Science & Technology Facilities Council

Preprint

RAL-P-2013-006

# Projected Krylov methods for saddle-point systems

NIM Gould, D Orban, T Rees

April 2013

Submitted for publication in SIAM Journal on Matrix Analysis and Applications

RAL Library
STFC Rutherford Appleton Laboratory
R61
Harwell Oxford
Didcot
OX11 0QX

Tel: +44(0)1235 445384
Fax: +44(0)1235 446403
email: libraryral@stfc.ac.uk



**ISSN 1361- 4762**

# Projected Krylov methods for saddle-point systems

Nicholas I. M. Gould,[1,2] Dominique Orban[3,4] and Tyrone Rees[1,2]

**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.

---

[1] Scientific Computing Department, Rutherford Appleton Laboratory,
Chilton, Oxfordshire, OX11 0QX, England, EU.
Email: nick.gould@stfc.ac.uk , tyrone.rees@stfc.ac.uk .
Current reports available from "http://www.numerical.rl.ac.uk/reports/reports.shtml".

[2] This work was supported by the EPSRC grant EP/I013067/1.

[3] GERAD and Department of Mathematics and Industrial Engineering École Polytechnique de Montréal, C. P. 6079, Succ. Centre Ville Montréal QC H3C 3A7, Canada.
Email : dominique.orban@gerad.ca .

[4] This work was partially supported by an NSERC Discovery Grant.

Scientific Computing Department
Rutherford Appleton Laboratory
Oxfordshire OX11 0QX

April 10, 2013

# 1 Introduction

We consider the solution of the saddle-point problem

$$\begin{bmatrix} \boldsymbol{Q} & \boldsymbol{A}^{\mathsf{T}} \\ \boldsymbol{A} & \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_* + \boldsymbol{x}_{\mathrm{F}} \\ \boldsymbol{y}_* \end{bmatrix} = \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{b} \end{bmatrix} \tag{1}$$

for some fixed $\boldsymbol{x}_{\mathrm{F}}$, where all data is real and $\boldsymbol{Q}$ 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). 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 $\boldsymbol{A}$, 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. For this we only require that $\boldsymbol{Q}$ be available as an operator but it must be possible to compute projections into the nullspace of $\boldsymbol{A}$ 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) 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 $\boldsymbol{A}$ 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 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). In particular, we do not require that a basis for 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) 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 $\boldsymbol{Q}$ is symmetric, (1) may be interpreted as the first-order optimality conditions of the quadratic program

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

where $\boldsymbol{y}$ are the Lagrange multipliers associated with the equality constraints. It is well known, see, e.g. Gould (1985), that (2) possesses a unique solution if and only if $\boldsymbol{Q}$ is positive definite on the nullspace of $\boldsymbol{A}$. Based on this, Gould et al. (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 et al. (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, 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) 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 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 $\boldsymbol{Q}$, $\boldsymbol{A}$, $\boldsymbol{x}$, etc., of (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 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., $\boldsymbol{Q}$, $\boldsymbol{A}$, $\boldsymbol{x}$. 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) *is nonsingular.*

Benzi et al. (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 semi-definite and $\boldsymbol{A}$ has full row rank, then

- (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) is nongingular,

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 $\boldsymbol{A}$ 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). In §5, 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), $\boldsymbol{x}_*$ and $\boldsymbol{y}_*$ satisfy

$$\begin{bmatrix} \boldsymbol{Q} & \boldsymbol{A}^{\mathsf{T}} \\ \boldsymbol{A} & \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_* \\ \boldsymbol{y}_* \end{bmatrix} = \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{0} \end{bmatrix} \tag{3}$$

where $\boldsymbol{c} := \boldsymbol{a} - \boldsymbol{Q}\boldsymbol{x}_{\text{F}}$. Our aim is thus to solve (3).

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

$$\boldsymbol{A}\boldsymbol{Z} = \boldsymbol{0}. \tag{4}$$

Then necessarily from the second block of (3), $\boldsymbol{x}_* = \boldsymbol{Z}x$ 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 (4),

$$Qx_* = c \qquad \text{where} \quad Q := \boldsymbol{Z}^{\mathsf{T}}\boldsymbol{Q}\boldsymbol{Z} \quad \text{and} \quad c := \boldsymbol{Z}^{\mathsf{T}}\boldsymbol{c}. \tag{5}$$

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 (4) that residuals are transformed according to

$$r = \boldsymbol{Z}^{\mathsf{T}}\boldsymbol{r}. \tag{6}$$

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

$$L^{-1}QL^{-T}(L^T x_*) = L^{-1}c \tag{7}$$

of (5). If we define

$$\boldsymbol{Q} := L^{-1}QL^{-T}, \quad \boldsymbol{c} := L^{-1}c \quad \text{and} \quad \boldsymbol{x}_* = L^{-T}x_*, \tag{8}$$

the system (7) may be expressed compactly as

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

Here, residuals are transformed according to

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

## 2.1 Krylov-Subspace Methods

Let $\mathcal{S}_{\text{L}}$ and $\mathcal{S}_{\text{R}}$ be given subspaces of equal dimension, say $k$, and let the columns of $\boldsymbol{S}_{\text{L}}$ and $\boldsymbol{S}_{\text{R}}$ be bases for these spaces. We say that a subspace approximation $\boldsymbol{x}$ to the solution $\boldsymbol{x}_*$ of the generic linear system $\boldsymbol{Q}\boldsymbol{x}_* = \boldsymbol{c}$ is a *Petrov-Galerkin* approximation (Bruaset, 1995, Saad, 2003, Ch. 6 & 7, van der Vorst, 2003, Ch. 4) if

$$\boldsymbol{x} = \boldsymbol{S}_{\text{R}}\boldsymbol{z} \quad \text{where} \quad \boldsymbol{S}_{\text{L}}^T\boldsymbol{Q}\boldsymbol{S}_{\text{R}}\boldsymbol{z} = \boldsymbol{S}_{\text{L}}^T\boldsymbol{c},$$

and we see that such methods find $\boldsymbol{x} \in \mathcal{S}_{\text{R}}$ such that

$$\boldsymbol{r} = \boldsymbol{c} - \boldsymbol{Q}\boldsymbol{x} \perp \mathcal{S}_{\text{L}}.$$

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 $\mathcal{S}$ of dimension $k$, and an associated matrix $\boldsymbol{S}$ whose columns form a basis of $\mathcal{S}$. If we set $\mathcal{S}_{\text{R}} = \mathcal{S}_{\text{L}} = \mathcal{S}$, then we get what is known as a *(Ritz-)Galerkin* method.

An alternative is to set $\mathcal{S}_{\text{L}} := \boldsymbol{Q}\mathcal{S}$, $\mathcal{S}_{\text{R}} := \mathcal{S}$, say, which gives a *minimum-residual* approximation. Here

$$\boldsymbol{S}^T\boldsymbol{Q}^T\boldsymbol{Q}\boldsymbol{S}\boldsymbol{z} = \boldsymbol{S}^T\boldsymbol{Q}^T\boldsymbol{c},$$

or equivalently

$$\boldsymbol{x} = \boldsymbol{S}\boldsymbol{z} \quad \text{where} \quad \boldsymbol{z} = \arg\min_z \ \|\boldsymbol{Q}\boldsymbol{S}z - \boldsymbol{c}\|.$$

Finally, if we set $\mathcal{S}_{\text{L}} = \boldsymbol{Q}^T\mathcal{S}$ and $\mathcal{S}_{\text{R}} = \mathcal{S}$, then the equation we wish to satisfy becomes

$$\boldsymbol{S}^T\boldsymbol{Q}\boldsymbol{Q}^T\boldsymbol{S}\boldsymbol{z} = \boldsymbol{S}^T\boldsymbol{c},$$

and hence we get a *minimum-error* approximation, where

$$\boldsymbol{x} = \boldsymbol{Q}^{\mathsf{T}}\boldsymbol{S}\boldsymbol{z} \quad \text{where} \quad \boldsymbol{z} = \arg\min_{z} \ \|\boldsymbol{Q}^{\mathsf{T}}\boldsymbol{S}z - \boldsymbol{x}_*\|.$$

Note that in all of these special examples of Petrov-Galerkin methods we choose the spaces such that $\mathcal{S}_{\text{L}} = \boldsymbol{B}\mathcal{S}_{\text{R}}$ for some matrix $\boldsymbol{B}$. Such processes are termed *balanced projection methods* (Bruaset, 1995, §3.1).

Given a trial space $\mathcal{S}_{\text{R}}$, we can now apply one of the recipes above to construct a test space $\mathcal{S}_{\text{L}}$ 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 $k$,

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

for specified $\boldsymbol{d}$ for which $\langle \boldsymbol{c}, \boldsymbol{d}\rangle \neq 0$. We focus on the aforementioned approximations to $\boldsymbol{x}_*$ as $k$ increases.

