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. In the symmetric case, their convergence is thus entirely described by the spectrum of the (preconditioned) operator restricted to the nullspace. We provide systematic principles for obtaining the projected form of any well-defined Krylov method. Equivalence properties between projected Krylov methods and standard Krylov methods applied to a saddle-point operator with a constraint preconditioner allow us to show that, contrary to common belief, 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.

1. Introduction. We consider the solution of the saddle-point problem

\[
\begin{bmatrix}
Q & A^T \\
A & 0
\end{bmatrix}
\begin{bmatrix}
x_\ast + x_F \\
y_\ast
\end{bmatrix} =
\begin{bmatrix}
a \\
b
\end{bmatrix}
\tag{1.1}
\]

for some fixed \(x_F\), where all data is real and \(Q\) may be unsymmetric. Such saddle-point systems arise throughout computational science (Benzi et al., 2005). Typical applications arise in conservative fluid flow, structural engineering and constrained optimization. The problem scale often precludes a direct factorization of the matrix in (1.1). Certain applications give rise to a symmetric saddle-point system, including optimization and the study of laminar fluid flow. A typical 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 et al., 2005).

In this paper we propose a family of iterative methods working implicitly in the nullspace of \(A\), requiring only operator-vector products with \(Q\), and possibly its transpose, as well as one or more projections of a vector into the nullspace of \(A\) per iteration. For this we only require that \(Q\) be available as an operator but it must be possible to compute projections into the nullspace of \(A\) accurately.

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

Our approach involves a sequence of key steps. We first reduce the saddle point problem to an equivalent one in the null space of \(A\) 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 un-preconditioned reduced space, and finally the iteration is moved back from the null space via a constraint preconditioner into its original full-space setting. Our framework is closely related to the nullspace method—see (Benzi et al., 2005, Section 6) and references therein—but differs from it in that we only use the data of (1.1). In particular, we do not require that a basis for \( \text{Null}(A) \) be computed.

An interesting consequence of our framework is to establish that well-known methods such as MINRES and SYMMLQ applied to (1.1) are well defined in the presence of an indefinite preconditioner. This is at variance with the commonly-issued warnings (e.g., Elman et al., 2005, p.287 Greenbaum, 1997, p.121, and van der Vorst, 2003, p.85) that those methods 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 examine the spectral implications of our preconditioners in Section 4, consider other variants in Section 5, and conclude in Section 6.

1.1. Related Research. In optimization contexts, where \( Q \) is symmetric, (1.1) may be interpreted as the first-order optimality conditions of the quadratic program

\[
\begin{align*}
\text{minimize} \quad -a^T(x + x_p) + \frac{1}{2}(x + x_p)^TQ(x + x_p) \\
\text{subject to} \quad A(x + x_p) = b,
\end{align*}
\]

where \( y \) 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 \( Q \) is positive definite on the nullspace of \( A \). Based on this, Gould et al. (2001) devised the projected conjugate gradient method, a variant of the standard conjugate gradient algorithm applied to \( Q \) and restricted to exploration of the nullspace of \( A \)—see also Polyak (1969), Coleman (1994), Lukšan and Vlček (1998) and Perugia and Simoncini (2000). Benzi et al. (2005, Section 6) describe and provide numerous references on the nullspace method, which requires the computation of a basis for \( \text{Null}(A) \). 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, it is possible to avoid computing a basis for the nullspace of \( A \) altogether and formulate the entire algorithm in terms of projections into this nullspace and operator-vector products with \( Q \). One way to compute projections efficiently when \( A \) is available as an explicit matrix is to perform a one-time factorization of a symmetric indefinite matrix of the form (1.1) where \( Q \) 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 et al. (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 $Q$, $A$, $x$, 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 $A$, we refer to corresponding quantities as reduced-space quantities and typeset them in italicized Roman lightface font, e.g., $Q$, $A$, $x$. We precondition reduced-space quantities and refer to them as preconditioned-space quantities. They are typeset in italicized bold sans-serif font, e.g., $Q$, $A$, $x$. The Euclidian 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 et al. (2005, Theorem 3.4) provide necessary and sufficient conditions for Assumption 2.1 to hold. In particular, if $H = \frac{1}{2}(Q + Q^T)$ denotes the symmetric part of $Q$, and if we assume that $H$ is positive semi-definite and $A$ has full row rank, then

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

but the reverse implications do not hold in general. The above conditions are sufficiently general to encompass numerous applications of interest, including optimization and the solution of the discretized Navier-Stokes equations. We refer the interested reader to (Benzi et al., 2005) for more details. At least in theory, it is possible to ensure that $A$ 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 §5, we explain how the methods presented in the next sections apply to such a regularized system.

Suppose that $x_F$ satisfies $Ax_F = b$; as we will see, finding such a value is easy in the framework we develop. Substituting into (1.1), $x_*$ and $y_*$ satisfy

\[
\begin{bmatrix}
Q & A^T \\
A & 0
\end{bmatrix}
\begin{bmatrix}
x_* \\
y_*
\end{bmatrix}
= \begin{bmatrix}
c \\
0
\end{bmatrix}
\]

(2.1)

where $c := a - Qx_F$. Our aim is thus to solve (2.1).

Let $Z$ be any full-rank matrix whose columns span the null-space of $A$, i.e., such that

\[AZ = 0.\]

(2.2)

Then necessarily from the second block of (2.1), $x_* = Zx$ for some $x$, in which case the first block gives $QZx_* + A^Ty_* = c$ and hence from (2.2),

\[Qx_* = c \quad \text{where} \quad Q := Z^TQZ \quad \text{and} \quad c := Z^Tc.\]

(2.3)

Notice that if we define

\[r := c - Qx - A^Ty \quad \text{and} \quad r := c - Qx,\]

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

\[r = Z^T r.\]

(2.4)

regardless of the value of $y$.

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

\[L^{-1}QL^{-T}(L^T x_*) = L^{-1}c\]

(2.5)
of (2.3). If we define

\[ Q := L^{-1}QL^{-T}, \quad c := L^{-1}c \quad \text{and} \quad x_\ast = L^{-T}x_\ast, \]

the system (2.5) may be expressed compactly as

\[ Qx_\ast = c. \]

Here, residuals are transformed according to

\[ r = c - Qx = L^{-1}(c - Qx) = L^{-1}r. \]

**2.1. Krylov-Subspace Methods.** Let \( S_L \) and \( S_R \) be given subspaces of equal dimension, say \( k \), and let the columns of \( S_L \) and \( S_R \) be bases for these spaces. We say that a subspace approximation \( x \) to the solution \( x_\ast \) of the generic linear system \( Qx_\ast = c \) is a Petrov-Galerkin approximation (Bruaset, 1995, Saad, 2003, Ch. 6 & 7, van der Vorst, 2003, Ch. 4) if

\[ x = S_Rz \quad \text{where} \quad S_L^TQ_Sz = S_L^Tc, \]

and we see that such methods find \( x \in S_R \) such that

\[ r = c - Qx \perp S_L. \]

Within this broad framework there are a number of choices that have proven to be successful in the development of iterative methods. First, consider some space \( S \) of dimension \( k \), and an associated matrix \( S \) whose columns form a basis of \( S \). If we set \( S_R = S_L = S \), then we get what is known as a (Ritz-)Galerkin method.

An alternative is to set \( S_L := QS \), \( S_R := S \), say, which gives a minimum-residual approximation. Here

\[ S_L^TQ^TQ_Sz = S_L^TQ^Tc, \]

or equivalently

\[ x = Sz \quad \text{where} \quad z = \arg \min_z \|QSz - c\|. \]

Finally, if we set \( S_L = Q^TQ \) and \( S_R = S \), then the equation we wish to satisfy becomes

\[ S^TQQTz = S^Tc, \]

and hence we get a minimum-error approximation, where

\[ x = Q^TQz \quad \text{where} \quad z = \arg \min_z \|Q^Tz - x_\ast\|. \]

Note that in all of these special examples of Petrov-Galerkin methods we choose the spaces such that \( S_L = BS_R \) for some matrix \( B \). Such processes are termed balanced projection methods (Bruaset, 1995, §3.1).

Given a trial space \( S_R \), we can now apply one of the recipes above to construct a test space \( S_L \) and derive an appropriate iterative method; we simply need to identify a
suitable trial space. In this context it has proved useful to consider the pair of Krylov spaces of dimension $k$,

$$\mathcal{K}_k = \text{Span}\left\{ Q^i c \right\}_{i=0}^{k-1} \quad \text{and} \quad \mathcal{K}_k^T = \text{Span}\left\{ (Q^T)^i d \right\}_{i=0}^{k-1},$$

for specified $d$ for which $\langle c, d \rangle \neq 0$. We focus on the aforementioned approximations to $x_k$ as $k$ increases.

