A BLOCK MINRES ALGORITHM BASED ON THE BANDED LANCZOS METHOD

KIRK M. SOODHALTER†

Abstract. We develop a block minimum residual (MINRES) algorithm for symmetric indefinite matrices. This version is built upon the banded Lanczos method, which generates one basis vector of the block Krylov subspace per iteration rather than a whole block, as in the block Lanczos process. However, we modify the method such that the most expensive operations are still performed in a block fashion. With the banded Lanczos method, one can detect breakdown from scalar values arising in the computation, allowing for a handling of breakdown which is straightforward to implement.

In this paper, we derive a progressive formulation of the MINRES method based on the banded Lanczos process and give some implementation details. Specifically, a simple reordering of the steps allows us to perform many of the operations at the block level, in order take advantage of communication efficiencies offered by the block Lanczos process. This is an important concern in the context of next generation super computing applications. We also discuss various techniques for handling breakdown of the banded Lanczos process. Some numerical results are shown to illustrate performance on some sample problems. We also present some experiments to show how the relationship between right-hand sides can affect the performance of this method.

1. Introduction. We wish to efficiently solve

\[ AX = B \]  \hspace{1cm} (1.1)

where \( A \in \mathbb{R}^{n \times n} \) is a symmetric, indefinite matrix, and \( B \in \mathbb{R}^{n \times p} \), with \( p \) right-hand sides. If \( p = 1 \), Krylov subspace methods such as the minimum residual MINRES method of Paige and Saunders [17] have been shown to be effective. For the case \( p > 1 \), block Krylov subspace methods have been proposed; see, e.g., [8, 15, 16, 26]. In general, a block Krylov subspace method functions in the same manner as a Krylov subspace method, but at each iteration the operator is applied to a block of vectors rather than just one. These methods generate \( p \) new search directions per iteration. Many scalar operations become operations involving small, dense matrices. With these methods, one can simultaneously solve the linear system for \( p \) right-hand sides, or solve a system with one right-hand side but over a block Krylov subspace generated by augmenting the given right-hand side vector with random vectors (also advocated in [13]). Though block methods increase the per iteration costs (as measure by floating-point operation counts), they can be more efficient from the standpoint metrics related to movement of data within the computer.

Our goal in this work is to develop a method which solves (1.1) over a block Krylov subspace, for any \( p \geq 1 \), designed to take advantage of the communication efficiencies of block operations (when possible) but with greater ease of implementation, specifically as it relates to breakdown of the process due to a dependent candidate basis vector being generated. In particular, the detection of breakdown should be possible through quantities arising in the computation. Therefore, we seek a Lanczos-type method to generate the block Krylov basis, which is amenable to reordering of the steps to perform as many computations in blocks (e.g., sparse block operations or dense BLAS-3 operations) as possible.

To this end, we introduce a version of the MINRES algorithm for block Krylov subspaces which satisfies our requirements. This algorithm is built upon the banded

†Industrial Mathematics Institute, Johannes Kepler University, Altenbergerstraße 69, A-4040 Linz, Austria. (kirk.soodhalter@indmath.uni-linz.ac.at)

∗This version dated December 16, 2013.
Lanczos process of Ruhe \cite{ruhe}, which generates a basis for the block Krylov subspace one vector at a time, rather than in a block fashion. Our algorithm can be considered a simplification of the algorithm presented in \cite{ong}, which extends Ruhe’s banded Lanczos to generalize the nonsymmetric Lanczos process to the block setting, in the case that \( A \) is symmetric. However, we make modifications to execute some operations in a block fashion.

We present the theoretical derivation needed to develop a minimum residual method based on the banded Lanczos procedure. We also discuss the simple modifications needed to execute some operations in a block fashion as well as some practical implementation details, to simplify the writing of the code. To our knowledge, this is the first paper to provide the implementation details of a block minimum residual algorithm for symmetric matrices.

In the next section, we introduce notation and give a brief review of Krylov subspace methods (both non-block and block). In Section \ref{section:algorithm} we derive a version of the block minimum residual method built upon the banded Lanczos method. In Section \ref{section:progressive} we derive the progressive formulation of this method in detail, built to take advantage of the potential memory savings afforded by the method. In Section \ref{section:dependence} we present modifications to our implementation which accommodate the occurrence of exact or inexact dependence of a candidate block Krylov subspace basis vector. In Section \ref{section:convergence} we discuss convergence properties of block methods. In Section \ref{section:storage} special attention is given to how certain data is stored, to keep the scheme as simple as possible. In Section \ref{section:numerical} we present numerical results.