For reasons of numerical stability we prefer orthogonal bases for $\mathcal{K}_k$ and $\mathcal{K}_k^{\mathsf{T}}$ if possible, and may generate them by the Arnoldi (1951) process. In particular, let $\boldsymbol{V}_k^\circ$ and $\boldsymbol{W}_k^\circ$ be orthogonal basis matrices of $\mathcal{K}$ and $\mathcal{K}^{\mathsf{T}}$ respectively. Then the corresponding columns $\boldsymbol{v}_i^\circ$ and $\boldsymbol{w}_i^\circ$ satisfy

$$\boldsymbol{Q}\boldsymbol{V}_k^\circ = \boldsymbol{V}_{k+1}^\circ H_{k+1,k} \quad \text{and} \quad \boldsymbol{Q}^{\mathsf{T}}\boldsymbol{W}_k^\circ = \boldsymbol{W}_{k+1}^\circ R_{k+1,k}, \tag{9}$$

where the $k+1$ by $k$ matrices $H_{k+1,k}$ and $R_{k+1,k}$ are upper Hessenberg, $\boldsymbol{v}_1^\circ = \boldsymbol{c}/\|\boldsymbol{c}\|$ and $\boldsymbol{w}_1^\circ = \boldsymbol{d}/\|\boldsymbol{d}\|$.

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

$$\boldsymbol{Q}\boldsymbol{V}_k = \boldsymbol{V}_{k+1}T_{k+1,k}, \quad \boldsymbol{W}_k^{\mathsf{T}}\boldsymbol{V}_k = D_k, \quad \text{and} \quad \boldsymbol{W}_k^{\mathsf{T}}\boldsymbol{Q}\boldsymbol{V}_k = D_kT_{k,k}, \tag{10}$$

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 $\boldsymbol{Q}$ is symmetric, breakdown will not occur if $\boldsymbol{c} = \boldsymbol{d}$ and then $\boldsymbol{V}_k = \boldsymbol{W}_k$, $D_k = I$ and $T_{k,k}$ is symmetric (van der Vorst, 2003, §7.1).

With the Galerkin approximation approach, we may choose $\mathcal{S}_k = \mathcal{K}_k$ and use (9) in which case

$$\boldsymbol{x}_k = \boldsymbol{V}_k^\circ z_k \quad \text{where} \quad H_{k,k}z_k = \beta_1 e_1, \tag{11}$$

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

Equally, we may instead use a Petrov-Galerkin approach with $\mathcal{S}_{\text{L}} = \mathcal{K}_k^{\mathsf{T}}$ and $\mathcal{S}_{\text{R}} = \mathcal{K}_k$ and use (10), which gives

$$\boldsymbol{x}_k = \boldsymbol{V}_k z_k \quad \text{where} \quad T_{k,k}z_k = \beta_1 e_1 \tag{12}$$

and is the basic Bi-CG method (Fletcher, 1976) in the unsymmetric case and equivalent to CG (Hestenes and Stiefel, 1952) when $\boldsymbol{Q}$ is symmetric. FOM requires that all of $\boldsymbol{V}_k$ be

stored, while Bi-CG/CG only require recent $\boldsymbol{v}_i$—a so-called short-term recurrence—since the solution to (12) 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 $\boldsymbol{Q}$ and its transpose, this is not necessarily the case. In particular both the basis matrix $\boldsymbol{V}_k$ and tridiagonal $T_{k,k}$ may be found by interlaced pairs of products with $\boldsymbol{Q}$, and this leads to the CGS method (Sonneveld, 1989).

For the minimum-residual approximation approach, choosing $\mathcal{S}_k = \mathcal{K}_k$ and using (9) gives

$$\boldsymbol{x}_k = \boldsymbol{V}_k^\circ z_k \quad \text{where} \quad H_{k+1,k}^{\mathsf{T}} H_{k+1,k} z_k = \beta_1 H_{k+1,k}^{\mathsf{T}} e_1, \tag{13}$$

which gives the GMRES method (Saad and Schultz, 1986). This simplifies when $\boldsymbol{Q}$ 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 $\boldsymbol{v}_i$. Alternatively, one can simply replace $\boldsymbol{V}_k^\circ$ by the non-orthogonal $\boldsymbol{V}_k$ from (10) and compute

$$\boldsymbol{x}_k = \boldsymbol{V}_k z_k \quad \text{where} \quad T_{k+1,k}^{\mathsf{T}} T_{k+1,k} z_k = \beta_1 T_{k+1,k}^{\mathsf{T}} e_1; \tag{14}$$

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 $\mathcal{S}_k = \mathcal{K}_k^{\mathsf{T}}$ with $\boldsymbol{d} = \boldsymbol{c}$ and using (9) gives

$$\boldsymbol{x}_k = \boldsymbol{W}_k^\circ z_k \quad \text{where} \quad z_k = R_{k+1,k} p_k \quad \text{and} \quad R_{k+1,k}^{\mathsf{T}} R_{k+1,k} p_k = \beta_1 e_1;$$

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

$$\boldsymbol{x}_k = \boldsymbol{V}_k^\circ z_k \quad \text{where} \quad z_k = T_{k+1,k} p_k \quad \text{and} \quad T_{k+1,k}^{\mathsf{T}} T_{k+1,k} p_k = \beta_1 e_1; \tag{15}$$

Notice that each of the possibilities (11)–(15) depends entirely on its basis matrix $\boldsymbol{V}_k^\circ$ or $\boldsymbol{V}_k$, 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

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

or, in terms of the un-preconditioned variables,

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

where

$$V_k^\circ = L^{-\mathsf{T}} \boldsymbol{V}_k^\circ \quad \text{and} \quad V_k = L^{-\mathsf{T}} \boldsymbol{V}_k \tag{16}$$

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

$$z_k = \Theta_k(T_{k+1,k}, \beta_1) \ \text{ or } \ z_k = \Theta_k(H_{k+1,k}, \beta_1), \tag{17}$$

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 $\boldsymbol{Q}$ as given by (8). 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 (3).

### 2.2.1 Orthogonal Bases and the Arnoldi Process

The Arnoldi process to compute $\boldsymbol{V}_k^\circ$ and $H_{k,k-1}$ for $\boldsymbol{Q}$ and $\boldsymbol{c}$ from (8) may be summarised as Algorithm 2.1.

**Algorithm 2.1** Arnoldi Process for $\boldsymbol{V}_k^\circ$ and $\boldsymbol{H}_{k,k-1}$

**Require:** $\boldsymbol{Q}$, $\boldsymbol{c}$ and $\boldsymbol{x}_0$
Set $\boldsymbol{v}_1^\circ = \boldsymbol{c} - \boldsymbol{Q}\boldsymbol{x}_0$ ▷ Initial Krylov vector
$\boldsymbol{h}_{1,0} = \sqrt{\langle \boldsymbol{v}_1^\circ, \boldsymbol{v}_1^\circ \rangle}$ ▷ Initial residual norm
**if** $\boldsymbol{h}_{1,0} \neq 0$ **then**
    $\boldsymbol{v}_1^\circ = \boldsymbol{v}_1^\circ / \boldsymbol{h}_{1,0}$
$k = 1$
**while** $\boldsymbol{h}_{k,k-1} \neq 0$ **do**
    $\boldsymbol{v}_{k+1}^\circ = \boldsymbol{Q}\boldsymbol{v}_k^\circ$ ▷ Compute next Krylov vector
    **for** $i = 1, \ldots, k$ **do** ▷ Modified Gram-Schmidt
        $\boldsymbol{h}_{i,k} = \langle \boldsymbol{v}_i^\circ, \boldsymbol{v}_{k+1}^\circ \rangle$
        $\boldsymbol{v}_{k+1}^\circ = \boldsymbol{v}_{k+1}^\circ - \boldsymbol{h}_{i,k}\boldsymbol{v}_i^\circ$
    $\boldsymbol{h}_{k+1,k} = \sqrt{\langle \boldsymbol{v}_{k+1}^\circ, \boldsymbol{v}_{k+1}^\circ \rangle}$ ▷ Residual norm
    **if** $\boldsymbol{h}_{k+1,k} \neq 0$ **then**
        $\boldsymbol{v}_{k+1}^\circ = \boldsymbol{v}_{k+1}^\circ / \boldsymbol{h}_{k+1,k}$
    $k = k + 1$

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

$$\boldsymbol{Q}\boldsymbol{V}_k = \boldsymbol{V}_{k+1}\boldsymbol{H}_{k+1,k} \tag{18}$$

$$= \boldsymbol{V}_k\boldsymbol{H}_k + \boldsymbol{h}_{k+1,k}\boldsymbol{v}_{k+1}e_k^T, \tag{19}$$

where $\boldsymbol{H}_k$ is $k$-by-$k$ upper Hessenberg and $\boldsymbol{H}_{k+1,k}$ is $\boldsymbol{H}_k$ with the extra row $\boldsymbol{h}_{k+1,k}e_k^T$.

In order to formulate corresponding algorithm involving un-preconditioned quantities, we apply changes of variables suggested by (8) and (16). All transformations in the next few sections will follow the same principle.