For reasons of numerical stability we prefer orthogonal bases for $\mathcal{K}_k$ and $\mathcal{K}_k^T$ if possible, and may generate them by the Arnoldi (1951) process. In particular, let $V_k^o$ and $W_k^o$ be orthogonal basis matrices of $\mathcal{K}$ and $\mathcal{K}^T$ respectively. Then the corresponding columns $v_i^o$ and $w_i^o$ satisfy

$$QV_k^o = V_{k+1}^o H_{k+1,k} \quad \text{and} \quad Q^T W_k^o = W_{k+1}^o R_{k+1,k},$$

(2.7)

where the $k+1$ by $k$ matrices $H_{k+1,k}$ and $R_{k+1,k}$ are upper Hessenberg, $v_1^o = c/\|c\|$ and $w_1^o = d/\|d\|$.

An alternative is to use a bi-orthogonal pair of basis matrices $V_k$ and $W_k$ for the Krylov spaces $\mathcal{K}_k$ and $\mathcal{K}_k^T$ for which, in exact arithmetic,

$$QV_k = V_{k+1} T_{k+1,k}, \quad W_k^T V_k = D_k, \quad \text{and} \quad W_k^T QV_k = D_k T_{k,k},$$

(2.8)

the leading $k$-by-$k$ portion $T_{k,k}$ of the $k+1$-by-$k$ matrix $T_{k+1,k}$ is tridiagonal, and $D_k$ is diagonal. This alternative is riskier and may break down, but when $Q$ is symmetric, breakdown will not occur if $c = d$ and then $V_k = W_k$, $D_k = I$ and $T_{k,k}$ is symmetric (van der Vorst, 2003, §7.1).

With the Galerkin approximation approach, we may choose $S_k = \mathcal{K}_k$ and use (2.7) in which case

$$x_k = V_k^o z_k \quad \text{where} \quad H_{k,k} z_k = \beta_1 e_1,$$

(2.9)

$H_{k,k}$ is the leading $k$-by-$k$ portion of the upper Hessenberg $H_{k+1,k}$ and $\beta_1 = \|c\|$; this is the FOM method (Saad, 1981, Algorithm 3.2).

Equally, we may instead use a Petrov-Galerkin approach with $S_k = \mathcal{K}_k^T$ and $S_h = \mathcal{K}_k$ and use (2.8), which gives

$$x_k = V_k z_k \quad \text{where} \quad T_{k,k} z_k = \beta_1 e_1$$

(2.10)

and is the basic Bi-CG method (Fletcher, 1976) in the unsymmetric case and equivalent to CG (Hestenes and Stiefel, 1952) when $Q$ is symmetric. FOM requires that all of $V_k$ be stored, while Bi-CG/CG only require recent $v_k$—a so-called short-term recurrence—since the solution to (2.10) may be updated rather than recomputed when $k$ increases because $T_{k,k}$ is tridiagonal. Although methods based on bi-orthogonalization might seem at first sight to require products with $Q$ and its transpose, this is not necessarily the case. In particular both the basis matrix $V_k$ and tridiagonal $T_{k,k}$ may be found by interlaced pairs of products with $Q$, and this leads to the CGS method (Sonneveld, 1989).

For the minimum-residual approximation approach, choosing $S_k = \mathcal{K}_k$ and using (2.7) gives

$$x_k = V_k^o z_k \quad \text{where} \quad H_{k+1,k}^T H_{k+1,k} z_k = \beta_1 H_{k+1,k}^T e_1,$$

(2.11)
which gives the GMRES method (Saad and Schultz, 1986). This simplifies when $Q$ is symmetric since then $H_{k+1,k}$ is tridiagonal and then the resulting MINRES method of Paige and Saunders (1975) again only requires recent $v_i$. Alternatively, one can simply replace $V_k$ by the non-orthogonal $V_k$ from (2.8) and compute

$$x_k = V_k z_k \quad \text{where} \quad T_{k+1,k}^T h_{k+1,k} z_k = \beta_1 T_{k+1,k}^T e_1;$$

(2.12)

this QMR method (Freund and Nachtigal, 1991) minimizes the so-called quasi-residual rather than the residual.

Finally, for the minimum-error approximation, the choice

$$S_k = K_k^T$$

and using (2.7) gives

$$x_k = W_k z_k \quad \text{where} \quad z_k = R_{k+1,k} p_k \quad \text{and} \quad R_{k+1,k}^T R_{k+1,k} p_k = \beta_1 e_1;$$

interesting methods based on this include SYMMLQ (Paige and Saunders, 1975), SYMMBK (Chandra, 1978) and CG when $Q$ is symmetric in which case

$$x_k = V_k z_k \quad \text{where} \quad z_k = T_{k+1,k} p_k \quad \text{and} \quad T_{k+1,k}^T T_{k+1,k} p_k = \beta_1 e_1;$$

(2.13)

Notice that each of the possibilities (2.9)–(2.13) depends entirely on its basis matrix $V_k$ or $V_k$, the corresponding Hessenberg or tridiagonal matrix $H_{k+1,k}$ or $T_{k+1,k}$, and the norm of the initial right-hand side, $\beta_1$. Thus we shall concentrate on general Krylov methods for which

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

or, in terms of the un-preconditioned variables,

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

where

$$V_k^O = L^{-T} V_k \quad \text{and} \quad V_k = L^{-T} V_k$$

(2.14)

and $z_k$ is computed by any means from appropriate data $H_{k+1,k}$, $T_{k+1,k}$ and $\beta_1$; we shall formally denote the algorithm used to compute $z_k$ as

$$z_k = \Theta_k(T_{k+1,k}, \beta_1) \quad \text{or} \quad z_k = \Theta_k(H_{k+1,k}, \beta_1),$$

(2.15)

as appropriate, for some vector valued function $\Theta_k$ of dimension $k$.

2.2. Computing Suitable Bases. The other main ingredient in our development is to recall how the bases we mentioned above are computed. We start by considering methods for $Q$ as given by (2.6). We then derive the corresponding iteration for the data $Q$ and its preconditioner $G$. In the next few sections, we state processes in a format suitable for direct implementation. We leave until §2.3 a detailed discussion on how these computed bases may be applied in the full space to solve (2.1).

2.2.1. Orthogonal Bases and the Arnoldi Process. The Arnoldi process to compute $V_k$ and $H_{k,k-1}$ for $Q$ and $c$ from (2.6) may be summarised as Algorithm 2.1.

After each pass through the while loop, Algorithm 2.1 ensures that

$$Q V_k = V_{k+1} H_{k+1,k}$$

$$= V_k H_k + h_{k+1,k} v_{k+1}^T,$$

(2.16)

(2.17)
Algorithm 2.1 Arnoldi Process for $V^o_k$ and $H_{k,k-1}$

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q$, $c$ and $x_0$</td>
<td>Initial Krylov vector</td>
</tr>
<tr>
<td>Set $v_1^o = c - Qx_0$</td>
<td>Compute next Krylov vector</td>
</tr>
<tr>
<td>$h_{1,0} = \sqrt{\langle v_1^o, v_1^o \rangle}$</td>
<td>Modified Gram-Schmidt</td>
</tr>
<tr>
<td>if $h_{1,0} \neq 0$ then</td>
<td>Residual norm</td>
</tr>
<tr>
<td>$v_1^o = v_1^o / h_{1,0}$</td>
<td></td>
</tr>
<tr>
<td>$k = 1$</td>
<td></td>
</tr>
<tr>
<td>while $h_{k,k-1} \neq 0$ do</td>
<td></td>
</tr>
<tr>
<td>$v_{k+1}^o = Qv_k^o$</td>
<td></td>
</tr>
<tr>
<td>for $i = 1, \ldots, k$ do</td>
<td></td>
</tr>
<tr>
<td>$h_{i,k} = \langle v_i^o, v_{k+1}^o \rangle$</td>
<td></td>
</tr>
<tr>
<td>$v_{k+1}^o = v_{k+1}^o - h_{i,k}v_i^o$</td>
<td></td>
</tr>
<tr>
<td>$h_{k+1,k} = \sqrt{\langle v_{k+1}^o, v_{k+1}^o \rangle}$</td>
<td></td>
</tr>
<tr>
<td>if $h_{k+1,k} \neq 0$ then</td>
<td></td>
</tr>
<tr>
<td>$v_{k+1}^o = v_{k+1}^o / h_{k+1,k}$</td>
<td></td>
</tr>
<tr>
<td>$k = k + 1$</td>
<td></td>
</tr>
</tbody>
</table>

where $H_k$ is $k$-by-$k$ upper Hessenberg and $H_{k+1,k}$ is $H_k$ with the extra row $h_{k+1,k}^T$.