2. Preliminaries. For the matrix \( A \) and starting vector \( r_0 \), recall that we generate an orthonormal basis for the Krylov subspace

\[
\mathcal{K}_j(A, r_0) = \text{span} \{ r_0, Ar_0, \ldots, A^{j-1}r_0 \}
\]

with the Arnoldi process, where \( r_0 \) is some starting vector. Let \( V_j \in \mathbb{R}^{n \times j} \) be the matrix with orthonormal columns generated by the Arnoldi process spanning \( \mathcal{K}_j(A, r_0) \). Then we have the Arnoldi relation

\[
AV_j = V_{j+1} \Pi_j
\]

with \( \Pi_j \in \mathbb{R}^{(j+1) \times j} \); see, e.g., \cite{hestenes-stiefel} Section 6.3 and \cite{saad}. In the case \( p = 1 \), we can solve \cite{ong} with a Krylov subspace iterative method. Suppose \( x_0 \) is an initial approximation such that \( r_0 = B - Ax_0 \) is the initial residual. At iteration \( j \), we can compute the minimum residual correction \( t_j \in \mathcal{K}_j(A, r_0) \) such that

\[
\arg\min_{t \in \mathcal{K}_j(A, r_0)} \| B - A(x_0 + t) \|
\]

by solving the equivalent small least squares problem

\[
y_j = \arg\min_{y \in \mathbb{R}^{j+1}} \left\| r_0 \| e_1^{(j+1)} - \Pi_j y \right\|
\]

and setting \( t_j = V_j y_j \). Implementations of minimum residual methods, such as GMRES \cite{GMRES} in the nonsymmetric case and MINRES \cite{minres} in the symmetric case, have been described.
Much has been written about the solution of linear systems with multiple right-hand sides. Extending the framework of a Krylov method to the block right-hand side setting involves generalizing the machinery to deal with block vectors; see, e.g., [21, Page 208]. Let \( V_1 \in \mathbb{R}^{n \times p} \) be a matrix with orthonormal columns. At step \( j \), the block Arnoldi process generates an orthonormal basis for a block Krylov subspace

\[
K_j(A, V_1) = K_j(A, v_1^{(1)}) + K_j(A, v_1^{(2)}) + \cdots + K_j(A, v_1^{(p)}).
\]

where \( V_1 = [v_1^{(1)}, v_1^{(2)}, \ldots, v_1^{(p)}] \). In this setting, sparse matrices act on a block of vectors per iteration. For this discussion, we assume for now that

\[
\dim K_m(A, V_1) = mp,
\]

i.e., no linear dependent block Arnoldi vectors are generated. We will return to the case of dependence later.

A strength of generalizing the Arnoldi process (or Lanczos) is that the block level operations (e.g., BLAS-3 operations for dense matrices and block operations for sparse matrices) have been shown to be quite efficient, when measured in metrics relevant in a high performance computing environment, i.e., amount of data moved through memory, frequency of cache missing, and the number of floating point operations performed on a unit of data while it is in cache. These considerations have led to the broader goal to design communication avoiding Krylov subspace methods; see, e.g., [13]. In next generation super computing machines, the movement of data within the machine (e.g., from main memory onto the cache) will represent the dominant computational cost, and algorithms should be judged according to an appropriate data movement metric [5]. When judged according to such metrics, the per iteration data movement costs of a block method are only marginally more expensive than their single vector counterparts (both dense, BLAS-3 operations and sparse block operations). Therefore, we minimize the residual over a larger constraint space without a concomitant increase in computational costs (related to the movement of data). For further work on this topic, see, e.g., [3, 18].

To derive our MINRES algorithm, we use the banded Lanczos process proposed by Ruhe [20] but with certain operations performed in a block fashion. For this method, we adopt the notation that

\[
U_p = [u_1, \ldots, u_p]
\]

denotes the normalized starting block of vectors. Note that \( U_p \) was called \( V_1 \) when describing the block Arnoldi method. Beginning with no symmetry assumption on \( A \), the banded Arnoldi process (see, e.g., [7, 8]) performs the same orthogonalization as the block method, only one vector at a time. We denote the matrix with the first \( j \) banded Arnoldi vectors as columns \( U_j \in \mathbb{R}^{n \times j} \), where for \( j < p \), \( U_j \) has only the first \( j \) starting vectors as columns. Therefore, at each iteration, one matrix-single-vector product is performed as opposed to a matrix-block-vector product in a block-level method. It proceeds in a similar fashion to the single-vector Arnoldi process but starts with \( p \) vectors against which the new vector must be to orthogonalized instead of one.