**Principle 2.1** *1. Basis vectors transform according to $v_k^\circ = L^{-T}\boldsymbol{v}_k^\circ$. Because of (8), each assignment 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 $v_k^\circ$ by solving the preconditioning system $Gv_k^\circ = u$ for $v_k^\circ$.*

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* (8)*, 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.

**Algorithm 2.2** Preconditioned Arnoldi Process for $V_k^\circ$ and $H_{k,k-1}$

**Require:** $Q$, $G = G^{\mathsf{T}} \succ 0$, $c$ and $x_0$
$u = c - Qx_0$
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** $i = 1, \ldots, k$ **do** ▷ Modified Gram-Schmidt
        $h_{i,k} = \langle u, v_i^\circ \rangle$
        $v_{k+1}^\circ = v_{k+1}^\circ - h_{i,k} v_i^\circ$
    $h_{k+1,k} = \sqrt{\langle u, v_{k+1}^\circ \rangle}$ ▷ Preconditioned residual norm
    **if** $h_{k+1,k} \neq 0$ **then**
        $v_{k+1}^\circ = v_{k+1}^\circ / h_{k+1,k}$
    $k = k + 1$

When $\boldsymbol{Q}$ 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 $\boldsymbol{Q}$ and $\boldsymbol{c}$, 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 $\boldsymbol{V}_k$, $\boldsymbol{W}_k$ and the corresponding $\boldsymbol{T}_{k,k}$ as described in (10). When applied to $\boldsymbol{Q}$ and $\boldsymbol{c}$, 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)).

Upon defining the (unsymmetric) tridiagonal matrix

$$\boldsymbol{T}_k := \begin{bmatrix} \boldsymbol{t}_{1,1} & \boldsymbol{t}_{1,2} & & & \\ 1 & \boldsymbol{t}_{2,2} & \boldsymbol{t}_{2,3} & & \\ & 1 & \boldsymbol{t}_{3,3} & \ddots & \\ & & \ddots & \ddots & \boldsymbol{t}_{k-1,k} \\ & & & 1 & \boldsymbol{t}_{k,k} \end{bmatrix}, \tag{20}$$

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

$$\boldsymbol{Q}\boldsymbol{V}_k = \boldsymbol{V}_k\boldsymbol{T}_k + \boldsymbol{v}_{k+1}e_k^T \tag{21a}$$
$$\boldsymbol{Q}^T\boldsymbol{W}_k = \boldsymbol{W}_k\boldsymbol{T}_k^T + \boldsymbol{t}_{k,k+1}\boldsymbol{w}_{k+1}e_k^T. \tag{21b}$$

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

**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 $x_0$
Set $v_0 = w_0 = 0$, $t_{0,1} = t_{1,0} = 1$
Set $s = c - Qx_0$. Set $w_1$ such that $\langle s, w_1\rangle = 1$.
Solve $Gv_1 = s$ for $v_1$ ▷ Initial Krylov vectors
$k = 1$
**while** $t_{k-1,k} \neq 0$ **do**
    $s = Qv_k$ and $u = Q^Tw_k$
    Solve $Gv_{k+1} = s$ and $Gw_{k+1} = u$ ▷ Compute next Krylov vectors
    $t_{k,k} = \langle s, w_k\rangle$
    $v_{k+1} = v_{k+1} - t_{k,k}v_k - t_{k-1,k}v_{k-1}$
    $w_{k+1} = w_{k+1} - t_{k,k}w_k - w_{k-1}$ ▷ Bi-orthogonalization
    $t_{k,k+1} = \langle s, w_{k+1}\rangle$
    **if** $t_{k,k+1} \neq 0$ **then**
        $w_{k+1} = w_{k+1}/t_{k,k+1}$
    $t_{k+1,k} = 1$
    $k = k + 1$

An alternative to Algorithm A.3 is the variant given by Chan et al. (1998) and Freund et al. (1993). When applied to $\boldsymbol{Q}$ and $\boldsymbol{c}$ from (8) 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 $\boldsymbol{Q}^T$ altogether.

In Algorithm A.3, the choice of off-diagonal entries of $\boldsymbol{T}_{k+1,k+1}$ was arbitrary; any pair $\boldsymbol{t}_{k+1,k}$ and $\boldsymbol{t}_{k,k+1}$ satisfying $\boldsymbol{t}_{k+1,k}\boldsymbol{t}_{k,k+1} = \langle \boldsymbol{v}_{k+1}, \boldsymbol{w}_{k+1}\rangle$ 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 $\boldsymbol{v}_{k+1}$ and $\boldsymbol{w}_{k+1}$ have the same form, and that the process is characterized by the identities

$$\boldsymbol{Q}\boldsymbol{V}_k = \boldsymbol{V}_k\boldsymbol{T}_k + \boldsymbol{v}_{k+1}e_k^T \tag{22a}$$
$$\boldsymbol{Q}^T\boldsymbol{W}_k = \boldsymbol{W}_k\boldsymbol{T}_k + \boldsymbol{w}_{k+1}e_k^T, \tag{22b}$$

in which there are no occurences of $\boldsymbol{T}_k^T$. It is not difficult to establish that Algorithm A.4 indeed generates bi-orthogonal vectors. The line of proof is identical to that of Saad (2003, Proposition 7.1).

**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 (22) and Theorem 2.1, we have, in exact arithmetic,

$$\boldsymbol{W}_k^T \boldsymbol{Q} \boldsymbol{V}_k = \boldsymbol{V}_k \boldsymbol{Q}^T \boldsymbol{W}_k = \boldsymbol{D}_k \boldsymbol{T}_k, \qquad \boldsymbol{D}_k = \operatorname{diag}(\boldsymbol{\delta}_1, \ldots, \boldsymbol{\delta}_k),$$

and $\boldsymbol{D}_k \boldsymbol{T}_k$ is clearly still tridiagonal.

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

**Algorithm 2.4** Preconditioned Variant of the Lanczos Bi-Orthogonalization Process for $V_k$, $W_k$ and $T_k$

**Require:** $Q$, $Q^T$, $G = G^T \succ 0$, $c$ and $x_0$

1: Set $v_0 = w_0 = 0$ 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** $\delta_k \neq 0$ **do**
6: $s = Qv_k$ 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: $k = k + 1$

Note that, as with the Arnoldi process, when $\boldsymbol{Q}$ 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 $\boldsymbol{T}_{k+1,k}$ (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 $w = G^{-1} w_G$.

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

**Require:** $Q$, $G = G^\mathsf{T} \succ 0$, $c$ 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** $\delta_k \neq 0$ **do**
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: $v_{k+1} = s - t_{k,k} d + t_{k-1,k}^2 v_{k-1}$ ▷ Compute next Krylov vector
10: $\delta_{k+1} = \langle v_{k+1}, w_G \rangle$
11: $k = k + 1$

## 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 $\boldsymbol{A}$ may actually be applied in the original (full) space. 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.

**Requirement 2.1** *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.

### 2.3.1 Orthogonal-Basis Methods

Consider first 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 2.2 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 (5), each assignment of $v_k^\circ$ 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* $\langle \boldsymbol{Z}^T \boldsymbol{u}, (\boldsymbol{Z}^T \boldsymbol{G} \boldsymbol{Z})^{-1} \boldsymbol{Z}^T \boldsymbol{u} \rangle = \langle \boldsymbol{u}, \boldsymbol{v}_k^\circ \rangle$.

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

$$\boldsymbol{P}_G := \boldsymbol{Z}(\boldsymbol{Z}^T \boldsymbol{G} \boldsymbol{Z})^{-1} \boldsymbol{Z}^T \tag{23}$$

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 2.2** *Let* $\boldsymbol{P}_G$ *be defined as in* (23) *where* $\boldsymbol{G}$ *satisfies Requirement 2.1. 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* $\boldsymbol{x} \in \text{Range}(\boldsymbol{A}^T)$
3. $\boldsymbol{P}_G \boldsymbol{G} x = x$ *for all* $x \in \text{Null}(\boldsymbol{A})$
4. $\boldsymbol{Z}^T \boldsymbol{G} \boldsymbol{P}_G = \boldsymbol{Z}^T$ *and* $\boldsymbol{Z}^T \boldsymbol{G}(\boldsymbol{I} - \boldsymbol{P}_G \boldsymbol{G}) = \boldsymbol{0}$
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 et al., 2001) that $\boldsymbol{v}_k^\circ = \boldsymbol{P}_G \boldsymbol{u}$ may instead be computed by solving the *constraint preconditioned* (Keller et al., 2000) saddle-point system

$$\begin{bmatrix} \boldsymbol{G} & \boldsymbol{A}^T \\ \boldsymbol{A} & \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_k^\circ \\ \boldsymbol{y}_k \end{bmatrix} = \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{0} \end{bmatrix}. \tag{24}$$

