boldsymbol{Z})^{-1}\boldsymbol{Z}^\mathsf{T} \tag{2.12}$$

has properties akin to those of an orthogonal projector into the nullspace of $\boldsymbol{A}$. More precisely, $\boldsymbol{P}_G\boldsymbol{G}$ is an oblique projector into Null($\boldsymbol{A}$).

We summarize a few immediate properties of $\boldsymbol{P}_G$ in the following result.

THEOREM 2.1. *Let* $\boldsymbol{P}_G$ *be defined as in* (2.12) *where* $\boldsymbol{G}$ *satisfies Assumption* 2.2. *Then*

1. $\boldsymbol{P}_G\boldsymbol{G}\boldsymbol{P}_G = \boldsymbol{P}_G$ *and* $(\boldsymbol{P}_G\boldsymbol{G})^2 = \boldsymbol{P}_G\boldsymbol{G}$,
2. $\boldsymbol{P}_G\boldsymbol{x} = 0$ *for all* $\boldsymbol{x} \in \mathrm{Range}(\boldsymbol{A}^\mathsf{T})$,
3. $\boldsymbol{P}_G\boldsymbol{G}x = x$ *for all* $x \in \mathrm{Null}(\boldsymbol{A})$,
4. $\boldsymbol{Z}^\mathsf{T}\boldsymbol{G}\boldsymbol{P}_G = \boldsymbol{Z}^\mathsf{T}$ *and* $\boldsymbol{Z}^\mathsf{T}\boldsymbol{G}(\boldsymbol{I} - \boldsymbol{P}_G\boldsymbol{G}) = \boldsymbol{0}$,
5. $\boldsymbol{P}_G\boldsymbol{G}\boldsymbol{Z} = \boldsymbol{Z}$.

A consequence of the above is that finding $\boldsymbol{Z}$ is unnecessary as it is easy to show (Gould, Hribar, and Nocedal, 2001) that $\boldsymbol{v}_k^\circ = \boldsymbol{P}_G\boldsymbol{u}$ may instead be computed by solving the *constraint-preconditioned* (Keller, Gould, and Wathen, 2000) saddle-point system

$$\begin{bmatrix}\boldsymbol{G} & \boldsymbol{A}^\mathsf{T}\\ \boldsymbol{A} & \end{bmatrix}\begin{bmatrix}\boldsymbol{v}_k^\circ\\ \boldsymbol{y}_k\end{bmatrix} = \begin{bmatrix}\boldsymbol{u}\\ \boldsymbol{0}\end{bmatrix}. \tag{2.13}$$

For the coefficient matrix $\boldsymbol{K}_G$ of (2.13) to be a constraint preconditioner, all that is needed is that $\boldsymbol{G}$ satisfies Assumption 2.2 or, equivalently, that $\boldsymbol{K}_G$ be nonsingular and have precisely as many negative eigenvalues as $\boldsymbol{A}$ has linearly independent rows. An instructive way to summarize the difference between projected and constraint-preconditioned Krylov methods is that projected methods use the preconditioner

$$\begin{bmatrix}\boldsymbol{I} & \boldsymbol{0}\end{bmatrix}\begin{bmatrix}\boldsymbol{G} & \boldsymbol{A}^\mathsf{T}\\ \boldsymbol{A} & \end{bmatrix}^{-1}\begin{bmatrix}\boldsymbol{I}\\ \boldsymbol{0}\end{bmatrix}.$$

This expression of the projected preconditioner effectively extracts the leading block of the inverse of the constraint preconditioner, which is indeed a projector into Null($\boldsymbol{A}$).

Typically, constraint preconditioners are obtained and used by factorizing—either explicitly Keller, Gould, and Wathen (2000) or implicitly Dollar et al. (2006)—$\boldsymbol{K}_G$; this has the further advantage that the required initial value $\boldsymbol{x}_{\mathrm{F}}$ may be found by solving

$$\begin{bmatrix}\boldsymbol{G} & \boldsymbol{A}^\mathsf{T}\\ \boldsymbol{A} & \end{bmatrix}\begin{bmatrix}\boldsymbol{x}_{\mathrm{F}}\\ \boldsymbol{y}_0\end{bmatrix} = \begin{bmatrix}\boldsymbol{g}\\ \boldsymbol{b}\end{bmatrix} \tag{2.14}$$

for arbitrary $\boldsymbol{g}$—usually, $\boldsymbol{g} = \boldsymbol{0}$ or $\boldsymbol{g} = \boldsymbol{a}$.