In order to formulate corresponding algorithm involving un-preconditioned quantities, we apply changes of variables suggested by (2.6) and (2.14). All transformations in the next few sections will follow the same principle.

**Principle 2.1.**

1. Basis vectors transform according to $v_k^o = L^{-T} v_k^o$. Because of (2.6), each assignment of $v_k^o$ in preconditioned space has the form $v_k^o = L^{-1} u$ for some vector $u$. Thus, we obtain $v_k^o$ by solving the preconditioning system $Gv_k^o = u$ for $v_k^o$.

2. Inner products of the form $\langle v_k^o, v_i^o \rangle$ become $\langle u, v_i^o \rangle$, again because of (2.6), where $u$ is the vector defined as in the first principle.

Applying the above principles to Algorithm 2.1, we obtain Algorithm 2.2 in which everything is expressed in reduced (un-preconditioned) space.

When $Q$ is symmetric the Arnoldi process simplifies to the symmetric Lanczos process. This variant and its reduced-space formulation is described in Appendix A.

### 2.2.2. Bi-Orthogonal Bases and Lanczos Bi-Orthogonalization

In the interest of space, processes in preconditioned space, i.e., applied directly to $Q$ and $c$, are stated in Appendix A. Henceforth, we only state the result of applying Principle 2.1 to those processes.

The Lanczos (1950) bi-orthogonalization process computes $V_k$, $W_k$ and the corresponding $T_{k,k}$ as described in (2.8). When applied to $Q$ and $c$, we obtain Algorithm A.3, which is a special case of Saad (2003, Algorithm 7.1) and Golub and van Loan (1996, (9.4.7)).
Algorithm 2.2 Preconditioned Arnoldi Process for $V_k^o$ and $H_{k,k-1}$

Require: $Q$, $G = G^T > 0$, $c$ and $x_0$
1: $u = c - Qx_0$
2: Solve $Gu_1^o = u$ for $v_1^o$ // Initial Krylov vector
3: $h_{1,0} = \sqrt{\langle u, v_1^o \rangle}$ // Initial preconditioned residual norm
4: if $h_{1,0} \neq 0$ then
5: $v_1^o = v_1^o / h_{1,0}$
6: $k = 1$
7: while $h_{k,k-1} \neq 0$ do
8: $u = Qv_k^o$ and Solve $Gu_k^o = u$ // Compute next Krylov vector
9: for $i = 1, \ldots, k$ do
10: $h_{i,k} = \langle u, v_i^o \rangle$
11: $v_{k+1}^o = v_{k+1}^o - h_{i,k} v_i^o$
12: $h_{k+1,k} = \sqrt{\langle u, v_{k+1}^o \rangle}$ // Preconditioned residual norm
13: if $h_{k+1,k} \neq 0$ then
14: $v_{k+1}^o = v_{k+1}^o / h_{k+1,k}$
15: $k = k + 1$

Upon defining the (unsymmetric) tridiagonal matrix

$$T_k := \begin{bmatrix} t_{1,1} & t_{1,2} & t_{2,3} \\ 1 & t_{2,2} & t_{3,3} \\ & 1 & \ddots \\ & & \ddots & t_{k-1,k} \\ & & & 1 & t_{k,k} \end{bmatrix},$$

we see that Algorithm A.3 is characterized by the identities

$$QV_k = V_k T_k + v_{k+1} e_k^T$$

(2.19a)

$$Q^T W_k = W_k T_k^T + t_{k,k+1} w_{k+1} e_k^T.$$  

(2.19b)

Applying Principle 2.1 to Algorithm A.3, we obtain Algorithm 2.3 formulated in terms of reduced-space quantities and the preconditioner $G$.

An alternative to Algorithm A.3 is the variant given by Chan et al. (1998) and Freund et al. (1993). When applied to $Q$ and $c$ from (2.6) it may be stated as Algorithm A.4. An advantage of this variant is, as we will see in the next section, that it may be easily reformulated so as to avoid operator-vector products with $Q^T$ altogether.

In Algorithm A.3, the choice of off-diagonal entries of $T_{k+1,k+1}$ was arbitrary; any pair $t_{k+1,k}$ and $t_{k,k+1}$ satisfying $t_{k+1,k} t_{k,k+1} = \langle v_{k+1}, w_{k+1} \rangle$ suffice. The choice of setting sub-diagonal entries to 1 is interesting because this form happens to coincide with the form of the tridiagonal matrix generated by Algorithm A.4. In Algorithm A.4, the differences with Algorithm A.3 are that the three-term recurrences for $v_{k+1}$ and $w_{k+1}$ have the same form, and that the process is characterized by the identities

$$QV_k = V_k T_k + v_{k+1} e_k^T$$

(2.20a)

$$Q^T W_k = W_k T_k^T + t_{k,k+1} w_{k+1} e_k^T.$$  

(2.20b)
Algorithm 2.3 Preconditioned Lanczos Bi-Orthogonalization for $V_k, W_k$ and $T_k$

Require: $Q$, $Q^T$, $G = G^T \succ 0$, $c$ and $x_0$
1: Set $v_0 = w_0 = 0$, $t_{0,1} = t_{1,0} = 1$
2: Set $s = c - Qx_0$. Set $w_1$ such that $\langle s, w_1 \rangle = 1$.
3: Solve $Gv_1 = s$ for $v_1$ \hspace{1cm} // Initial Krylov vectors
4: $k = 1$
5: while $t_{k-1,k} \neq 0$ do
6: \hspace{1cm} $s = Qv_k$ and $u = Q^T w_k$
7: \hspace{1cm} Solve $Gv_{k+1} = s$ and $Gw_{k+1} = u$ \hspace{1cm} // Compute next Krylov vectors
8: \hspace{1cm} $t_{k,k} = \langle s, w_k \rangle$
9: \hspace{1cm} $v_{k+1} = v_{k+1} - t_{k,k}v_k - t_{k-1,k}v_{k-1}$
10: \hspace{1cm} $w_{k+1} = w_{k+1} - t_{k,k}w_k - w_{k-1}$ \hspace{1cm} // Bi-orthogonalization
11: \hspace{1cm} $t_{k,k+1} = \langle s, w_{k+1} \rangle$
12: \hspace{1cm} if $t_{k,k+1} \neq 0$ then
13: \hspace{2cm} $w_{k+1} = w_{k+1} / t_{k,k+1}$
14: \hspace{2cm} $t_{k+1,k} = 1$
15: \hspace{1cm} $k = k + 1$


Theorem 2.1. If Algorithm A.4 does not break down before step $m$, then the families of vectors \{v_1, \ldots, v_m\} and \{w_1, \ldots, w_m\} are bi-orthogonal in the sense that $\langle v_i, w_j \rangle = 0$ if and only if $i \neq j$.

As a consequence of (2.20) and Theorem 2.1, we have, in exact arithmetic,

$$W_k^T Q V_k = V_k^T Q^T W_k = D_k T_k, \quad D_k = \text{diag}(\delta_1, \ldots, \delta_k),$$

and $D_k T_k$ is clearly still tridiagonal.

Once again, we may apply Principle 2.1 to Algorithm A.4 to obtain Algorithm 2.4.

Note that, as with the Arnoldi process, when $Q$ is symmetric, both Algorithm 2.3 and 2.4 reduce to the symmetric Lanczos process stated as Algorithm A.2.

2.2.3. Transpose-Free Bi-Orthogonal Bases. An annoyance of the Lanczos bi-orthogonalization process is the need to form products with both the operator and its transpose. Fortunately, the transpose may be avoided at the cost of a pair of extra operator-vector products and a more complicated recurrence to obtain $T_{k+1,k}$ (Brezinski and Redivo-Zaglia, 1998; Chan et al., 1998). Our version follows from Algorithm A.4 and is stated as Algorithm A.5.

In practice, scaled versions of this basic recurrence may be preferred to avoid computational over- and under-flow (Chan et al., 1998). The reduced-space variant of Algorithm A.5 is stated as Algorithm 2.5 on the following page.

At line 3 of Algorithm 2.5, a possible choice for $w$ and $w_G$ are $w_G := v_1$ and $w = G^{-1} w_G$.