For the coefficient matrix $\boldsymbol{K}_G$ of (24) to be a constraint preconditioner, all that is needed is that $\boldsymbol{G}$ satisfies Requirement 2.1 or, equivalently, that $\boldsymbol{K}_G$ be nonsingular and have precisely as many negative eigenvalues as $\boldsymbol{A}$ 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)—$\boldsymbol{K}_G$; this has the further advantage that the required initial value $\boldsymbol{x}_{\text{F}}$ may be found by solving

$$\begin{bmatrix} \boldsymbol{G} & \boldsymbol{A}^T \\ \boldsymbol{A} & \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{\text{F}} \\ \boldsymbol{y}_0 \end{bmatrix} = \begin{bmatrix} \boldsymbol{g} \\ \boldsymbol{b} \end{bmatrix} \tag{25}$$

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

Applying Principle 2.2 to Algorithm 2.2, we derive the class of Krylov methods for (3) 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 (3)

**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
        $\boldsymbol{h}_{i,k} = \langle \boldsymbol{v}_i, \boldsymbol{u}\rangle$
        $v_{k+1} = \boldsymbol{v}_{k+1} - \boldsymbol{h}_{i,k}\boldsymbol{v}_i$
    $\boldsymbol{h}_{k+1,k} = \sqrt{\langle \boldsymbol{v}_{k+1}, \boldsymbol{u}\rangle}$ ▷ Preconditioned residual norm
    **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
    $k = k + 1$

Observe that Algorithm 2.6 (and those that will follow) only aim to find $\boldsymbol{x}_*$ satisfying (3) 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

$$\boldsymbol{y}_* = \arg\min_{\boldsymbol{y}\in\mathbb{R}^m} \|\boldsymbol{A}^\mathsf{T}\boldsymbol{y} - (\boldsymbol{c} - \boldsymbol{Q}\boldsymbol{x})\|_N \tag{26}$$

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 (24). The same is true using the dual to the semi-norm $\sqrt{\langle \boldsymbol{s}, \boldsymbol{G}\boldsymbol{s}\rangle}$ defined on the manifold $\boldsymbol{A}\boldsymbol{s} = \boldsymbol{0}$ under the more general Requirement 2.1 on $\boldsymbol{G}$ (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 $\boldsymbol{x}_k = \boldsymbol{Z}x_k = \boldsymbol{Z}V_k z_k$, 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 (3) based on Lanczos bi-orthogonalization may be stated as Algorithm 2.7. The variant based on Algorithm 2.3 appears as Algorithm A.6. Note

**Algorithm 2.7** Lanczos-Based Projected Krylov Methods for (3)

**Require:** $\boldsymbol{Q}$, $\boldsymbol{Q}^T$, $\boldsymbol{P}_G$, $\boldsymbol{c}$ and $\boldsymbol{x}_0$
1: Set $\boldsymbol{v}_0 = \boldsymbol{w}_0 = \boldsymbol{0}$ and $\boldsymbol{\delta}_0 = 1$
2: Set $\boldsymbol{s} = \boldsymbol{c} - \boldsymbol{Q}\boldsymbol{x}_0$ and $\boldsymbol{w}_1$ such that $\boldsymbol{\delta}_1 := \langle \boldsymbol{s}, \boldsymbol{w}_1 \rangle \neq 0$
3: Compute $\boldsymbol{v}_1 = \boldsymbol{P}_G \boldsymbol{s}$ for $\boldsymbol{v}_1$ ▷ Initial Krylov vectors
4: Compute the solution estimate $\boldsymbol{x}_1 = \boldsymbol{V}_1 \Theta_1(\boldsymbol{T}_{1,0}, \boldsymbol{t}_{1,0})$
5: $k = 1$
6: **while** $\boldsymbol{\delta}_k \neq 0$ **do**
7: $\quad \boldsymbol{s} = \boldsymbol{Q}\boldsymbol{v}_k$ and $\boldsymbol{u} = \boldsymbol{Q}^T \boldsymbol{w}_k$
8: $\quad \boldsymbol{v}_{k+1} = \boldsymbol{P}_G \boldsymbol{s}$ and $\boldsymbol{w}_{k+1} = \boldsymbol{P}_G \boldsymbol{u}$ ▷ Compute next Krylov vectors
9: $\quad \boldsymbol{t}_{k,k} = \langle \boldsymbol{s}, \boldsymbol{w}_k \rangle / \delta_k$
10: $\quad \boldsymbol{t}_{k-1,k} = \boldsymbol{\delta}_k / \boldsymbol{\delta}_{k-1}$
11: $\quad \boldsymbol{v}_{k+1} = \boldsymbol{v}_{k+1} - \boldsymbol{t}_{k,k} \boldsymbol{v}_k - \boldsymbol{t}_{k-1,k} \boldsymbol{v}_{k-1}$
12: $\quad \boldsymbol{w}_{k+1} = \boldsymbol{w}_{k+1} - \boldsymbol{t}_{k,k} \boldsymbol{w}_k - \boldsymbol{t}_{k-1,k} \boldsymbol{w}_{k-1}$ ▷ Bi-orthogonalization
13: $\quad \boldsymbol{\delta}_{k+1} = \langle \boldsymbol{s}, \boldsymbol{w}_{k+1} \rangle$
14: $\quad$ Compute the solution estimate $\boldsymbol{x}_{k+1} = \boldsymbol{V}_{k+1} \Theta_{k+1}(\boldsymbol{T}_{k+1,k}, \boldsymbol{t}_{1,0})$
15: $\quad k = k + 1$

that for each $k$, Algorithm 2.7 makes the implicit assignment $\boldsymbol{t}_{k+1,k} = 1$.

### 2.3.3 Transpose-Free Bi-Orthogonal Methods

Finally, we remain with iterates generated by $\boldsymbol{x}_k = \boldsymbol{Z} x_k = \boldsymbol{Z} V_k z_k$, 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 on the following page.

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 (3) with a constraint preconditioner of the form (24). 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.

**Algorithm 2.8** Transpose-Free Lanczos-Based Projected Krylov Methods for (3)

**Require:** $\boldsymbol{Q}$, $\boldsymbol{P}_G$, $\boldsymbol{c}$ and $\boldsymbol{x}_0$
1: Set $\boldsymbol{v}_0 = \boldsymbol{u}_0 = 0$ and $\boldsymbol{\delta}_0 = 1$
2: Set $\boldsymbol{y} = \boldsymbol{c} - \boldsymbol{Q}\boldsymbol{x}_0$ and compute $\boldsymbol{v}_1 = \boldsymbol{P}_G y$ ▷ Initial Krylov vector
3: Set $\boldsymbol{w}_G$ and $\boldsymbol{w} = \boldsymbol{P}_G(fsw_G$ such that $\boldsymbol{\delta}_1 := \langle \boldsymbol{v}_1, \boldsymbol{w}_G \rangle \neq 0$
4: Compute the solution estimate $\boldsymbol{x}_1 = \boldsymbol{V}_1 \Theta_1(\boldsymbol{T}_{1,0}, \boldsymbol{t}_{1,0})$
5: $k = 1$
6: **while** $\boldsymbol{\delta}_k \neq 0$ **do**
7: Compute $\boldsymbol{s} = \boldsymbol{P}_G(\boldsymbol{Q}\boldsymbol{v}_k)$, $\boldsymbol{t}_{k,k} = \langle \boldsymbol{s}, \boldsymbol{w}_G \rangle / \boldsymbol{\delta}_k$ and $\boldsymbol{t}_{k-1,k} = \boldsymbol{\delta}_k / \boldsymbol{\delta}_{k-1}$
8: $\boldsymbol{u}_k = \boldsymbol{s} - \boldsymbol{t}_{k,k}\boldsymbol{v}_k - \boldsymbol{t}_{k-1,k}\boldsymbol{u}_{k-1}$
9: Set $\boldsymbol{d} = \boldsymbol{u}_k - \boldsymbol{t}_{k-1,k}\boldsymbol{u}_{k-1}$ and compute $\boldsymbol{s} = \boldsymbol{P}_G(\boldsymbol{Q}\boldsymbol{d})$
10: $\boldsymbol{v}_{k+1} = \boldsymbol{s} - \boldsymbol{t}_{k,k}\boldsymbol{d} + \boldsymbol{t}_{k-1,k}^2 \boldsymbol{v}_{k-1}$
11: $\boldsymbol{\delta}_{k+1} = \langle \boldsymbol{v}_{k+1}, \boldsymbol{w}_G \rangle$
12: Compute the solution estimate $\boldsymbol{x}_{k+1} = \boldsymbol{V}_{k+1} \Theta_{k+1}(\boldsymbol{T}_{k+1,k}, \boldsymbol{t}_{1,0})$
13: $k = k + 1$

### 2.4.1 Orthogonal-Basis Methods

Suppose Algorithm 2.2 is initialized with a starting guess of the form $(\boldsymbol{x}_0, \boldsymbol{y}_0)$ for which $\boldsymbol{x}_0 \in \text{Null}(\boldsymbol{A})$. Line 1 of Algorithm 2.2 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 2.6 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 2.2. In particular, $v_{1,x}$ lies in the nullspace of $\boldsymbol{A}$. At line 3 of Algorithm 2.2, 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 2.6.