Applying Principle 2.2 to Algorithm 2.1, we derive the class of Krylov methods for (2.1) described by Algorithm 2.2 below. We note that although formally we may express methods using Algorithm 2.2, an actual implementation will usually be more

ALGORITHM 2.2. Arnoldi-based projected Krylov methods for (2.1).

```
Require: $\boldsymbol{Q}$, $\boldsymbol{P}_G$, $\boldsymbol{c}$ and $\boldsymbol{x}_0$
$\boldsymbol{u} = \boldsymbol{c} - \boldsymbol{Q}\boldsymbol{x}_0$
Compute $\boldsymbol{v}_1 = \boldsymbol{P}_G\boldsymbol{u}$                         // Initial Krylov vector
$\boldsymbol{h}_{1,0} = \sqrt{\langle \boldsymbol{v}_1, \boldsymbol{u}\rangle}$           // Initial preconditioned residual norm
if $\boldsymbol{h}_{1,0} \neq 0$ then
    $\boldsymbol{v}_1 = \boldsymbol{v}_1/\boldsymbol{h}_{1,0}$
$\boldsymbol{x}_1 = \boldsymbol{V}_1^{\circ}\Theta_1(\boldsymbol{H}_{1,0}, \boldsymbol{h}_{1,0})$            // Initial solution estimate
$k = 1$
while $\boldsymbol{h}_{k,k-1} \neq 0$ do
    $\boldsymbol{u} = \boldsymbol{Q}\boldsymbol{v}_k$
    Compute $\boldsymbol{v}_{k+1} = \boldsymbol{P}_G\boldsymbol{u}$             // Compute next Krylov vector
    for $i = 1, \ldots, k$ do                           // Modified Gram–Schmidt
        $\boldsymbol{h}_{i,k} = \langle \boldsymbol{v}_i, \boldsymbol{u}\rangle$
        $\boldsymbol{v}_{k+1} = \boldsymbol{v}_{k+1} - \boldsymbol{h}_{i,k}\boldsymbol{v}_i$
    $\boldsymbol{h}_{k+1,k} = \sqrt{\langle \boldsymbol{v}_{k+1}, \boldsymbol{u}\rangle}$
    if $\boldsymbol{h}_{k+1,k} \neq 0$ then
        $\boldsymbol{v}_{k+1} = \boldsymbol{v}_{k+1}/\boldsymbol{h}_{k+1,k}$
    $\boldsymbol{x}_{k+1} = \boldsymbol{V}_{k+1}^{\circ}\Theta_{k+1}(\boldsymbol{H}_{k+1,k}, \boldsymbol{h}_{1,0})$    // Update solution estimate
    $k = k + 1$
```

streamlined. In particular, the estimate of the solution may often be expressed more succinctly using quantities computed elsewhere in the algorithm. We give an example of this in section 3.

Observe that Algorithm 2.2 (and those that will follow) only aim to find $\boldsymbol{x}_*$ satisfying (2.1) and not $\boldsymbol{y}_*$. One way to obtain the complete solution is to compute an estimate of the $\boldsymbol{y}_*$ component of the solution only once a good approximation to $\boldsymbol{x}_*$ has been found. To do so, suppose that $\boldsymbol{x}$ has been obtained as an estimate of $\boldsymbol{x}_*$. We may then compute

$$\boldsymbol{y}_* = \underset{\boldsymbol{y}\in\mathbb{R}^m}{\arg\min}\ \|\boldsymbol{A}^{\mathsf{T}}\boldsymbol{y} - (\boldsymbol{c} - \boldsymbol{Q}\boldsymbol{x})\|_N \tag{2.15}$$

for some appropriate norm $\|\cdot\|_N$. If $\boldsymbol{G}$ were positive definite, the norm $\|\cdot\|_{\boldsymbol{G}^{-1}}$ would be suitable, and $\boldsymbol{y}_*$ may be found by solving

$$\begin{bmatrix} \boldsymbol{G} & \boldsymbol{A}^{\mathsf{T}} \\ \boldsymbol{A} & \end{bmatrix} \begin{bmatrix} \boldsymbol{w}_* \\ \boldsymbol{y}_* \end{bmatrix} = \begin{bmatrix} \boldsymbol{c} - \boldsymbol{Q}\boldsymbol{x} \\ \boldsymbol{0} \end{bmatrix}$$

involving the matrix $\boldsymbol{K}_G$ used in (2.13). The same is true using the dual to the seminorm $\sqrt{\langle \boldsymbol{s}, \boldsymbol{G}\boldsymbol{s}\rangle}$ defined on the manifold $\boldsymbol{A}\boldsymbol{s} = \boldsymbol{0}$ under the more general Assumption 2.2 on $\boldsymbol{G}$ (see Conn et al., 2000, section 2.2; and Gould, Hribar, and Nocedal, 2001, section 6).

Principle 2.2 may equally be applied to the Lanczos biorthgonalization process or its transpose-free variant to obtain corresponding full-space statements.

2.4. Constraint-preconditioned variants. In section 2.2 we showed how to form a G-orthogonal basis for the Krylov space generated from Q and c, using the Arnoldi process (Algorithm 2.1). This then lead us to an iteration (Algorithm 2.2) involving $\boldsymbol{Q}$ and the constraint preconditioner $\boldsymbol{K}_G$ in the full space. One might naturally ask whether it is possible to bypass this two-stage approach, and instead to apply the preconditioned processes of section 2.2 directly to (2.1), with a constraint preconditioner of the form (2.13). Note that we temporarily ignore the fact that such a "preconditioner" is indefinite (and thus formally not applicable) and thus flout conventional wisdom; we simply write what the preconditioned processes would be if it were to be employed.

The following result establishes the equivalence between the projected methods derived above and standard preconditioned Krylov methods applied with a constraint preconditioner. Such a result has been known for some time in the case where $\boldsymbol{Q}$ is symmetric and positive definite (see, e.g., Lukšan and Vlček, 1998; Perugia, Simoncini, and Arioli, 1999; Rozložník and Simoncini, 2002). Here we generalize the standard proofs to allow for general matrices.