2.3. Iteration in the Full Space. We are now in a position to describe how all of the Krylov methods we have considered in the null space of $A$ may actually be
Algorithm 2.4 Preconditioned Variant of the Lanczos Bi-Orthogonalization Process for $V_k, W_k$ and $T_k$

Require: $Q, Q^T, G = G^T > 0$, $c$ and $x_0$

1. Set $v_0 = u_0 = 0$ and $\delta_0 = 1$
2. Set $s = c - Qx_0$ and $w_1$ such that $\delta_1 := \langle s, w_1 \rangle \neq 0$
3. Solve $Gv_1 = s$ for $v_1$  // Initial Krylov vectors
4. $k = 1$
5. while $\delta_k \neq 0$
6. $\delta = Qv_k$ and $u = Q^T w_k$
7. Solve $Gv_{k+1} = s$ and $Gw_{k+1} = u$  // Compute next Krylov vectors
8. $t_{k,k} = \langle s, w_k \rangle / \delta_k$
9. $t_{k-1,k} = \delta_k / \delta_{k-1}$
10. $v_{k+1} = v_{k+1} - t_{k,k} v_k - t_{k-1,k} v_{k-1}$
11. $w_{k+1} = w_{k+1} - t_{k,k} w_k - t_{k-1,k} w_{k-1}$  // Bi-orthogonalization
12. $\delta_{k+1} = \langle s, w_{k+1} \rangle$
13. $k = k + 1$

Algorithm 2.5 Preconditioned Transpose-Free Lanczos Bi-Orthogonalization Process for $V_k$

Require: $Q, G = G^T > 0$, $c$ and $x_0$

1. Set $v_0 = u_0 = 0$ and $\delta_0 = 1$
2. Set $y = c - Qx_0$ and solve $Gv_1 = y$ for $v_1$  // Initial Krylov vector
3. Set $w$ and $w_G := Gw$ such that $\delta_1 := \langle v_1, w_G \rangle \neq 0$
4. $k = 1$
5. while $\delta_k \neq 0$
6. Solve $Gs = Qv_k$ for $s$, $t_{k,k} = \langle s, w_G \rangle / \delta_k$ and $t_{k-1,k} = \delta_k / \delta_{k-1}$
7. $u_k = s - t_{k,k} v_k - t_{k-1,k} u_{k-1}$
8. Set $d = u_k - t_{k-1,k} u_{k-1}$ and solve $Gs = Qd$ for $s$
9. $v_{k+1} = s - t_{k,1} d + t_{k-1,1} u_{k-1}$  // Compute next Krylov vector
10. $\delta_{k+1} = \langle v_{k+1}, w_G \rangle$
11. $k = k + 1$

applied in the original (full) space. Recall that in the full space we have

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

in which case we have

$$x_k = Z x_k = Z V_k^c z_k \quad \text{or} \quad x_k = Z x_k = Z V_k z_k .$$

Consider also a preconditioner of the form

$$G := Z^T G Z ,$$

where $G$ satisfies the following requirement.

Requirement 2.1. The matrix $G$ is symmetric and positive definite on Null($A$).

Such a requirement on $G$ clearly implies that $G$ is symmetric, positive definite. But while choosing $G$ itself to be positive definite will suffice, this is far from necessary and indeed undesirable if, as is common for constrained optimization, $Q$ is indefinite.
2.3.1. Orthogonal-Basis Methods. Consider first iterates generated according to \( x_k = Zx_k = ZV_z k \) by the Arnoldi process. We now apply the following principle to express Algorithm 2.2 in terms of full-space quantities.

**Principle 2.2.**

1. Basis vectors transform according to \( v_k = Zu_k \). Because of (2.3), each assignment of \( v_k \) in the reduced space has the form \( (Z^T G Z)^{-1} u \) for some \( u \) of the form \( u = Z^T u \). Thus, we obtain \( v_k = Z(Z^T G Z)^{-1} Z^T u \).

2. Inner products of the form \( \langle u, v_k \rangle \) become \( \langle Z^T u, (Z^T G Z)^{-1} Z^T u \rangle = \langle u, v_k \rangle \).

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

\[
P_G := Z(Z^T G Z)^{-1} Z^T
\]  

(2.21)

has properties akin to those of an orthogonal projector into the nullspace of \( A \). More precisely, \( P_G \) is an oblique projector into \( \text{Null}(A) \).

We summarize a few immediate properties of \( P_G \) in the following result.

**Theorem 2.2.** Let \( P_G \) be defined as in (2.21) where \( G \) satisfies Requirement 2.1. Then

1. \( P_G GP_G = P_G \) and \((P_G G)^2 = P_G G \)
2. \( P_G x = 0 \) for all \( x \in \text{Range}(A^T) \)
3. \( P_G G x = x \) for all \( x \in \text{Null}(A) \)
4. \( Z^T G P_G = Z^T \) and \( Z^T G(I - P_G G) = 0 \)
5. \( P_G G Z = Z \).

A consequence of the above is that finding \( Z \) is unnecessary as it is easy to show (Gould et al., 2001) that \( v_k = P_G u \) may instead be computed by solving the constraint preconditioned (Keller et al., 2000) saddle-point system

\[
\begin{bmatrix}
G & A^T \\
A & & \\
\end{bmatrix}
\begin{bmatrix}
v_k \\
y_k \\
\end{bmatrix} = \begin{bmatrix}
u \\
0 \\
\end{bmatrix}.
\]  

(2.22)

For the coefficient matrix \( K_G \) of (2.22) to be a constraint preconditioner, all that is needed is that \( G \) satisfies Requirement 2.1 or, equivalently, that \( K_G \) be nonsingular and have precisely as many negative eigenvalues as \( A \) has linearly independent rows.

Typically, constraint preconditioners are obtained and used by factorizing—either explicitly (Keller et al., 2000) or implicitly (Dollar et al., 2007)—\( K_G \); this has the further advantage that the required initial value \( x_F \) may be found by solving

\[
\begin{bmatrix}
G & A^T \\
A & & \\
\end{bmatrix}
\begin{bmatrix}
x_F \\
y_0 \\
\end{bmatrix} = \begin{bmatrix}
g \\
b \\
\end{bmatrix}
\]  

(2.23)

for arbitrary \( g \)—usually, \( g = 0 \) or \( g = a \).

Applying Principle 2.2 to Algorithm 2.2, we derive the class of Krylov methods for (2.1) described by Algorithm 2.6 on the following page. We note that although formally we may express methods using Algorithm 2.6, an actual implementation will usually be more 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.
Algorithm 2.6  Arnoldi-Based Projected Krylov Methods for (2.1)

Require: $Q$, $P$, $c$ and $x_0$

1: $u = c - Qx_0$  
2: Compute $v_1 = Pu$  // Initial Krylov vector
3: $h_{1,0} = \sqrt{\langle v_1, u \rangle}$  // Initial preconditioned residual norm
4: if $h_{1,0} \neq 0$ then
5: $v_1 = v_1/h_{1,0}$
6: $x_1 = V_1^0 \Theta_1(H_{1,0}, h_{1,0})$  // Initial solution estimate
7: $k = 1$
8: while $h_{k,k-1} \neq 0$ do
9: $u = Qv_k$  // Compute next Krylov vector
10: for $i = 1, \ldots, k$ do  // Modified Gram-Schmidt
11: $h_{i,k} = \langle v_i, u \rangle$
12: $v_{k+1} = v_{k+1} - h_{i,k} v_i$
13: $h_{k+1,k} = \sqrt{\langle v_{k+1}, u \rangle}$  // Preconditioned residual norm
14: if $h_{k+1,k} \neq 0$ then
15: $v_{k+1} = v_{k+1}/h_{k+1,k}$
16: $x_{k+1} = V_{k+1}^0 \Theta_{k+1}(H_{k+1,k}, h_{1,0})$  // Update solution estimate
17: $k = k + 1$

Observe that Algorithm 2.6 (and those that will follow) only aim to find $x_*$ satisfying (2.1) and not $y_*$. One way to obtain the complete solution is to compute an estimate of the $y_*$ component of the solution only once a good approximation to $x_*$ has been found. To do so, suppose that $x$ has been obtained as an estimate of $x_*$. We may then compute

$$y_* = \arg\min_{y \in \mathbb{R}^m} \| A^T y - (c - Qx) \|_N$$

(2.24)

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

$$\begin{bmatrix} G & A^T \\ A & \end{bmatrix} \begin{bmatrix} w_* \\ y_* \end{bmatrix} = \begin{bmatrix} c - Qx \\ 0 \end{bmatrix}$$

involving the matrix $K_G$ used in (2.22). The same is true using the dual to the semi-norm $\sqrt{\langle s, Gs \rangle}$ defined on the manifold $As = 0$ under the more general Requirement 2.1 on $G$ (see Conn et al., 2000, §2.2, and Gould et al., 2001, §6).