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 2.2 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 $\boldsymbol{u} + \boldsymbol{A}^T v_{k,y}$, where $\boldsymbol{u}$ is computed at line 9 of Algorithm 2.6. Next, $v_{k+1}$ is given as the solution of

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

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

Finally, Algorithm 2.2 computes

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

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

**Theorem 2.3** *Algorithm 2.2 applied to (3) with an initial guess of the form $(\boldsymbol{x}_0, \boldsymbol{y}_0)$ satisfying $\boldsymbol{A}\boldsymbol{x}_0 = \boldsymbol{0}$ and using the constraint preconditioner (24) generates Krylov vectors $v_k = (v_{k,x}, v_{k,y})$ such that at each iteration $k$, $v_{k,x}$ is the Krylov vector $\boldsymbol{v}_k$ generated by Algorithm 2.6 with initial guess $\boldsymbol{x}_0$ at iteration $k$. The temporary vector $u$ has the form $(\boldsymbol{u} + \boldsymbol{A}^T \boldsymbol{w}, \boldsymbol{0})$ for some vector $\boldsymbol{w}$, where $\boldsymbol{u}$ 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 $\boldsymbol{u}$ and $\boldsymbol{v}_k$ 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 $\boldsymbol{y}$ segment of the solution of (3). Depending on how the application of $\boldsymbol{P}_G$ 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 (24) is a well-defined Krylov method.

**Example: Projected GMRES** Standard GMRES performs $m$ iterations of Algorithm 2.1 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

$$\underset{\boldsymbol{z}_{m+1}}{\text{minimize}} \ \| \boldsymbol{h}_{1,0} e_1 - \boldsymbol{H}_{m+1,m} \boldsymbol{z}_{m+1} \|_2. \tag{27}$$

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 (27), 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.

### 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 (3) with the preconditioner (24). Suppose the latter is initialized with $c = (\boldsymbol{c}, \boldsymbol{0})$ and $x_0 = (\boldsymbol{x}_0, \boldsymbol{y}_0)$ such that $\boldsymbol{A}\boldsymbol{x}_0 = \boldsymbol{0}$. We first compute

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

During initialization, we thus have $v_{1,x} = \boldsymbol{P}_G(\boldsymbol{s} - \boldsymbol{A}^T \boldsymbol{y}_0) = \boldsymbol{P}_G(\boldsymbol{s}) = \boldsymbol{v}_1$. Suppose Algorithm 2.3 selects $w_1$ such that $\boldsymbol{A} w_{1,x} = 0$.

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

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

and

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

before solving the preconditioning systems

$$\begin{bmatrix} \boldsymbol{G} & \boldsymbol{A}^T \\ \boldsymbol{A} & \end{bmatrix} \begin{bmatrix} v_{k+1,x} \\ v_{k+1,y} \end{bmatrix} = \begin{bmatrix} s_x \\ \boldsymbol{0} \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \boldsymbol{G} & \boldsymbol{A}^T \\ \boldsymbol{A} & \end{bmatrix} \begin{bmatrix} w_{k+1,x} \\ w_{k+1,y} \end{bmatrix} = \begin{bmatrix} u_x \\ \boldsymbol{0} \end{bmatrix}.$$

In other words, $v_{k+1,x} = \boldsymbol{P}_G s_x = \boldsymbol{P}_G(\boldsymbol{s} + \boldsymbol{A}^T w_{k,y}) = \boldsymbol{P}_G \boldsymbol{s} = \boldsymbol{v}_{k+1}$. Similarly, $w_{k+1,x} = \boldsymbol{w}_{k+1}$. Moreover, $\boldsymbol{A} v_{k+1,x} = \boldsymbol{A} w_{k+1,x} = 0$ before the computation of $t_{k,k}$.

In Algorithm 2.3,

$$t_{k,k} = \langle s, w_k \rangle = \langle s_x, w_{k,x} \rangle = \langle \boldsymbol{s} + \boldsymbol{A}^T w_{k,y}, \boldsymbol{w}_k \rangle = \langle \boldsymbol{s}, \boldsymbol{w}_k \rangle,$$

which is the same $\boldsymbol{t}_{k,k}$ computed by Algorithm A.6. Similarly, we see that the two algorithms compute the same $\boldsymbol{t}_{k+1,k}$. By recursion, we have established the following result.

**Theorem 2.4** *Algorithm 2.3 applied to (3) with an initial guess of the form $(\boldsymbol{x}_0, \boldsymbol{y}_0)$ satisfying $\boldsymbol{A}\boldsymbol{x}_0 = \boldsymbol{0}$ and using the constraint preconditioner (24) 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 $\boldsymbol{v}_k$ and $\boldsymbol{w}_k$ generated at iteration $k$ of Algorithm A.6 with initial guess $\boldsymbol{x}_0$ and using $\boldsymbol{w}_1 = w_{1,x}$ chosen so that $\boldsymbol{A}\boldsymbol{w}_1 = 0$. The temporary vectors $s$ and $u$ have the form $(\boldsymbol{s} + \boldsymbol{A}^T v_{k,y}, \boldsymbol{0})$ and $(\boldsymbol{u} + \boldsymbol{A}^T w_{k,y}, \boldsymbol{0})$, where $\boldsymbol{s}$ and $\boldsymbol{u}$ 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 §2.4.2, 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 Rozložník and Simoncini (2002), who restrict their attention to the case where $\boldsymbol{Q}$ is symmetric and positive definite.

## 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 $c = \boldsymbol{Z}^T\boldsymbol{c}$, $Q = \boldsymbol{Z}^T\boldsymbol{Q}\boldsymbol{Z}$ and $G = \boldsymbol{Z}^T\boldsymbol{G}\boldsymbol{Z}$ is positive definite. The quantity minimized by the projected MINRES at each iteration may thus be written

$$\|r_k\|^2_{G^{-1}} = \langle r_k, (\boldsymbol{Z}^T\boldsymbol{G}\boldsymbol{Z})^{-1} r_k \rangle. \tag{28}$$

We now establish directly that standard MINRES applied with a constraint preconditioner also minimizes (28), and therefore that MINRES with the indefinite preconditioner (24) is a well-defined Krylov method.

Theorem 2.3 guarantees that with appropriate initial conditions, the approximate solution $(\boldsymbol{x}_k, \boldsymbol{y}_k)$ generated at the $k$-th iteration of the constraint-preconditioned MINRES is such that $\boldsymbol{x}_k \in \text{Null}(\boldsymbol{A})$. Consider the residual

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

It is tempting to claim that it is this residual that is minimized in the norm defined by the inverse of the preconditioner (24). Unfortunately, the preconditioner is indefinite. The appropriate interpretation of this claim is to consider the *seminorm* associated to the preconditioner and defined by

$$\|\boldsymbol{r}_k\|_{[\boldsymbol{G}]}^2 := \langle \boldsymbol{r}_k, \boldsymbol{s}_k \rangle \quad \text{where} \quad \begin{bmatrix} \boldsymbol{G} & \boldsymbol{A}^{\mathsf{T}} \\ \boldsymbol{A} & \end{bmatrix} \begin{bmatrix} \boldsymbol{s}_k \\ \boldsymbol{t}_k \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}_k \\ \boldsymbol{0} \end{bmatrix}, \tag{29}$$

i.e.,

$$\|\boldsymbol{r}_k\|_{[\boldsymbol{G}]}^2 = \langle \boldsymbol{c} - \boldsymbol{Q}\boldsymbol{x}_k - \boldsymbol{A}^{\mathsf{T}}\boldsymbol{y}_k, \boldsymbol{s}_k \rangle = \langle \boldsymbol{c} - \boldsymbol{Q}\boldsymbol{x}_k, \boldsymbol{s}_k \rangle. \tag{30}$$

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 (29) that $\boldsymbol{s}_k \in \text{Null}(\boldsymbol{A})$ and therefore, $\boldsymbol{s}_k = \boldsymbol{Z}s_k$ for some $s_k$. The first block equation of (29) 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 (30), we finally obtain

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

which coincides with (28). On substituting $r_k = \boldsymbol{Z}^{\mathsf{T}}\boldsymbol{r}_k$ in this last identity, we also see that

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

We conclude that the projected MINRES algorithm minimizes the Euclidian norm of the projected residual. 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 presente 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 $\boldsymbol{x}_k$ 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 non-negligible component in the range of $\boldsymbol{A}^{\mathsf{T}}$ 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 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:*