Theorem 2.2. *Algorithm* 2.1 *applied to* (2.1) *with an initial guess of the form* $[\boldsymbol{x}_0^\mathsf{T}, \boldsymbol{y}_0^\mathsf{T}]^\mathsf{T}$ *satisfying* $\boldsymbol{A}\boldsymbol{x}_0 = \boldsymbol{0}$ *and using the constraint preconditioner* (2.13) *generates Krylov vectors* $v_k = [v_{k,x}^\mathsf{T}, v_{k,y}^\mathsf{T}]^\mathsf{T}$ *such that at each iteration* k*,* $v_{k,x}$ *is the Krylov vector* $\boldsymbol{v}_k$ *generated by Algorithm* 2.2 *with initial guess* $\boldsymbol{x}_0$ *at iteration* k*. The temporary vector* u *has the form* $[(\boldsymbol{u} + \boldsymbol{A}^\mathsf{T}\boldsymbol{w})^\mathsf{T}, \boldsymbol{0}^\mathsf{T}]^\mathsf{T}$ *for some vector* $\boldsymbol{w}$*, where* $\boldsymbol{u}$ *is the corresponding temporary vector generated by Algorithm* 2.2*. In addition, the Hessenberg matrix generated is identical to that generated by Algorithm* 2.2*.*

Proof. Suppose Algorithm 2.1 is initialized with a starting guess of the form $\left[\boldsymbol{x}_0^\mathsf{T}, \boldsymbol{y}_0^\mathsf{T}\right]^\mathsf{T}$ for which $\boldsymbol{x}_0 \in \mathrm{Null}(\boldsymbol{A})$. Line 1 of Algorithm 2.1 reads

$$u = \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{0} \end{bmatrix} - \begin{bmatrix} \boldsymbol{Q} & \boldsymbol{A}^\mathsf{T} \\ \boldsymbol{A} & \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_0 \\ \boldsymbol{y}_0 \end{bmatrix} = \begin{bmatrix} \boldsymbol{c} - \boldsymbol{Q}\boldsymbol{x}_0 + \boldsymbol{A}^\mathsf{T}\boldsymbol{y}_0 \\ \boldsymbol{0} \end{bmatrix}$$

because $\boldsymbol{A}\boldsymbol{x}_0 = \boldsymbol{0}$ so that the initial Krylov vector is given as the solution of

$$\begin{bmatrix} \boldsymbol{G} & \boldsymbol{A}^\mathsf{T} \\ \boldsymbol{A} & \end{bmatrix} \begin{bmatrix} v_{1,x} \\ v_{1,y} \end{bmatrix} = \begin{bmatrix} \boldsymbol{c} + \boldsymbol{Q}\boldsymbol{x}_0 + \boldsymbol{A}^\mathsf{T}\boldsymbol{y}_0 \\ \boldsymbol{0} \end{bmatrix}$$

and this shows that $v_{1,x}$ is identical to the vector $\boldsymbol{v}_1$ given at line 2 of Algorithm 2.2 since the term $\boldsymbol{A}^\mathsf{T}\boldsymbol{y}_0$ does not impact the component $v_{1,x}$ of the solution—see the second property of Theorem 2.1. In particular, $v_{1,x}$ lies in the nullspace of $\boldsymbol{A}$. At line 3 of Algorithm 2.1, we compute

$$h_{1,0} = \sqrt{\langle v_{1,x}, \boldsymbol{c} - \boldsymbol{Q}\boldsymbol{x}_0 + \boldsymbol{A}^\mathsf{T}\boldsymbol{y}_0 \rangle + \langle v_{1,y}, \boldsymbol{0} \rangle} = \sqrt{\langle \boldsymbol{v}_1, \boldsymbol{c} - \boldsymbol{Q}\boldsymbol{x}_0 \rangle},$$

which is identical to $\boldsymbol{h}_{1,0}$ computed at line 3 of Algorithm 2.2.

Assuming now that $v_{k,x}$ lies in the nullspace of $\boldsymbol{A}$ and coincides with $\boldsymbol{v}_k$, we establish by recursion that $v_{k+1,x}$ also lies in the nullspace of $\boldsymbol{A}$ and coincides with $\boldsymbol{v}_{k+1}$.

The vector u computed at line 8 of Algorithm 2.1 is

$$u = \begin{bmatrix} \boldsymbol{Q} & \boldsymbol{A}^\mathsf{T} \\ \boldsymbol{A} & \end{bmatrix} \begin{bmatrix} v_{k,x} \\ v_{k,y} \end{bmatrix} = \begin{bmatrix} \boldsymbol{Q}v_{k,x} + \boldsymbol{A}^\mathsf{T}v_{k,y} \\ \boldsymbol{0} \end{bmatrix}$$

and the first component of u is $\boldsymbol{u} + \boldsymbol{A}^\mathsf{T}v_{k,y}$, where $\boldsymbol{u}$ is computed at line 9 of Algorithm 2.2. Next, v_{k+1} is given as the solution of

$$\begin{bmatrix} \boldsymbol{G} & \boldsymbol{A}^\mathsf{T} \\ \boldsymbol{A} & \end{bmatrix} \begin{bmatrix} v_{k+1,x} \\ v_{k+1,y} \end{bmatrix} = \begin{bmatrix} \boldsymbol{u} + \boldsymbol{A}^\mathsf{T}v_{k,y} \\ \boldsymbol{0} \end{bmatrix}.$$

This shows that $v_{k+1,x} = \boldsymbol{v}_{k+1}$. Moreover, $\boldsymbol{A}v_{k+1,x} = \boldsymbol{0}$.