2.3.2. Bi-Orthogonal-Basis Methods. We now turn to iterates generated by $x_k = Z_k z_k$, i.e. a solution algorithm based on the bi-orthogonal basis computed by Algorithm 2.3 or 2.4. Following arguments similar to those used for Arnoldi-based algorithms, the analogue of Algorithm 2.6 for the class of projected Krylov algorithms for (2.1) based on Lanczos bi-orthogonalization may be stated as Algorithm 2.7 below. The variant based on Algorithm 2.3 appears as Algorithm A.6. Note that for each $k$, Algorithm 2.7 makes the implicit assignment $t_{k+1,k} = 1$.  

Commit 12affb2 by Dominique Orban on 2013-04-12 21:36:04 -0400
Algorithm 2.7 Lanczos-Based Projected Krylov Methods for (2.1)

Require: $Q$, $Q^T$, $P_G$, $c$ and $x_0$
1: Set $y_0 = u_0 = 0$ and $\delta_0 = 1$
2: Set $s = c - Qx_0$ and $w_1$ such that $\delta_1 := \langle s, w_1 \rangle \neq 0$
3: Compute $v_1 = P_G s$ for $v_1$
4: Compute the solution estimate $x_1 = V_1 \Theta_1 (T_{1,0}, t_{1,0})$ // Initial Krylov vectors
5: $k = 1$
6: while $\delta_k \neq 0$ do
7: $s = Qv_k$ and $u = Q^T w_k$
8: $v_{k+1} = P_G s$ and $w_{k+1} = P_G u$ // Compute next Krylov vectors
9: $t_{k,k} = \langle s, w_k \rangle / \delta_k$
10: $t_{k-1,k} = \delta_k / \delta_{k-1}$
11: $v_{k+1} = v_{k+1} - t_{k,k} v_k - t_{k-1,k} v_{k-1}$
12: $w_{k+1} = w_{k+1} - t_{k,k} w_k - t_{k-1,k} w_{k-1}$ // Bi-orthogonalization
13: $\delta_{k+1} = \langle s, w_{k+1} \rangle$
14: Compute the solution estimate $x_{k+1} = V_{k+1} \Theta_{k+1} (T_{k+1,k}, t_{1,0})$
15: $k = k + 1$

2.3.3. Transpose-Free Bi-Orthogonal Methods. Finally, we remain with iterates generated by $x_k = Zx_k = ZV_{k,z_k}$, but now consider the case where $V_k$ is calculated by the transpose-free Algorithm 2.5. Applying Principle 2.2, we have derived a transpose-free variant of Algorithm 2.7, stated as Algorithm 2.8.

Algorithm 2.8 Transpose-Free Lanczos-Based Projected Krylov Methods for (2.1)

Require: $Q$, $P_G$, $c$ and $x_0$
1: Set $y_0 = u_0 = 0$ and $\delta_0 = 1$
2: Set $y = c - Qx_0$ and compute $v_1 = P_G y$ // Initial Krylov vector
3: Set $w_G$ and $w = P_G fsw_G$ such that $\delta_1 := \langle v_1, w_G \rangle \neq 0$
4: Compute the solution estimate $x_1 = V_1 \Theta_1 (T_{1,0}, t_{1,0})$
5: $k = 1$
6: while $\delta_k \neq 0$ do
7: Compute $s = P_G (Qv_k)$, $t_{k,k} = \langle s, w_G \rangle / \delta_k$ and $t_{k-1,k} = \delta_k / \delta_{k-1}$
8: $u_k = s - t_{k,k} v_k - t_{k-1,k} u_{k-1}$
9: Set $d = u_k - t_{k-1,k} u_{k-1}$ and compute $s = P_G (Qd)$
10: $v_{k+1} = s - t_{k,k} d + t_{k-1,k} v_{k-1}$
11: $\delta_{k+1} = \langle v_{k+1}, w_G \rangle$
12: Compute the solution estimate $x_{k+1} = V_{k+1} \Theta_{k+1} (T_{k+1,k}, t_{1,0})$
13: $k = k + 1$

At line 3 of Algorithm 2.8, note that the choice of $w_G$ is irrelevant because the algorithm computes inner products between $w_G$ and vectors lying in the nullspace of $B$. Therefore, for any such vector $v$, we have $\langle v, w_G \rangle = \langle v, w \rangle$. A typical choice is $w := v_1$.

2.4. Constraint-Preconditioned Variants. In this section we establish equivalence relationships between the projected Krylov processes of §2.3 and the standard preconditioned processes of §2.2 applied to (2.1) with a constraint preconditioner of the form (2.22). Note that we temporarily ignore the fact the constraint preconditioner is
indefinite and thus flout conventional wisdom; we simply write what the preconditioned processes would be if it were to be employed.

2.4.1. Orthogonal-Basis Methods. Suppose Algorithm 2.2 is initialized with a starting guess of the form \((x_0, y_0)\) for which \(x_0 \in \text{Null}(A)\). Line 1 of Algorithm 2.2 reads

\[
u = \begin{bmatrix} c \\ 0 \\ Q \\ A \\ A^T \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \begin{bmatrix} c - Qx_0 + A^Ty_0 \end{bmatrix}
\]

because \(Ax_0 = 0\) so that the initial Krylov vector is given as the solution of

\[
\begin{bmatrix} G & A^T \\ A \\ A^T \end{bmatrix} \begin{bmatrix} v_{1,x} \\ v_{1,y} \end{bmatrix} = \begin{bmatrix} c + Qx_0 + A^Ty_0 \end{bmatrix}
\]

and this shows that \(v_{1,x}\) is identical to the vector \(v_1\) given at line 2 of Algorithm 2.6 since the term \(A^Ty_0\) does not impact the component \(v_{1,x}\) of the solution—see the second property of Theorem 2.2. In particular, \(v_{1,x}\) lies in the nullspace of \(A\). At line 3 of Algorithm 2.2, we compute

\[
h_{1,0} = \sqrt{\langle v_{1,x}, c - Qx_0 + A^Ty_0 \rangle + \langle v_{1,y}, 0 \rangle} = \sqrt{\langle v_1, c - Qx_0 \rangle},
\]

which is identical to \(h_{1,0}\) computed at line 3 of Algorithm 2.6.

Assuming now that \(v_{k,x}\) lies in the nullspace of \(A\) and coincides with \(v_k\), we establish by recursion that \(v_{k+1,x}\) also lies in the nullspace of \(A\) and coincides with \(v_{k+1}\).

The vector \(u\) computed at line 8 of Algorithm 2.2 is

\[
u = \begin{bmatrix} Q \\ A \\ A^T \end{bmatrix} \begin{bmatrix} v_{k,x} \\ v_{k,y} \end{bmatrix} = \begin{bmatrix} Qv_{k,x} + A^Tv_{k,y} \\ 0 \end{bmatrix}
\]

and the first component of \(u\) is \(u + A^Tv_{k,y}\), where \(u\) is computed at line 9 of Algorithm 2.6. Next, \(v_{k+1}\) is given as the solution of

\[
\begin{bmatrix} G & A^T \\ A \\ A^T \end{bmatrix} \begin{bmatrix} v_{k+1,x} \\ v_{k+1,y} \end{bmatrix} = \begin{bmatrix} u + A^Tv_{k,y} \\ 0 \end{bmatrix}.
\]

This shows that \(v_{k+1,x} = v_{k+1}\). Moreover, \(Av_{k+1,x} = 0\).

Finally, Algorithm 2.2 computes

\[
h_{i,k} = \langle v_{i,x}, Qv_{k,x} + A^Tv_{k,y} \rangle + \langle v_{i,y}, 0 \rangle = \langle v_{i,x}, Qv_{k,x} \rangle = \langle v_i, u \rangle = h_{i,k},
\]

where we used the fact that \(Av_{i,x} = 0\) for all \(i \leq k\). Similarly, we can show that \(h_{k+1,k} = h_{k+1,k}\). We have established the following result.