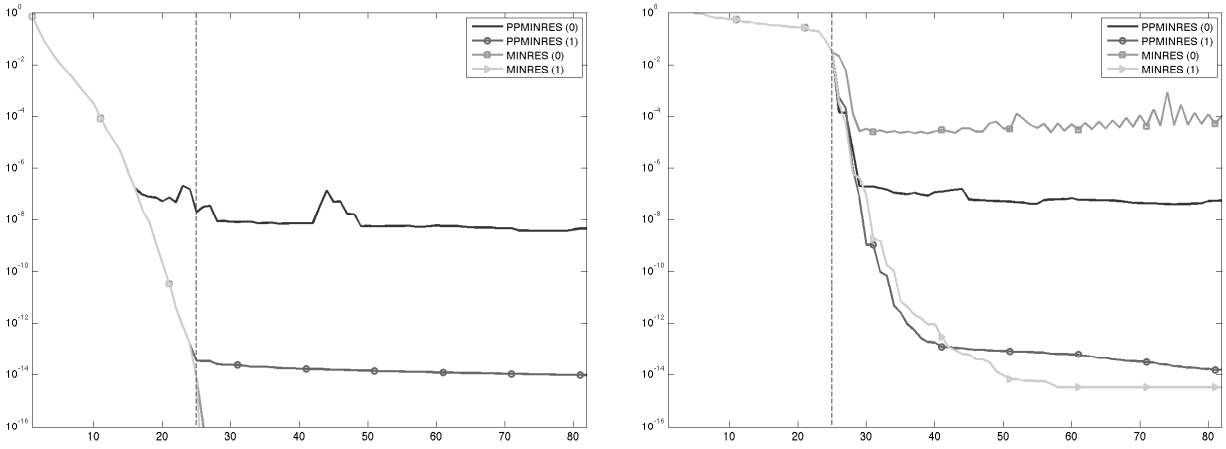
```
n = 100; m = 75;
Q = rand(n,n);
Q = Q + Q' + 5*eye(n);
A = rand(m,n);
K = [Q A'; A zeros(m,m)];
G = diag(abs(diag(Q)));
```

*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.




(a) Diagonally dominant leading block (b) Random symmetric leading block.

Figure 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.

From Figure 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 $\boldsymbol{Q}$, 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.

# 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 instuctive to consider the generalized eigenvalue problem

$$\begin{bmatrix} \boldsymbol{Q} & \boldsymbol{A}^\mathsf{T} \\ \boldsymbol{A} & \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix} = \lambda \begin{bmatrix} \boldsymbol{G} & \boldsymbol{A}^\mathsf{T} \\ \boldsymbol{A} & \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix}. \tag{31}$$

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 $\boldsymbol{Q}$ and $\boldsymbol{G}$, but this fact is not needed in the proof.

**Theorem 4.1** *Suppose that $\boldsymbol{A}$ is $m$-by-$n$ and has full row rank. Then the generalized eigenvalue problem (31) has*

- *an eigenvalue at 1 with multiplicity $2m$, and*
- *$n - m$ eigenvalues defined by the generalized eigenvalue problem*

$$\boldsymbol{Z}^\mathsf{T}\boldsymbol{Q}\boldsymbol{Z}x = \lambda \boldsymbol{Z}^\mathsf{T}\boldsymbol{G}\boldsymbol{Z}x.$$

This result is unsurprising, given the close relationship between solving (3) with a constraint preconditioner and solving the reduced problem (5) described in the preceeding sections.

When $\boldsymbol{Z}^\mathsf{T}\boldsymbol{Q}\boldsymbol{Z}$ 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 $(\boldsymbol{Q}, \boldsymbol{G})$. Fortunately, when $\boldsymbol{Z}^\mathsf{T}\boldsymbol{Q}\boldsymbol{Z}$ is symmetric, but possibly indefinite, we can say more. First, note that in this case—since Requirement 2.1 implies that $\boldsymbol{Z}^\mathsf{T}\boldsymbol{G}\boldsymbol{Z}$ 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 $\boldsymbol{Q}\boldsymbol{x} = \lambda \boldsymbol{G}\boldsymbol{x}$. Therefore,

provided that $\boldsymbol{G}$ is a good approximation to $\boldsymbol{Q}$, the eigenvalues satisfying (31) 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

$$t_{k+1,k} v_{k+1} = Q w_k - t_{k,k} v_k - t_{k-1,k} v_{k-1}, \tag{32a}$$
$$t_{k,k+1} w_{k+1} = Q^T v_k - t_{k,k} w_k - t_{k,k-1} w_{k-1}, \tag{32b}$$

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

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

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 $\langle v_j, Q w_k \rangle = \langle w_j, Q v_k \rangle = 0$ for $k < j - 1$ (Saunders et al., 1988, Theorem 1). The above relations differ considerably from (21) and (22) 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 (3) 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

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

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

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

The latter system has the form (1) 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

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

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

Finally, we note that it appears as if $\boldsymbol{A}$ must be available as an explicit matrix to factorize (24). There are other ways to compute projections. One of them is to solve a linear least-squares problem of the form (26) 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.

# 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

$$\begin{bmatrix} \boldsymbol{Q} & \boldsymbol{B}^{\mathsf{T}} \\ \boldsymbol{A} & \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_* \\ \boldsymbol{y}_* \end{bmatrix} = \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{0} \end{bmatrix}, \tag{33}$$

where $\boldsymbol{Q}$, as before, may be unsymmetric. Variants of the approach described in the present paper may be employed to solve (33). We leave the study of such variants for future work.

# References

W. E. Arnoldi. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quarterly of Applied Mathematics*, 9:1729, 1951.

M. Benzi, G. H. Golub, and J. Liesen. Numerical solution of saddle point problems. *Acta Numerica*, 14:1–137, 2005. DOI: 10.1017/S0962492904000212.

C. Brezinski and M. Redivo-Zaglia. Transpose-free Lanczos-type algorithms for non-symmetric linear systems. *Numerical Algorithms*, 17(1-2):67–103, 1998. DOI: 10.1023/A:1012085428800.

A. M. Bruaset. *A survey of preconditioned iterative methods*. Longman Scientific and Technical, Harlow, England, 1995.

Z. Cao. A note on constraint preconditioning for nonsymmetric indefinite matrices. *SIAM Journal on Matrix Analysis and Applications*, 24(1):121–125, 2002. DOI: 10.1137/S0895479801391424.

T. F. Chan, L. de Pillis, and H. van der Vorst. Transpose-free formulations of Lanczos-type methods for nonsymmetric linear systems. *Numerical Algorithms*, 17(1-2):51–66, 1998. DOI: 10.1023/A:1011637511962.

R. Chandra. *Conjugate gradient methods for partial differential equations*. PhD thesis, Yale University, New Haven, USA, 1978.

T. F. Coleman. Linearly constrained optimization and projected preconditioned conjugate gradients. In J. Lewis, editor, *Proceedings of the Fifth SIAM Conference on Applied Linear Algebra*, pages 118–122, Philadelphia, 1994. SIAM.

A. R. Conn, N. I. M. Gould, D. Orban, and Ph.L. Toint. A primal-dual trust-region algorithm for non-convex nonlinear programming. *Mathematical Programming B*, 87(2): 215–249, 2000. DOI: 10.1007/s101070050112.

H. S. Dollar and A. J. Wathen. Approximate factorization constraint preconditioners for saddle-point matrices. *SIAM J. Sci. Comput.*, 27(5):1555–1572, 2006. DOI: 10.1137/04060768X. URL `http://link.aip.org/link/?SCE/27/1555/1`.

H. S. Dollar, N. I. M. Gould, W. H. A. Schilders, and A. J. Wathen. Implicit-factorization preconditioning and iterative solvers for regularized saddle-point systems. *SIAM Journal on Matrix Analysis and Applications*, 28(1):170–189, 2007. DOI: 10.1137/05063427X.

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

R. Fletcher. Conjugate gradient methods for indefinite systems. In G. A. Watson, editor, *Numerical Analysis, Dundee 1975*, number 506 in Lecture Notes in Mathematics, page 7389, Heidelberg, Berlin, New York, 1976. Springer Verlag. DOI: 10.1007/BFb0080116.

R. Freund, M. Gutknecht, and N. Nachtigal. An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices. *SIAM Journal on Scientific Computing*, 14(1): 137–158, 1993. DOI: 10.1137/0914009.

R. W. Freund and N. M. Nachtigal. QMR: a quasi-minimal method for non-Hermitian linear systems. *Numerische Mathematik*, 60:315–339, 1991. DOI: 10.1007/BF01385726.

G. H. Golub and W. Kahan. Calculating the singular values and pseudo-inverse of a matrix. *SIAM Journal on Numerical Analysis*, 2(2):205–224, 1965. DOI: 10.1137/0702016.

G. H. Golub and C. F. van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, MD, third edition, 1996.

N. I. M. Gould. On practical conditions for the existence and uniqueness of solutions to the general equality quadratic-programming problem. *Mathematical Programming*, 32(1): 90–99, 1985. DOI: 10.1007/BF01585660.

N. I. M. Gould, M. E. Hribar, and J. Nocedal. On the solution of equality constrained quadratic problems arising in optimization. *SIAM Journal on Scientific Computing*, 23 (4):1375–1394, 2001. DOI: 10.1137/S1064827598345667.