Finally, Algorithm 2.1 computes

$$h_{i,k} = \langle v_{i,x}, \boldsymbol{Q} v_{k,x} + \boldsymbol{A}^{\mathsf{T}} v_{k,y} \rangle + \langle v_{i,y}, \boldsymbol{0} \rangle = \langle v_{i,x}, \boldsymbol{Q} v_{k,x} \rangle = \langle \boldsymbol{v}_i, \boldsymbol{u} \rangle = \boldsymbol{h}_{i,k},$$

where we used the fact that $\boldsymbol{A} v_{i,x} = 0$ for all $i \leq k$. Similarly, we can show that $h_{k+1,k} = \boldsymbol{h}_{k+1,k}$. □

In exact arithmetic, the advantage of Algorithm 2.2 over Algorithm 2.1 with a constraint preconditioner is in terms of memory requirements. The vectors $\boldsymbol{u}$ and $\boldsymbol{v}_k$ generated have size n while the vectors u and v_k of Algorithm 2.1 have size $n + m$. The advantage of Algorithm 2.1 with constraint preconditioner is that the subvectors $v_{k,y}$ may be used to recur approximations to the $\boldsymbol{y}$ segment of the solution of (2.1). Depending on how the application of $\boldsymbol{P}_G$ is implemented, however, the same approximations may be obtained and recurred in Algorithm 2.2.

The above equivalence applies to all Krylov methods deriving directly from the Arnoldi process, including GMRES and FOM, but also to methods deriving directly from the symmetric Lanczos process, including the conjugate gradient method, MINRES, and SYMMLQ (see Gould, Orban, and Rees, 2013). This illustrates the point that in the special context of the constraint-preconditioned symmetric Lanczos process and with a suitable initial guess, MINRES with the indefinite preconditioner (2.13) is a well-defined Krylov method.

Example 2.1. Projected GMRES Standard GMRES performs m iterations of Lanczos orthogonalization to obtain $\boldsymbol{V}_{m+1}$ and $\boldsymbol{H}_{m+1,m}$, and then computes an approximation $\boldsymbol{x}_{m+1}$ in the $(m+1)$st Krylov subspace $\boldsymbol{x}_{m+1} := \boldsymbol{V}_{m+1} \boldsymbol{z}_{m+1}$ by solving the linear least-squares problem

$$\underset{\boldsymbol{z}_{m+1}}{\text{minimize}} \ \|\boldsymbol{h}_{1,0} e_1 - \boldsymbol{H}_{m+1,m} \boldsymbol{z}_{m+1}\|_2. \tag{2.16}$$

The preconditioned and projected GMRES algorithms are based on Algorithms 2.1 and 2.2, respectively. Besides Algorithm 2.2, the details of the projected GMRES algorithm are standard. The constraint-preconditioned GMRES algorithm defines its $(m+1)$st approximation as

$$\begin{bmatrix} x_{m+1} \\ y_{m+1} \end{bmatrix} = V_{m+1} z_{m+1} = \begin{bmatrix} V_{x,m+1} z_{m+1} \\ V_{y,m+1} z_{m+1} \end{bmatrix}.$$

From Theorem 2.2 and (2.16), it is clear that the $\boldsymbol{x}_{m+1}$ defined by the projected variant coincides with the x_{m+1} defined by the constraint-preconditioned variant.

Results similar in spirit to Theorem 2.2 may be stated to establish a formal equivalence between projected and constraint-preconditioned methods based on the Lanczos biorthogonalization process or its transpose-free variant (see Gould, Orban, and Rees, 2013). In particular, for any Krylov method deriving from the Lanczos biorthogonalization process, the x component of iterates generated by an appropriately initialized constraint-preconditioned variant of this method will coincide with the iterates generated by the projected variant of this method. Among others, this conclusion applies to the two-sided Lanczos method for linear systems, the biconjugate gradient algorithm (Bi-CG), and the quasi-minimum residual method (QMR). Just as for the Arnoldi process, such results have been known for some time in the case where $\boldsymbol{Q}$ is symmetric and positive definite (see, e.g., Lukšan and Vlček, 1998; Perugia, Simoncini, and Arioli, 1999; Rozložník and Simoncini, 2002).