Theorem 2.3. Algorithm 2.2 applied to (2.1) with an initial guess of the form \((x_0, y_0)\) satisfying \(Ax_0 = 0\) and using the constraint pre conditioner (2.22) generates Krylov vectors \(v_k = (v_{k,x}, v_{k,y})\) such that at each iteration \(k\), \(v_{k,x}\) is the Krylov vector \(v_k\) generated by Algorithm 2.6 with initial guess \(x_0\) at iteration \(k\). The temporary vector \(u\) has the form \((u + A^Tw, 0)\) for some vector \(w\), where \(u\) is the corresponding temporary vector generated by Algorithm 2.6. In addition, the Hessenberg matrix generated is identical to that generated by Algorithm 2.6.
In exact arithmetic, the advantage of Algorithm 2.6 over Algorithm 2.2 with a constraint preconditioner is in terms of memory requirements. The vectors $u_k$ and $v_k$ generated have size $n$ while the vectors $u$ and $v_k$ of Algorithm 2.2 have size $n + m$. The advantage of Algorithm 2.2 with constraint preconditioner is that the subvectors $v_{k,y}$ may be used to recur approximations to the $y$ segment of the solution of (2.1). Depending on how the application of $P_G$ is implemented, however, the same approximations may be obtained and recurred in Algorithm 2.6.

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. 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 (2.22) is a well-defined Krylov method.

Example: Projected GMRES. Standard GMRES performs $m$ iterations of Algorithm 2.1 to obtain $V_{m+1}$ and $H_{m+1,m}$, and then computes an approximation $x_{m+1}$ in the $m + 1$-st Krylov subspace $x_{m+1} := V_{m+1}z_{m+1}$ by solving the linear least-squares problem

$$\minimize_{z_{m+1}} \|h_{1,0} e_1 - H_{m+1,m}z_{m+1}\|_2.$$  \hfill (2.25)

The preconditioned and projected GMRES algorithms are based on Algorithms 2.2 and 2.6, respectively. Besides Algorithm 2.6, 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 2.3 and (2.25), it is clear that the $x_{m+1}$ defined by the projected variant coincides with the $x_{m+1}$ defined by the constraint-preconditioned variant.

### 2.4.2. Bi-Orthogonal-Basis Methods

As in §2.4.1, we establish a formal equivalence between Algorithm A.6 and Algorithm 2.3 applied to the augmented system (2.1) with the preconditioner (2.22). Suppose the latter is initialized with $c = (c,0)$ and $x_0 = (x_0,y_0)$ such that $Ax_0 = 0$. We first compute

$$s = \begin{bmatrix} c \\ 0 \end{bmatrix} - \begin{bmatrix} Q & A^T \\ A \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \begin{bmatrix} s - A^T y_0 \\ 0 \end{bmatrix}.$$  

During initialization, we thus have $v_{1,x} = P_G(s - A^T y_0) = P_G(s) = v_1$. Suppose Algorithm 2.3 selects $w_1$ such that $A w_{1,x} = 0$.

Assume that for iterations 1 through $k - 1$, each $v_k$ and $w_k$ is such that $Av_{k,x} = Aw_{k,x} = 0$. During the $k$-th iteration, Algorithm 2.3 computes

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

and

$$u = \begin{bmatrix} Q & A^T \\ A \end{bmatrix} \begin{bmatrix} w_{k,x} \\ w_{k,y} \end{bmatrix} = \begin{bmatrix} Q w_{k,x} + A^T w_{k,y} \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} u + A^T w_{k,y} \\ 0 \end{bmatrix}.$$
We now establish directly that standard MINRES applied with a constraint pre
with the iterates generated by the projected variant of this method. Among others, this
which is the same
t
2.7, and similar conclusions can be drawn about iterative methods deriving from those
preconditioner (2.22) is a well-defined Krylov method.
conditioner also minimizes (3.1), and therefore that MINRES with the indefinite
quantity minimized by the projected MINRES at each iteration may
generate their attention to the case where Q is symmetric and positive definite.
example of the projected MINRES algorithm, in which approximate solutions xk =
their 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 bi-conjugate
orthogonal method generalize results of Rozlozník and Simoncini (2002), who restrict
Note that Theorems 2.3, 2.4 and the corresponding result for transpose-free bi-
orthogonal methods, including corresponding variants of QMR and Bi-CG.

3. Example: Projected MINRES. In this section, we consider the specific
example of the projected MINRES algorithm, in which approximate solutions xk =
V̂kz̄k are chosen so as to minimize the norm of the residual ṙ := c − Qxk in the
norm defined by G−1. Recall that c = ZTc, Q = ZTQZ and G = ZT GZ is positive
definite. The quantity minimized by the projected MINRES at each iteration may
thus be written
\[ \|r_k\|_{G^{-1}}^2 = \langle r_k, (Z^T G Z)^{-1} r_k \rangle. \] (3.1)

We now establish directly that standard MINRES applied with a constraint pre
conditioner also minimizes (3.1), and therefore that MINRES with the indefinite
preconditioner (2.22) is a well-defined Krylov method.

In other words, v_{k+1,x} = P_G s_x = P_G (s + A^Tw_{k,y}) = P_G s = v_{k+1}. Similarly, w_{k+1,y} = w_{k+1}. Moreover, A v_{k+1,x} = A w_{k+1,x} = 0 before the computation of t_{k,k}.

In Algorithm 2.3,
\[ t_{k,k} = \langle s, w_k \rangle = \langle s_x, w_{k,x} \rangle = \langle s + A^Tw_{k,y}, w_k \rangle = \langle s, w_k \rangle, \]
which is the same t_{k,k} computed by Algorithm A.6. Similarly, we see that the two
algorithms compute the same t_{k+1,k}. By recursion, we have established the following
result.

**Theorem 2.4.** Algorithm 2.3 applied to (2.1) with an initial guess of the form (x_0, y_0) satisfying Ax_0 = 0 and using the constraint preconditioner (2.22) generates Krylov vectors v_k = (v_{k,x}, v_{k,y}) and w_k = (w_{k,x}, w_{k,y}) such that at each iteration k, v_{k,x} and w_{k,x} are the Krylov vectors v_k and w_k generated at iteration k of Algorithm A.6 with initial guess x_0 and using w_1 = w_{1,x} chosen so that Aw_1 = 0. The temporary vectors s and u have the form (s + A^Tw_{k,y}, 0) and (u + A^Tw_{k,y}, 0), where s and u are the corresponding temporary vectors generated by Algorithm A.6. In addition, the tridiagonal matrix generated is identical to that generated by Algorithm A.6.

A consequence of Theorem 2.4 is that for any Krylov method deriving from the
Lanczos bi-orthogonalization 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 bi-conjugate
gradient algorithm (Bi-CG) and the quasi-minimum residual method (QMR).

**2.4.3. Transpose-Free Bi-Orthogonal Methods.** Proceeding exactly as in
\S2.4.2, a result identical to Theorem 2.4 can be established about Algorithms 2.4 and
2.7, and similar conclusions can be drawn about iterative methods deriving from those
processes, including corresponding variants of QMR and Bi-CG.

Note that Theorems 2.3, 2.4 and the corresponding result for transpose-free bi-
orthogonal method generalize results of Rozlozník and Simoncini (2002), who restrict
their attention to the case where Q is symmetric and positive definite.
Theorem 2.3 guarantees that with appropriate initial conditions, the approximate solution \((x_k, y_k)\) generated at the \(k\)-th iteration of the constraint-preconditioned MINRES is such that \(x_k \in \text{Null}(A)\). Consider the residual
\[
r_k := \begin{bmatrix} c \\ 0 \end{bmatrix} - \begin{bmatrix} Q & A^T \\ A & y_k \end{bmatrix} \begin{bmatrix} x_k \\ y_k \end{bmatrix} = \begin{bmatrix} c - Qx_k - A^Ty_k \end{bmatrix}.
\]