A. Greenbaum. *Iterative Methods for Solving Linear Systems*. SIAM, Philadelphia, 1997.

A. Greenbaum, V. Pták, and Z. Strakoš. Any nonincreasing convergence curve is possible for GMRES. *SIAM Journal on Matrix Analysis and Applications*, 17(3):465–469, 1996. DOI: 10.1137/S0895479894275030.

M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49:409–436, 1952.

C. Keller, N. I. M. Gould, and A. J. Wathen. Constraint preconditioning for indefinite linear systems. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1300–1317, 2000. DOI: 10.1137/S0895479899351805.

C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of research of the National Bureau of Standards B*, 45:225–280, 1950.

L. Lukšan and J. Vlček. Indefinitely preconditioned inexact Newton method for large sparse equality constrained nonlinear programming problems. *Numerical Linear Algebra with Applications*, 5(3):219–247, 1998.

N. M. Nachtigal, S. C. Reddy, and L. N. Trefethen. How fast are nonsymmetric matrix iterations? *SIAM Journal on Matrix Analysis and Applications*, 13(3):778–795, 1992. DOI: 10.1137/0613049.

D. Orban. Projected Krylov methods for unsymmetric augmented systems. GERAD Technical Report G-2008-46, Mathematics and Industrial Engineering Department, Ecole Polytechnique de Montreal, 2008.

C. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, 12(4):617–629, 1975. DOI: 10.1137/0712047.

I. Perugia and V. Simoncini. Block-diagonal and indefinite symmetric preconditioners for mixed finite element formulations. *Numerical Linear Algebra with Applications*, 7(7-8): 585–616, 2000.

J. Pestana and A.J. Wathen. On choice of preconditioner for minimum residual methods for nonsymmetric matrices. Technical Report 10/07, Mathematical Institute, University of Oxford, 2011.

B. T. Polyak. The conjugate gradient method in extremal problems. *U.S.S.R. Computational Mathematics and Mathematical Physics*, 9:94–112, 1969.

M. Rozložník and V. Simoncini. Krylov subspace methods for saddle point problems with indefinite preconditioning. *SIAM Journal on Matrix Analysis and Applications*, 24(2): 368–391, 2002. DOI: 10.1137/S0895479800375540.

Y. Saad. Krylov subspace methods for solving large unsymmetric linear systems. *Mathematics of Computation*, 37:105–126, 1981. DOI: 10.1090/S0025-5718-1981-0616364-6.

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

Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7 (3):856–869, 1986. DOI: http://dx.doi.org/10.1137/0907058.

M. A. Saunders, H. D. Simon, and E. L. Yip. Two conjugate-gradient-type methods for unsymmetric linear equations. *SIAM Journal on Numerical Analysis*, 25(4):927–940, 1988. DOI: 10.1137/0725052.

P. Sonneveld. CGS, a fast Lanczos-type solver for nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 10(1):36–52, 1989. DOI: 10.1137/0910004.

H. A. van der Vorst. *Iterative Krylov methods for large linear systems*. Cambridge University Press, Cambridge, 2003.

# Appendix 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 $\boldsymbol{Q}$ is symmetric in Algorithm 2.1, $\boldsymbol{H}_{k+1,k}$ is tridiagonal and the resulting simplified procedure is the symmetric Lanczos process, summarised as Algorithm A.1.

**Algorithm A.1** Lanczos Process for $\boldsymbol{V}_k^\circ$ and $\boldsymbol{T}_{k-1,k}$

**Require:** $\boldsymbol{Q}$, $\boldsymbol{c}$ and $\boldsymbol{x}_0$
Set $\boldsymbol{v}_0^\circ = 0$ and $\boldsymbol{v}_1^\circ = \boldsymbol{c} - \boldsymbol{Q}\boldsymbol{x}_0$ ▷ Initial Krylov vector
$\boldsymbol{t}_{1,0} = \sqrt{\langle \boldsymbol{v}_1^\circ, \boldsymbol{v}_1^\circ \rangle}$ ▷ Initial residual norm
**if** $\boldsymbol{t}_{1,0} \neq 0$ **then**
    $\boldsymbol{v}_1^\circ = \boldsymbol{v}_1^\circ / \boldsymbol{t}_{1,0}$
$k = 1$
**while** $\boldsymbol{t}_{k,k-1} \neq 0$ **do**
    $\boldsymbol{v}_{k+1}^\circ = \boldsymbol{Q}\boldsymbol{v}_k^\circ$ ▷ Compute next Krylov vector
    $\boldsymbol{t}_{k,k} = \langle \boldsymbol{v}_k^\circ, \boldsymbol{v}_{k+1}^\circ \rangle$
    $\boldsymbol{v}_{k+1}^\circ = \boldsymbol{v}_{k+1}^\circ - \boldsymbol{t}_{k,k}\boldsymbol{v}_k^\circ - \boldsymbol{t}_{k,k-1}\boldsymbol{v}_{k-1}^\circ$ ▷ Modified Gram-Schmidt
    $\boldsymbol{t}_{k+1,k} = \sqrt{\langle \boldsymbol{v}_{k+1}^\circ, \boldsymbol{v}_{k+1}^\circ \rangle}$ ▷ Residual norm
    **if** $\boldsymbol{t}_{k+1,k} \neq 0$ **then**
        $\boldsymbol{v}_{k+1}^\circ = \boldsymbol{v}_{k+1}^\circ / \boldsymbol{t}_{k+1,k}$
    $k = k + 1$

Algorithm A.1 is characterized by the identities

$$\boldsymbol{Q}\boldsymbol{V}_k = \boldsymbol{V}_{k+1}\boldsymbol{T}_{k+1,k} \tag{34}$$

$$= \boldsymbol{V}_k\boldsymbol{T}_k + \boldsymbol{t}_{k+1,k}\boldsymbol{v}_{k+1}e_k^T, \tag{35}$$

where $\boldsymbol{T}_k$ is $k$-by-$k$ upper triangular and $\boldsymbol{T}_{k+1,k}$ is $\boldsymbol{T}_k$ with the extra row $\boldsymbol{t}_{k+1,k}e_k^T$.

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

**Algorithm A.2** Preconditioned Lanczos Process for $V_k^\circ$ and $T_{k-1,k}$

**Require:** $Q$, $G = G^T \succ 0$, $c$ and $x_0$

Set $v_0^\circ = 0$, $u = c - Qx_0$ and solve $Gv_1^\circ = u$ for $v_1^\circ$ ▷ Initial Krylov vector

$t_{1,0} = \sqrt{\langle u, v_1^\circ \rangle}$ ▷ Initial residual norm

**if** $t_{1,0} \neq 0$ **then**

    $v_1^\circ = v_1^\circ / t_{1,0}$

$k = 1$

**while** $t_{k,k-1} \neq 0$ **do**

    Set $u = Qv_k^\circ$ and solve $Gv_{k+1}^\circ = u$ for $v_{k+1}^\circ$ ▷ Compute next Krylov vector

    $t_{k,k} = \langle u, v_k^\circ \rangle$

    $v_{k+1}^\circ = v_{k+1}^\circ - t_{k,k} v_k^\circ - t_{k,k-1} v_{k-1}^\circ$ ▷ Modified Gram-Schmidt

    $t_{k+1,k} = \sqrt{\langle u, \boldsymbol{v}_{k+1}^\circ \rangle}$ ▷ Residual norm

    **if** $t_{k+1,k} \neq 0$ **then**

        $v_{k+1}^\circ = v_{k+1}^\circ / t_{k+1,k}$

    $k = k + 1$

## A.2 The Lanczos Bi-Orthogonalization Process

The Lanczos (1950) bi-orthogonalization process as described in (10) and applied to $\boldsymbol{Q}$ and $\boldsymbol{c}$ leads to Algorithm A.3. This is a special case of Saad (2003, Algorithm 7.1) and Golub and van Loan (1996, (9.4.7)).