3. Example. Projected MINRES.

In this section, we consider the specific example of the projected MINRES algorithm, in which approximate solutions $x_k = V_k z_k$ are chosen so as to minimize the norm of the residual $r_k := c - Qx_k$ in the norm defined by G^{-1}. Recall that $c = \boldsymbol{Z}^\mathsf{T}\boldsymbol{c}$, $Q = \boldsymbol{Z}^\mathsf{T}\boldsymbol{Q}\boldsymbol{Z}$, and $G = \boldsymbol{Z}^\mathsf{T}\boldsymbol{G}\boldsymbol{Z}$ is positive definite. The quantity minimized by the projected MINRES at each iteration may thus be written

$$\|r_k\|^2_{G^{-1}} = \langle r_k, (\boldsymbol{Z}^\mathsf{T}\boldsymbol{G}\boldsymbol{Z})^{-1} r_k\rangle. \tag{3.1}$$

We now establish directly that standard MINRES applied with a constraint preconditioner also minimizes (3.1), and therefore that MINRES with the indefinite preconditioner (2.13) is a well-defined Krylov method.

Theorem 2.2 guarantees that with appropriate initial conditions, the approximate solution $(\boldsymbol{x}_k, \boldsymbol{y}_k)$ generated at the kth iteration of the constraint-preconditioned MINRES is such that $\boldsymbol{x}_k \in \operatorname{Null}(\boldsymbol{A})$. Consider the residual

$$\boldsymbol{r}_k := \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{0} \end{bmatrix} - \begin{bmatrix} \boldsymbol{Q} & \boldsymbol{A}^\mathsf{T} \\ \boldsymbol{A} & \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{y}_k \end{bmatrix} = \begin{bmatrix} \boldsymbol{c} - \boldsymbol{Q}\boldsymbol{x}_k - \boldsymbol{A}^\mathsf{T}\boldsymbol{y}_k \\ \boldsymbol{0} \end{bmatrix}.$$

It is tempting to claim that it is this residual that is minimized in the norm defined by the inverse of the preconditioner (2.13). Unfortunately, the preconditioner is indefinite. The appropriate interpretation of this claim is to consider the *seminorm* associated with the preconditioner and defined by

$$\|\boldsymbol{r}_k\|^2_{[\boldsymbol{G}]} := \langle \boldsymbol{r}_k, \boldsymbol{s}_k\rangle, \quad \text{where} \quad \begin{bmatrix} \boldsymbol{G} & \boldsymbol{A}^\mathsf{T} \\ \boldsymbol{A} & \end{bmatrix} \begin{bmatrix} \boldsymbol{s}_k \\ \boldsymbol{t}_k \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}_k \\ \boldsymbol{0} \end{bmatrix}, \tag{3.2}$$

i.e.,

$$\|\boldsymbol{r}_k\|^2_{[\boldsymbol{G}]} = \langle \boldsymbol{c} - \boldsymbol{Q}\boldsymbol{x}_k - \boldsymbol{A}^\mathsf{T}\boldsymbol{y}_k, \boldsymbol{s}_k\rangle = \langle \boldsymbol{c} - \boldsymbol{Q}\boldsymbol{x}_k, \boldsymbol{s}_k\rangle. \tag{3.3}$$

Note that $\|\boldsymbol{r}_k\|_{[\boldsymbol{G}]}$ measures deviation of $\boldsymbol{r}_k$ from the range space of $\boldsymbol{A}^\mathsf{T}$ and vanishes if and only if $\boldsymbol{r}_k$ is orthogonal to the nullspace of $\boldsymbol{A}$. In effect, $\|\cdot\|_{[\boldsymbol{G}]}$ defines a norm on the nullspace of $\boldsymbol{A}$. Such seminorms have been used in optimization contexts (Conn et al., 2000). Observe from (3.2) that $\boldsymbol{s}_k \in \operatorname{Null}(\boldsymbol{A})$ and therefore, $\boldsymbol{s}_k = \boldsymbol{Z}s_k$ for some s_k. The first block equation of (3.2) premultiplied by $\boldsymbol{Z}^\mathsf{T}$ then yields

$$s_k = (\boldsymbol{Z}^\mathsf{T}\boldsymbol{G}\boldsymbol{Z})^{-1}\boldsymbol{Z}^\mathsf{T}(\boldsymbol{c} - \boldsymbol{Q}\boldsymbol{x}_k) = (\boldsymbol{Z}^\mathsf{T}\boldsymbol{G}\boldsymbol{Z})^{-1} r_k.$$

Introducing this expression of s_k into (3.3), we finally obtain

$$\|\boldsymbol{r}_k\|^2_{[\boldsymbol{G}]} = \langle \boldsymbol{Z}^\mathsf{T}(\boldsymbol{c} - \boldsymbol{Q}\boldsymbol{x}_k), s_k\rangle = \langle r_k, (\boldsymbol{Z}^\mathsf{T}\boldsymbol{G}\boldsymbol{Z})^{-1} r_k\rangle = \|r_k\|^2_{G^{-1}},$$

which coincides with (3.1). On substituting $r_k = \boldsymbol{Z}^\mathsf{T}\boldsymbol{r}_k$ in this last identity, we also see that

$$\|\boldsymbol{r}_k\|^2_{[\boldsymbol{G}]} = \langle \boldsymbol{r}_k, \boldsymbol{Z}(\boldsymbol{Z}^\mathsf{T}\boldsymbol{G}\boldsymbol{Z})^{-1}\boldsymbol{Z}^\mathsf{T}\boldsymbol{r}_k\rangle = \langle \boldsymbol{r}_k, P_G(\boldsymbol{r}_k)\rangle = \|P_G(\boldsymbol{r}_k)\|^2_2.$$

We conclude that the projected MINRES algorithm minimizes the Euclidean norm of the projected residual, i.e., its seminorm induced by the constraint preconditioner. Since the spaces over which this quantity are minimized are the same in both methods (Rozložník and Simoncini, 2002), projected MINRES and standard MINRES applied with a constraint preconditioner are equivalent in exact arithmetic.