It is tempting to claim that it is this residual that is minimized in the norm defined by the inverse of the preconditioner (2.22). Unfortunately, the preconditioner is indefinite. The appropriate interpretation of this claim is to consider the seminorm associated to the preconditioner and defined by
\[
\|r_k\|_{\|G\|} := \langle r_k, s_k \rangle \quad \text{where} \quad \begin{bmatrix} G & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} s_k \\ t_k \end{bmatrix} = \begin{bmatrix} r_k \\ 0 \end{bmatrix}, \quad (3.2)
\]
i.e.,
\[
\|r_k\|_{\|G\|} = \langle c - Qx_k - A^Ty_k, s_k \rangle = \langle c - Qx_k, s_k \rangle. \quad (3.3)
\]
Note that \(\|r_k\|_{\|G\|}\) measures deviation of \(r_k\) from the range space of \(A^T\) and vanishes if and only if \(r_k\) is orthogonal to the nullspace of \(A\). In effect, \(\|\cdot\|_{\|G\|}\) defines a norm on the nullspace of \(A\). Such seminorms have been used in optimization contexts (Conn et al., 2000). Observe from (3.2) that \(s_k \in \text{Null}(A)\) and therefore, \(s_k = Zs_k\) for some \(s_k\). The first block equation of (3.2) premultiplied by \(Z^T\) then yields
\[
s_k = (Z^TGZ)^{-1}Z^T(c - Qx_k) = (Z^TGZ)^{-1}r_k.
\]
Introducing this expression of \(s_k\) into (3.3), we finally obtain
\[
\|r_k\|_{\|G\|} = \langle Z^T(c - Qx_k), s_k \rangle = \langle r_k, (Z^TGZ)^{-1}r_k \rangle = \|r_k\|_{G^{-1}},
\]
which coincides with (3.1). On substituting \(r_k = Z^Tr_k\) in this last identity, we also see that
\[
\|r_k\|_{\|G\|} = \langle r_k, Z(Z^TGZ)^{-1}Z^Tr_k \rangle = \langle r_k, P_G\theta(r_k) \rangle = \|P_G\theta(r_k)\|_{2}^2.
\]
We conclude that the projected MINRES algorithm minimizes the Euclidian norm of the projected residual. Since the spaces over which this quantity are minimized are the same in both methods (Rozlozník 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.

3.1. Implementation considerations. 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 \(x_k\) in the nullspace of \(A\). While this is always true in exact arithmetic, the accumulation of rounding errors can quickly give \(x_k\) a non-negligible component in the range of \(A^T\) which may cause the method to break down.