**Algorithm A.3** Lanczos Bi-Orthogonalization Process for $\boldsymbol{V}_k$, $\boldsymbol{W}_k$ and $\boldsymbol{T}_k$

**Require:** $\boldsymbol{Q}$, $\boldsymbol{Q}^\mathsf{T}$, $\boldsymbol{c}$ and $\boldsymbol{x}_0$
Set $\boldsymbol{v}_0 = \boldsymbol{w}_0 = 0$, $\boldsymbol{t}_{0,1} = \boldsymbol{t}_{1,0} = 1$
Set $\boldsymbol{v}_1 = \boldsymbol{c} - \boldsymbol{Q}\boldsymbol{x}_0$ and $\boldsymbol{w}_1$ such that $\langle \boldsymbol{v}_1, \boldsymbol{w}_1 \rangle = 1$. ▷ Initial Krylov vectors
$k = 1$
**while** $\boldsymbol{t}_{k-1,k} \neq 0$ **do**
    $\boldsymbol{v}_{k+1} = \boldsymbol{Q}\boldsymbol{v}_k$ and $\boldsymbol{w}_{k+1} = \boldsymbol{Q}^\mathsf{T}\boldsymbol{w}_k$ ▷ Compute next Krylov vectors
    $\boldsymbol{t}_{k,k} = \langle \boldsymbol{v}_{k+1}, \boldsymbol{w}_k \rangle$
    $\boldsymbol{v}_{k+1} = \boldsymbol{v}_{k+1} - \boldsymbol{t}_{k,k}\boldsymbol{v}_k - \boldsymbol{t}_{k-1,k}\boldsymbol{v}_{k-1}$
    $\boldsymbol{w}_{k+1} = \boldsymbol{w}_{k+1} - \boldsymbol{t}_{k,k}\boldsymbol{w}_k - \boldsymbol{w}_{k-1}$ ▷ Bi-orthogonalization
    $\boldsymbol{t}_{k,k+1} = \langle \boldsymbol{v}_{k+1}, \boldsymbol{w}_{k+1} \rangle$
    **if** $\boldsymbol{t}_{k,k+1} \neq 0$ **then**
        $\boldsymbol{w}_{k+1} = \boldsymbol{w}_{k+1} / \boldsymbol{t}_{k,k+1}$
    $\boldsymbol{t}_{k+1,k} = 1$
    $k = k + 1$

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

**Algorithm A.4** Variant of the Lanczos Bi-Orthogonalization Process for $\boldsymbol{V}_k$, $\boldsymbol{W}_k$ and $\boldsymbol{T}_k$

**Require:** $\boldsymbol{Q}$, $\boldsymbol{Q}^\mathsf{T}$, $\boldsymbol{c}$ and $\boldsymbol{x}_0$
Set $\boldsymbol{v}_0 = \boldsymbol{w}_0 = 0$ and $\boldsymbol{\delta}_0 = 1$
Set $\boldsymbol{v}_1 = \boldsymbol{c} - \boldsymbol{Q}\boldsymbol{x}_0$ and $\boldsymbol{w}_1$ such that $\boldsymbol{\delta}_1 := \langle \boldsymbol{v}_1, \boldsymbol{w}_1 \rangle \neq 0$ ▷ Initial Krylov vectors
$k = 1$
**while** $\boldsymbol{\delta}_k \neq 0$ **do**
    $\boldsymbol{v}_{k+1} = \boldsymbol{Q}\boldsymbol{v}_k$ and $\boldsymbol{w}_{k+1} = \boldsymbol{Q}^\mathsf{T}\boldsymbol{w}_k$ ▷ Compute next Krylov vectors
    $\boldsymbol{t}_{k,k} = \langle \boldsymbol{v}_{k+1}, \boldsymbol{w}_k \rangle / \boldsymbol{\delta}_k$
    $\boldsymbol{t}_{k-1,k} = \boldsymbol{\delta}_k / \boldsymbol{\delta}_{k-1}$
    $\boldsymbol{v}_{k+1} = \boldsymbol{v}_{k+1} - \boldsymbol{t}_{k,k}\boldsymbol{v}_k - \boldsymbol{t}_{k-1,k}\boldsymbol{v}_{k-1}$
    $\boldsymbol{w}_{k+1} = \boldsymbol{w}_{k+1} - \boldsymbol{t}_{k,k}\boldsymbol{w}_k - \boldsymbol{t}_{k-1,k}\boldsymbol{w}_{k-1}$ ▷ Bi-orthogonalization
    $\boldsymbol{\delta}_{k+1} = \langle \boldsymbol{v}_{k+1}, \boldsymbol{w}_{k+1} \rangle$
    $k = k + 1$

## 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.

**Algorithm A.5** Transpose-Free Lanczos Bi-Orthogonalization Process for $\boldsymbol{V}_k$

**Require:** $\boldsymbol{Q}$, $\boldsymbol{c}$ and $\boldsymbol{x}_0$

Set $\boldsymbol{v}_0 = \boldsymbol{u}_0 = 0$ and $\boldsymbol{\delta}_0 = 1$
Set $\boldsymbol{v}_1 = \boldsymbol{c} - \boldsymbol{Q}x_0$ and $\boldsymbol{w}$ such that $\boldsymbol{\delta}_1 := \langle \boldsymbol{v}_1, \boldsymbol{w} \rangle \neq 0$ ▷ Initial Krylov vector
$k = 1$
**while** $\boldsymbol{\delta}_k \neq 0$ **do**
    Set $\boldsymbol{s} = \boldsymbol{Q}\boldsymbol{v}_k$, $\boldsymbol{t}_{k,k} = \langle \boldsymbol{s}, \boldsymbol{w} \rangle / \boldsymbol{\delta}_k$ and $\boldsymbol{t}_{k-1,k} = \boldsymbol{\delta}_k / \boldsymbol{\delta}_{k-1}$
    $\boldsymbol{u}_k = \boldsymbol{s} - \boldsymbol{t}_{k,k}\boldsymbol{v}_k - \boldsymbol{t}_{k-1,k}\boldsymbol{u}_{k-1}$
    Set $\boldsymbol{d} = \boldsymbol{u}_k - \boldsymbol{t}_{k-1,k}\boldsymbol{u}_{k-1}$ and $\boldsymbol{s} = \boldsymbol{Q}\boldsymbol{d}$
    $\boldsymbol{v}_{k+1} = \boldsymbol{s} - \boldsymbol{t}_{k,k}\boldsymbol{d} + \boldsymbol{t}_{k-1,k}^2\boldsymbol{v}_{k-1}$
    $\boldsymbol{\delta}_{k+1} = \langle \boldsymbol{v}_{k+1}, \boldsymbol{w} \rangle$
    $k = k + 1$

## A.4 Projected Lanczos Bi-Orthogonalization Process

The analogue of Algorithm 2.6 for the class of projected Krylov algorithms for (3) based on the Lanczos bi-orthogonalization Algorithm 2.3 may be stated as Algorithm A.6.

**Algorithm A.6** Lanczos-Based Projected Krylov Methods for (3)

**Require:** $\boldsymbol{Q}$, $\boldsymbol{Q}^\mathsf{T}$, $\boldsymbol{P}_G$, $\boldsymbol{c}$ and $\boldsymbol{x}_0$

Set $\boldsymbol{v}_0 = \boldsymbol{w}_0 = \mathbf{o}$, $\boldsymbol{t}_{0,1} = \boldsymbol{t}_{1,0} = 1$
Set $\boldsymbol{s} = \boldsymbol{c} - \boldsymbol{Q}\boldsymbol{x}_0$. Set $\boldsymbol{w}_1$ such that $\langle \boldsymbol{s}, \boldsymbol{w}_1 \rangle = 1$.
Compute $\boldsymbol{v}_1 = \boldsymbol{P}_G(\boldsymbol{s})$ ▷ Initial Krylov vectors
Compute the solution estimate $\boldsymbol{x}_1 = \boldsymbol{V}_1 \Theta_1(\boldsymbol{T}_{1,0}, \boldsymbol{t}_{1,0})$
$k = 1$
**while** $\boldsymbol{t}_{k-1,k} \neq 0$ **do**
    $\boldsymbol{s} = \boldsymbol{Q}\boldsymbol{v}_k$ and $\boldsymbol{u} = \boldsymbol{Q}^\mathsf{T}\boldsymbol{w}_k$
    Compute $\boldsymbol{v}_{k+1} = \boldsymbol{P}_G(\boldsymbol{s})$ and $\boldsymbol{w}_{k+1} = \boldsymbol{P}_G(\boldsymbol{u})$ ▷ Compute next Krylov vectors
    $\boldsymbol{t}_{k,k} = \langle \boldsymbol{s}, \boldsymbol{w}_k \rangle$
    $\boldsymbol{v}_{k+1} = \boldsymbol{v}_{k+1} - \boldsymbol{t}_{k,k}\boldsymbol{v}_k - \boldsymbol{t}_{k-1,k}\boldsymbol{v}_{k-1}$
    $\boldsymbol{w}_{k+1} = \boldsymbol{w}_{k+1} - \boldsymbol{t}_{k,k}\boldsymbol{w}_k - \boldsymbol{w}_{k-1}$ ▷ Bi-orthogonalization
    $\boldsymbol{t}_{k,k+1} = \langle \boldsymbol{s}, \boldsymbol{w}_{k+1} \rangle$
    **if** $\boldsymbol{t}_{k,k+1} \neq 0$ **then**
        $\boldsymbol{w}_{k+1} = \boldsymbol{w}_{k+1} / \boldsymbol{t}_{k,k+1}$
    $\boldsymbol{t}_{k+1,k} = 1$
    Compute the solution estimate $\boldsymbol{x}_{k+1} = \boldsymbol{V}_{k+1} \Theta_{k+1}(\boldsymbol{T}_{k+1,k}, \boldsymbol{t}_{1,0})$
    $k = k + 1$