Similar conclusions can be drawn about the projected SYMMLQ algorithm, which is also well defined in the presence of a constraint preconditioner.

As with the projected conjugate gradient method, the numerical stability of both projected MINRES and standard MINRES applied with a constraint preconditioner is dependent on keeping the components $\boldsymbol{x}_k$ in the nullspace of $\boldsymbol{A}$. While this is always true in exact arithmetic, the accumulation of rounding errors can quickly give $\boldsymbol{x}_k$ a nonnegligible component in the range of $\boldsymbol{A}^\mathsf{T}$ which may cause the method to break down.

To help to ameliorate this effect Gould, Hribar, and Nocedal (2001) suggest that the constraint preconditioner be applied exactly, which is achieved by not only solving the preconditioned system with a direct method, but by applying one (or more) steps of iterative refinement to the appproximate solution obtained. This was suggested in the context of projected CG, but the technique is still worthwhile here.

Example 3.1. We take the matrix and preconditioner formed in MATLAB by the commands.

```
n = 100; m = 75;
Q = rand(n,n);
Q = Q + Q’ + 5*eye(n);
A = rand(m,n);
K = [Q A’; A zeros(m,m)];
G = diag(abs(diag(Q)));
```

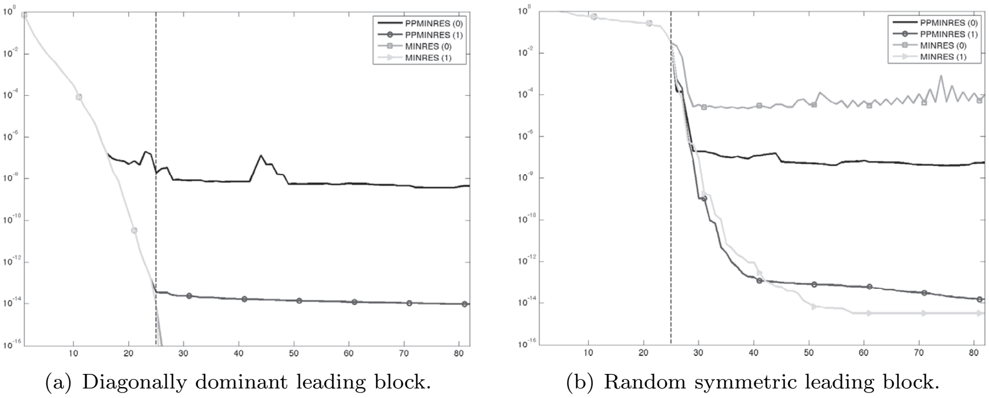
That is, we consider a random saddle-point matrix with a diagonally dominant leading block and take the constraint preconditioner, where $\boldsymbol{G}$ is the positive diagonal matrix whose nonzero entries coincide with those of $\boldsymbol{Q}$.

Example 3.1 is chosen so that the approximate leading block $\boldsymbol{G}$ is a good approximation of the actual $\boldsymbol{Q}$. In the next example this is not the case.

Example 3.2. Here we modify the matrix from Example 3.1 by setting `Q = Q - 5 * eye(5)`, and taking the equivalent $(1, 1)$ block in the constraint preconditioner.

We apply projected MINRES (denoted PPMINRES) and standard MINRES with a constraint preconditioner to the linear systems described in Examples 3.1 and 3.2. The algorithms are applied both with and without a step of iterative refinement. The results are reported as convergence curves in Figure 1.

From Figure 1, we see that—as was the case with projected CG (Gould, Hribar, and Nocedal, 2001)—it is advisable to apply these methods with iterative refinement.



(a) Diagonally dominant leading block.

(b) Random symmetric leading block.

FIG. 1. *Convergence curves for MINRES and PPMINRES. The number in brackets denotes the number of steps of iterative refinement used. The vertical dashed line shows where the algorithm would converge in exact arithmetic.*

Our experiments suggest that a single step of iterative refinement is sufficient. If the $(1,1)$ block of the preconditioner is a good approximation to $\boldsymbol{Q}$, then it seems that standard MINRES alone does very well. However, when the approximation is not so good, as in Example 3.2, this becomes the worst-performing method in terms of maximum attainable accuracy.

We note that Fong and Saunders (2012) argue that it is often advantageous to use MINRES over CG even for symmetric positive definite systems of equations. Since we showed that MINRES applied with a constraint preconditioner is a well-defined method, there may be advantages to using projected MINRES in the place of projected CG even for cases where $\boldsymbol{Q}$ is symmetric positive definite.