To help to ameliorate this effect Gould et al. (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 approximate 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:

```plaintext
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 $G$ is the positive diagonal matrix whose nonzero entries coincide with those of $Q$.

Example 3.1 is chosen so that the approximate leading block $G$ is a good approximation of the actual $Q$. 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 3.1.

![Fig. 3.1: 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.](image)

From Figure 3.1, we see that—as was the case with projected CG (Gould et al., 2001)—it is advisable to apply these methods with iterative refinement. 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 $Q$, 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.
4. Eigenvalue Bounds and Convergence. The convergence of Krylov subspace methods is known to be strongly linked to the clustering of the eigenvalues of the preconditioned system. If the matrix and preconditioner are symmetric, then the eigenvalues tell the whole story (Paige and Saunders, 1975). For non-symmetric matrices well clustered eigenvalues do not guarantee rapid convergence, as famously demonstrated by Greenbaum et al. (1996), but they often do in practice—see, e.g., Pestana and Wathen (2011). Nachtigal et al. (1992) illustrate that the convergence of GMRES depends on the eigenvalues, not the singular values, of the operator.

In exploring the convergence of the proposed methods, it is therefore instructive to consider the generalized eigenvalue problem

\[
\begin{bmatrix} Q & A^T \\ A & \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \lambda \begin{bmatrix} G & A^T \\ A & \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix},
\]

\[(4.1)\]

The following theorem was proved by Keller, Gould, and Wathen (2000) in the symmetric case and generalized by Cao (2002) to the non-symmetric case. Note that the statement of Theorem 2.1 by Keller, Gould, and Wathen assumes symmetry of \( Q \) and \( G \), but this fact is not needed in the proof.

**Theorem 4.1.** Suppose that \( A \) is \( m \)-by-\( n \) and has full row rank. Then the generalized eigenvalue problem (4.1) has

- an eigenvalue at 1 with multiplicity \( 2m \), and
- \( n - m \) eigenvalues defined by the generalized eigenvalue problem

\[
Z^T Q Z x = \lambda Z^T G Z x.
\]

This result is unsurprising, given the close relationship between solving (2.1) with a constraint preconditioner and solving the reduced problem (2.3) described in the preceding sections.

When \( Z^T Q Z \) is unsymmetric, the non-unit eigenvalues will, in general, be complex and we are unable to apply an interlacing theorem to tie these eigenvalues to the eigenpencil \((Q, G)\). Fortunately, when \( Z^T Q Z \) is symmetric, but possibly indefinite, we can say more. First, note that in this case—since Requirement 2.1 implies that \( Z^T G Z \) is symmetric positive definite—the eigenvalues are all real (Keller et al., 2000). Also, in this case Keller, Gould, and Wathen also show that the eigenvalues of the generalized eigenvalue problem of Theorem 4.1 interlace the generalized eigenvalues, which satisfy \( Q x = \lambda G x \). Therefore, provided that \( G \) is a good approximation to \( Q \), the eigenvalues satisfying (4.1) will be well clustered and we can expect good convergence of a projected Krylov method.

5. 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 et al. (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

\[
\begin{align*}
t_{k+1,k} v_{k+1} &= Q w_k - t_{k,k} v_k - t_{k-1,k} v_{k-1}, \\
t_{k,k+1} w_{k+1} &= Q^T v_k - t_{k,k} w_k - t_{k-1,k} w_{k-1},
\end{align*}
\]

\[(5.1a, 5.1b)\]
where $t_{k,k} := \langle v_k, Qw_k \rangle$ 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

$$QW_k = V_kT_k + t_{k+1,k}v_{k+1}e_k^T$$
$$Q^TV_k = W_kT_k + t_{k,k+1}w_{k+1}e_k^T,$$

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 $\langle v_j, Qw_k \rangle = \langle w_j, Qv_k \rangle = 0$ for $k < j - 1$ (Saunders et al., 1988, Theorem 1). The above relations differ considerably from (2.19) and (2.20) 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 §2.3.

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

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

where $C$ is typically symmetric and positive semi-definite (Elman et al., 2005). Assuming more generally that $C$ may be decomposed as $ED^TE^T$ where $D$ is nonsingular, and introducing $w := -DE^Ty$ as suggested by Dollar et al. (2007), such systems may be equivalently reformulated

$$\begin{bmatrix} Q & D^{-1}A^T \\ A^T & E \\ D^{-1} & E^T \end{bmatrix} \begin{bmatrix} x \\ w \end{bmatrix} = \begin{bmatrix} a \\ 0 \\ b \end{bmatrix}.$$ 

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

An instructive way to summarize the difference between projected and constraint-preconditioned Krylov methods is that projected methods use the preconditioner

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

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

Finally, we note that it appears as if $A$ must be available as an explicit matrix to factorize (2.22). There are other ways to compute projections. One of them is to solve a linear least-squares problem of the form (2.24) using an appropriate metric, which only requires $A$ 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.

6. Outlook. In most programming languages, it is possible to implement projected Krylov methods non intrusively, i.e., without modifying the underlying Krylov method, by specifying an appropriate preconditioner—be it a function or an abstract object—that performs the projection, the iterative refinement but also the residual...
update. One language where this is not possible is Matlab, the limitation being due to Matlab’s passing arguments by value and not by reference.

We have only considered saddle-point systems in which, in PDE parlance, the “gradient” and “divergence” terms are adjoints of one another. In optimize-then-discretize approaches to solving certain PDE-constrained optimization problems, we encounter saddle-point problems of the form

\[
\begin{bmatrix}
Q & B^T \\
A & 0
\end{bmatrix}
\begin{bmatrix}
x^* \\
y^*
\end{bmatrix}
=
\begin{bmatrix}
c \\
0
\end{bmatrix},
\]

(6.1)

where \(Q\), as before, may be unsymmetric. Variants of the approach described in the present paper may be employed to solve (6.1). We leave the study of such variants for future work.

References.


Appendix A. Preconditioned-Space Processes and Variants. This section contains the symmetric Lanczos, unsymmetric Lanczos and transpose-free variants formulated in preconditioned space. In addition, it contains the unsymmetric Lanczos process discussed by Chan et al. (1998) and Freund et al. (1993).

A.1. The Symmetric Lanczos Process. When $Q$ is symmetric in Algorithm 2.1, $H_{k+1,k}$ is tridiagonal and the resulting simplified procedure is the symmetric Lanczos process, summarised as Algorithm A.1.

Algorithm A.1 Lanczos Process for $V_k^o$ and $T_{k-1,k}$

Require: $Q$, $c$ and $x_0$
1: Set $v_0^o = 0$ and $v_1^o = c - Qx_0$ \hspace{1cm} // Initial Krylov vector
2: $t_{1,0} = \sqrt{\langle v_1^o, v_1^o \rangle}$ \hspace{1cm} // Initial residual norm
3: if $t_{1,0} \neq 0$ then
4: $v_1^o = v_1^o / t_{1,0}$
5: $k = 1$
6: while $t_{k,k-1} \neq 0$ do
7: $v_{k+1} = Qv_k^o$ \hspace{1cm} // Compute next Krylov vector
8: $t_{k,k} = \sqrt{\langle v_k^o, v_k^o \rangle}$
9: $v_{k+1} = v_{k+1} - t_{k,k}v_k^o - t_{k,k-1}v_k^o$ \hspace{1cm} // Modified Gram-Schmidt
10: $t_{k+1,k} = \sqrt{\langle v_{k+1}, v_{k+1} \rangle}$ \hspace{1cm} // Residual norm
11: if $t_{k+1,k} \neq 0$ then
12: $v_{k+1}^o = v_{k+1}^o / t_{k+1,k}$
13: $k = k + 1$

Algorithm A.1 is characterized by the identities

\[
QV_k = V_{k+1}T_{k+1,k} = V_kT_k + t_{k+1,k}v_k^oe_k^T, \tag{A.1}
\]

where $T_k$ is $k$-by-$k$ upper triangular and $T_{k+1,k}$ is $T_k$ with the extra row $t_{k+1,k}e_k^T$.

Applying the principles of §2.2.1 to Algorithm A.1, we obtain Algorithm A.2 on the following page.

A.2. The Lanczos Bi-Orthogonalization Process. The Lanczos (1950) bi-orthogonalization process as described in (2.8) and applied to $Q$ and $c$ leads to Algorithm A.3 on the following page. This is a special case of Saad (2003, Algorithm 7.1) and Golub and van Loan (1996, (9.4.7)).

The variant described by Chan et al. (1998) and Freund et al. (1993) is given as Algorithm A.4 on the following page.

A.3. Transpose-Free Lanczos Bi-Orthogonalization Process. It is straightforward to obtain a transpose-free variant of Algorithm A.4 that we state as Algorithm A.5 on page 26.

A.4. Projected Lanczos Bi-Orthogonalization Process. The analogue of Algorithm 2.6 for the class of projected Krylov algorithms for (2.1) based on the Lanczos bi-orthogonalization Algorithm 2.3 may be stated as Algorithm A.6.
Algorithm A.2 Preconditioned Lanczos Process for $V_k^c$ and $T_{k-1,k}$

Require: $Q$, $G = G^T > 0$, $c$ and $x_0$
1: Set $v_0^c = 0$, $u = c - Qx_0$ and solve $Gu_1 = u$ for $v_1^c$ // Initial Krylov vector
2: $t_{1,0} = \sqrt{(u,v_1^c)}$ // Initial residual norm
3: if $t_{1,0} \neq 0$ then
4: $v_1^c = v_1^c/t_{1,0}$
5: $k = 1$
6: while $t_{k,k-1} \neq 0$ do
7: Set $u = Qv_k^c$ and solve $Gu_{k+1} = u$ for $v_{k+1}^c$ // Compute next Krylov vector
8: $t_{k,k} = (u,v_k^c)$
9: $v_{k+1}^c = v_{k+1}^c - t_{k,k}v_k^c - t_{k,k-1}v_{k-1}^c$ // Modified Gram-Schmidt
10: $t_{k+1,k} = \sqrt{(u,v_{k+1}^c)}$ // Residual norm
11: if $t_{k+1,k} \neq 0$ then
12: $v_{k+1}^c = v_{k+1}^c/t_{k+1,k}$
13: $k = k + 1$

Algorithm A.3 Lanczos Bi-Orthogonalization Process for $V_k$, $W_k$ and $T_k$

Require: $Q$, $Q^T$, $c$ and $x_0$
1: Set $v_0 = w_0 = 0$, $t_{0,1} = t_{1,0} = 1$
2: Set $v_1 = c - Qx_0$ and $w_1$ such that $\langle v_1, w_1 \rangle = 1$ // Initial Krylov vectors
3: $k = 1$
4: while $t_{k-1,k} \neq 0$ do
5: $v_{k+1} = Qv_k$ and $w_{k+1} = Q^Tw_k$ // Compute next Krylov vectors
6: $t_{k,k} = \langle v_{k+1}, w_k \rangle$
7: $v_{k+1} = v_{k+1} - t_{k,k}v_k - t_{k,k-1}v_{k-1}$
8: $w_{k+1} = w_{k+1} - t_{k,k}w_k - w_{k-1}$ // Bi-orthogonalization
9: $t_{k+1,k} = \langle v_{k+1}, w_{k+1} \rangle$
10: if $t_{k+1,k} \neq 0$ then
11: $w_{k+1} = w_{k+1}/t_{k+1,k}$
12: $t_{k+1,k} = 1$
13: $k = k + 1$

Algorithm A.4 Variant of the Lanczos Bi-Orthogonalization Process for $V_k$, $W_k$ and $T_k$

Require: $Q$, $Q^T$, $c$ and $x_0$
1: Set $w_0 = 0$ and $\delta_0 = 1$
2: Set $v_1 = c - Qx_0$ and $w_1$ such that $\delta_1 := \langle v_1, w_1 \rangle \neq 0$ // Initial Krylov vectors
3: $k = 1$
4: while $\delta_k \neq 0$ do
5: $v_{k+1} = Qv_k$ and $w_{k+1} = Q^Tw_k$ // Compute next Krylov vectors
6: $t_{k,k} = \langle v_{k+1}, w_k \rangle/\delta_k$
7: $t_{k-1,k} = \delta_k/\delta_{k-1}$
8: $v_{k+1} = v_{k+1} - t_{k,k}v_k - t_{k,k-1}v_{k-1}$
9: $w_{k+1} = w_{k+1} - t_{k,k}w_k - t_{k,k-1}w_{k-1}$ // Bi-orthogonalization
10: $\delta_{k+1} = \langle v_{k+1}, w_{k+1} \rangle$
11: $k = k + 1$
Algorithm A.5 Transpose-Free Lanczos Bi-Orthogonalization Process for $V_k$

Require: $Q$, $c$ and $x_0$
1: Set $v_0 = u_0 = 0$ and $\delta_0 = 1$
2: Set $v_1 = c - Qx_0$ and $w$ such that $\delta_1 := \langle v_1, w \rangle \neq 0$  
   // Initial Krylov vector
3: $k = 1$
4: while $\delta_k \neq 0$ do
5:   Set $s = Qv_k$, $t_{k,k} = \langle s, w \rangle / \delta_k$ and $t_{k-1,k} = \delta_k / \delta_{k-1}$
6:   $u_k = s - t_{k,k}v_k - t_{k-1,k}u_{k-1}$
7:   Set $d = u_k - t_{k-1,k}u_{k-1}$ and $s = Qd$
8:   $v_{k+1} = s - t_{k,k}d + t_{k-1,k}^2v_{k-1}$
9:   $\delta_{k+1} = \langle v_{k+1}, w \rangle$
10: $k = k + 1$

Algorithm A.6 Lanczos-Based Projected Krylov Methods for (2.1)

Require: $Q$, $Q^T$, $P_G$, $c$ and $x_0$
1: Set $v_0 = w_0 = 0$, $t_{1,0} = t_{1,0} = 1$
2: Set $s = c - Qx_0$. Set $w_1$ such that $\langle s, w_1 \rangle = 1$.
3: Compute $v_1 = P_G(s)$  
   // Initial Krylov vectors
4: Compute the solution estimate $x_1 = V_1\Theta_1(T_{1,0}, t_{1,0})$
5: $k = 1$
6: while $t_{k-1,k} \neq 0$ do
7:   $s = Qv_k$ and $u = Q^Tw_k$
8:   Compute $v_{k+1} = P_G(s)$ and $w_{k+1} = P_G(u)$  
    // Compute next Krylov vectors
9:   $t_{k,k} = \langle s, w_k \rangle$
10: $v_{k+1} = v_{k+1} - t_{k,k}v_k - t_{k-1,k}v_{k-1}$
11: $w_{k+1} = w_{k+1} - t_{k,k}w_k - w_{k-1}$  
    // Bi-orthogonalization
12: $t_{k,k+1} = \langle s, w_{k+1} \rangle$
13: if $t_{k,k+1} \neq 0$ then
14:   $w_{k+1} = w_{k+1} / t_{k,k+1}$
15: $t_{k+1,k} = 1$
16: Compute the solution estimate $x_{k+1} = V_{k+1}\Theta_{k+1}(T_{k+1,k}, t_{1,0})$
17: $k = k + 1$