4. Discussion.

We have considered the families of processes that form the basis of the great majority of Krylov methods. There are other types of processes that also possess projected variants although they do not strictly qualify as Krylov methods. An example is the tridiagonalization process described by Saunders, Simon, and Yip (1988) on which the iterative methods USYMLQ and USYMQR are based. Initialized with $v_0 = w_0 = 0$ and arbitrary unit vectors v_1 and w_1, this process generates sequences $\{v_k\}$ and $\{w_k\}$ according to

$$t_{k+1,k} v_{k+1} = Q w_k - t_{k,k} v_k - t_{k-1,k} v_{k-1}, \tag{4.1a}$$

$$t_{k,k+1} w_{k+1} = Q^{\mathsf{T}} v_k - t_{k,k} w_k - t_{k,k-1} w_{k-1}, \tag{4.1b}$$

where $t_{k,k} := \langle v_k, Q w_k \rangle$ and the off-diagonal elements $t_{k+1,k}$ and $t_{k,k+1}$ are chosen to normalize v_{k+1} and w_{k+1}. On rearranging the above, the process is characterized by the identities

$$\begin{aligned} Q W_k &= V_k T_k + t_{k+1,k} v_{k+1} e_k^{\mathsf{T}}, \\ Q^{\mathsf{T}} V_k &= W_k T_k + t_{k,k+1} w_{k+1} e_k^{\mathsf{T}}, \end{aligned}$$

where T_k is a k-by-k tridiagonal matrix with positive off-diagonal elements. It is possible to show that both sequences $\{v_k\}$ and $\{w_k\}$ are orthonormal and they are mutually conjugate in the sense that $\langle v_j, Q w_k \rangle = \langle w_j, Q v_k \rangle = 0$ for $k < j - 1$ (Saunders, Simon, and Yip, 1988, Theorem 1). The above relations differ considerably from those arising from biorthogonalization processes but are reminiscent of the Golub and Kahan (1965) bidiagonalization process. Though the above process does not generate Krylov spaces but somewhat larger spaces, it is possible to derive a projected variant applicable directly to (2.1) in the same way as in section 2.3.

Orban (2014) applies Principles 2.1 and 2.2 to the Golub and Kahan (1965) process to derive projected variants of iterative methods for linear least-squares problems such as LSQR (Paige and Saunders, 1982) and LSMR (Fong and Saunders, 2011). The resulting methods are suited to linear least-squares problems with linear equality constraints and are intimately related to full-space methods such as MINRES applied to the symmetric system representing the optimality conditions of such problems.

Certain saddle-point systems, such as those arising from the discretization of stabilized Navier–Stokes flow problems, have the form

$$\begin{bmatrix} \boldsymbol{Q} & \boldsymbol{A}^{\mathsf{T}} \\ \boldsymbol{A} & -\boldsymbol{C} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_* \\ \boldsymbol{y}_* \end{bmatrix} = \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{b} \end{bmatrix},$$

where $\boldsymbol{C}$ is typically symmetric and positive semidefinite (Elman, Silvester, and Wathen, 2005). Assuming more generally that $\boldsymbol{C}$ may be decomposed as $\boldsymbol{E}\boldsymbol{D}\boldsymbol{E}^{\mathsf{T}}$, where

$\boldsymbol{D}$ is nonsingular, and introducing $\boldsymbol{w} := -\boldsymbol{D}\boldsymbol{E}^\top \boldsymbol{y}$ as suggested by Dollar et al. (2006), such systems may be equivalently reformulated as

$$\begin{bmatrix} \boldsymbol{Q} & & \boldsymbol{A}^\top \\ & \boldsymbol{D}^{-1} & \boldsymbol{E}^\top \\ \boldsymbol{A} & \boldsymbol{E} & \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_* \\ \boldsymbol{w}_* \\ \boldsymbol{y}_* \end{bmatrix} = \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{0} \\ \boldsymbol{b} \end{bmatrix}.$$

The latter system has the form (1.1) and the methods proposed in this paper are applicable.

In the case where $\boldsymbol{Q}$ is symmetric positive definite on the nullspace of $\boldsymbol{A}$, $\langle \cdot, \cdot \rangle_{\boldsymbol{K}_Q}$ defines a semidefinite bilinear form on the space $\mathcal{L} := \{ [\begin{smallmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{smallmatrix}] \text{ s.t. } \boldsymbol{x} \in \ker(\boldsymbol{A})\}$, and hence induces a seminorm on $\mathcal{L}$. If we restrict $\mathcal{L}$ to the subset where $\boldsymbol{y} = \boldsymbol{0}$, then the bilinear form above becomes an inner product. The matrix

$$\mathcal{A} := \begin{bmatrix} \boldsymbol{G} & \boldsymbol{A}^\top \\ \boldsymbol{A} & \boldsymbol{0} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{Q} & \boldsymbol{A}^\top \\ \boldsymbol{A} & \boldsymbol{0} \end{bmatrix}$$

is trivially self-adjoint in this bilinear form, and since the space $\mathcal{L}$ is invariant under multiplication with $\mathcal{A}$, projected conjugate gradients can be thought of as standard conjugate gradients, with a suitable starting vector in $\mathcal{L}$, in this nonstandard (indefinite) inner product. The extensions to other Krylov methods, as outlined in the preceding sections, can be interpreted similarly where appropriate.

Finally, we note that it appears as if $\boldsymbol{A}$ must be available as an explicit matrix to factorize (2.13). There are other ways to compute projections. One of them is to solve a linear least-squares problem of the form (2.15) using an appropriate metric, which only requires $\boldsymbol{A}$ to be available as an operator. It remains important, however, that such projections be computed accurately and this can mean that an iterative solver must be supplied with stringent stopping conditions.

5. Conclusion.

We have demonstrated systematic principles to show how any Krylov subspace method may be applied as a projected method, which acts purely in the nullspace. In particular—contrary to popular belief—methods for symmetric but indefinite matrices, such as MINRES, my be applied as projected methods.

Furthermore, we have shown that any Krylov subspace method applied with a constraint preconditioner is numerically equivalent to a projected Krylov method. A comparison of the relative merits of the different methods when applied with a constraint preconditioner would be interesting to perform, but is beyond the scope of the current work. Although the projected and standard methods are equivalent in exact arithmetic they have been shown to behave differently in the presence of rounding errors; a detailed investigation of this phenomenon would be another interesting avenue for further work.

REFERENCES

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

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

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

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

T. F. Coleman (1994), *Linearly constrained optimization and projected preconditioned conjugate gradients*, in Proceedings of the Fifth SIAM Conference on Applied Linear Algebra, J. G. Lewis, ed., SIAM, Philadelphia, pp. 118–122.

A. R. Conn, N. I. M. Gould, D. Orban, and P. Toint (2000), *A primal-dual trust-region algorithm for non-convex nonlinear programming*, Math. Program. B, 87, pp. 215–249.

H. S. Dollar and A. J. Wathen (2006), *Approximate factorization constraint preconditioners for saddle-point matrices*, SIAM J. Sci. Comput., 27, pp. 1555–1572.

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

H. C. Elman, D. J. Silvester, and A. J. Wathen (2005), *Finite-Elements and Fast Iterative Solvers: With Applications in Incompressible Fluid Dynamics*, Oxford University Press, Oxford.

D. C.-L. Fong and M. Saunders (2011), *LSMR: An iterative algorithm for sparse least-squares problems*, SIAM J. Sci. Comput., 33, pp. 2950–2971.

D. C.-L. Fong and M. A. Saunders, *CG versus MINRES: An empirical comparison*, SQU J. Sci., 17 (2012), pp. 44–62.

R. Freund, M. Gutknecht, and N. Nachtigal (1993), *An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices*, SIAM J. Sci. Comput., 14, pp. 137–158.

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

N. I. M. Gould (1985), *On practical conditions for the existence and uniqueness of solutions to the general equality quadratic-programming problem*, Math. Program., 32, pp. 90–99.

N. I. M. Gould, M. E. Hribar, and J. Nocedal (2001), *On the solution of equality constrained quadratic problems arising in optimization*, SIAM J. Sci. Comput., 23, pp. 1376–1395.

N. I. M. Gould, D. Orban, and T. Rees (2013), *Projected Krylov Methods for Saddle-Point Systems*, Technical report RAL-P-2013-006, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England.

A. Greenbaum (1997), *Iterative Methods for Solving Linear Systems*, Frontiers Appl. Math. 17, SIAM, Philadelphia.

C. Keller, N. I. M. Gould, and A. J. Wathen (2000), *Constraint preconditioning for indefinite linear systems*, SIAM J. Matrix Anal. Appl., 21, pp. 1300–1317.

C. Lanczos (1950), *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, J. Res. Nat. Bur. Stand. B, 45, pp. 225–280.

L. Lukšan and J. Vlček (1998), *Indefinitely preconditioned inexact Newton method for large sparse equality constrained nonlinear programming problems*, Numer. Linear Algebra Appl., 5, pp. 219–247.

D. Orban (2008), *Projected Krylov Methods for Unsymmetric Augmented Systems*, GERAD Technical report G-2008-46, Mathematics and Industrial Engineering Department, Ecole Polytechnique de Montreal, Montreal, Canada.

D. Orban (2014), *The Projected Golub-Kahan Process for Constrained Linear Least-Squares Problems*, Technial reort Cahier du GERAD G-2014-15, GERAD, Montréal, Canada.

C. C. Paige and M. A. Saunders (1975), *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12, pp. 617–629.

C. C. Paige and M. A. Saunders (1982), *LSQR: An algorithm for sparse linear equations and sparse least squares*, ACM Trans. Math. Software, 8, pp. 43–71.

I. Perugia and V. Simoncini (2000), *Block-diagonal and indefinite symmetric preconditioners for mixed finite element formulations*, Numer. Linear Algebra Appl., 7, pp. 585–616.

I. Perugia, V. Simoncini, and M. Arioli (1999), *Linear algebra methods in a mixed approximation of magnetostatic problems*, SIAM J. Sci. Comput., 21, pp. 1085–1101.

B. T. Polyak (1969), *The conjugate gradient method in extremal problems*, U.S.S.R. Comput. Math. Math. Phys., 9, pp. 94–112.

M. Rozložník and V. Simoncini (2002), *Krylov subspace methods for saddle point problems with indefinite preconditioning*, SIAM J. Matrix Anal. Appl., 24, pp. 368–391.

Y. Saad (2003), *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia.

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

H. A. van der Vorst (2003), *Iterative Krylov Methods for Large Linear Systems*, Cambridge University Press, Cambridge.