This algorithm allows one to detect a breakdown from the scalar quantities generated by the banded Lanczos process. By reordering the computations, the banded Lanczos algorithm can be formulated with many of the same block level operations as
the block Lanczos algorithm, e.g., the operator is applied to a block of vectors every $p$ iterations, while maintaining the ease with which we detect breakdown in the banded Lanczos algorithm.

To describe the Arnoldi relation in this setting, we must take care as the iteration number $j$ does not match the dimension of the block Krylov subspace. At iteration $j$, we generate the $(p + j)$th banded Arnoldi vector. At this iteration, we have the banded Arnoldi relation

$$ AU_j = U_{j+p} \mathbf{H}_j. $$

(2.2)

The banded Hessenberg matrix $\mathbf{H}_j \in \mathbb{R}^{(j+p) \times j}$ has $p$ lower subdiagonal entries per column and has the structure

$$ \mathbf{H}_j = \begin{bmatrix} \mathbf{H}_j \\ \mathbf{H}_{p \times j} \end{bmatrix} $$

where $\mathbf{H}_j$ is a square $j \times j$ matrix satisfying the identity

$$ \mathbf{H}_j = U^*_j A U_j. $$

(2.3)

Observe that $\mathbf{H}_{p \times j}$ only has nonzero entries in the last $p$ columns with structure

$$ \mathbf{H}_{p \times j} = \begin{bmatrix} \mathbf{0}_{p \times (m-p)} \\ \mathbf{C}_j \end{bmatrix} $$

where $\mathbf{C}_j \in \mathbb{R}^{p \times p}$ is upper triangular. At iteration $j$, assuming there has been no breakdown, the dimension of the subspace built is $p + j$. The subspace that has been generated at iteration $j$ is the sum of Krylov subspaces generated by each column of $U_p$, i.e.,

$$ \mathbb{K}_{k,m}(A, U_p) = \sum_{\ell=1}^{m} K_{k+1}(A, u_\ell) + \sum_{\ell=m+1}^{p} K_k(A, u_\ell) = \mathcal{R}(U_j). $$

(2.4)

where the pair $(k, m)$ are such that $j = (k - 1)p + m$ with $0 < m < p$. We use the following notation to describe the space we have constructed. The pair $(k, m)$ is used to index the block Krylov subspace built by the banded Lanczos process. The Krylov subspaces in the sum (2.4) for the first $m$ right-hand sides are of dimension $k + 1$ and the remaining are of dimension $k$, as shown in (2.4). For $j = (K - 1)p$, a multiple of the block size, the banded Lanczos process has produced an orthonormal basis spanning the $K$th block Krylov subspace generated by $A$ and $U_p$, i.e.,

$$ \mathbb{K}_K(A, U_p) = \mathbb{K}_{K,0}(A, U_p). $$

(2.5)

At each iteration, one of the subspaces in the sum (2.4) increases by one dimension.

Similar to the symmetric Lanczos relation in the case of a single-vector Krylov method, observe that if $A$ is symmetric, the relation (2.3) implies that $\mathbf{H}_j$ is also symmetric. Due to the banded Hessenberg structure of $\mathbf{H}_j$, we see that $\mathbf{H}_j$ is banded matrix with $p$ superdiagonal entries and $p$ subdiagonal entries per column. This structure implies that we only need the most recent $2p$ basis vectors in order to compute $u_{j+p}$. We have the $2p + 1$ term recurrence relation

$$ Au_j = \sum_{\ell=\min(1,p-j)}^{p+j} h_{\ell,j} u_\ell $$

(2.6)
Due to symmetry, we do not need to compute \( h_{\ell,j} \) where \( \ell < j \) since it was computed previously as \( h_{j,\ell} \). This will require the storage of the lower subdiagonal entries of the last \( p \) columns of \( \overline{H}_j \). This yields Ruhe’s banded Lanczos method, which we present as Algorithm 2.1.

**Algorithm 2.1: The Banded Lanczos Process**

**Input**: \( A \in \mathbb{R}^{n \times n} \) symmetric, \( U_{p} \in \mathbb{R}^{n \times p} \), \( U_{p}^*U_{p} = I_{p} \)

**Output**: \( U_{p+j} \in \mathbb{R}^{n \times (p+j)} \), \( U_{p+j}^*U_{p+j} = I_{p+j} \) and \( \overline{H}_j \in \mathbb{R}^{(p+j) \times m} \), \( \overline{H}_j \) is symmetric with \( p \) lower subdiagonal entries

1. for \( \ell = 1, 2, \ldots, j \) do
2. Compute \( w := Au_{\ell} \)
3. for \( i = \text{max}\{1, p - \ell\}, \ldots, p + \ell - 1 \) do
   4. \( h_{i,\ell} := u_{i}^*w \)
   5. \( w \leftarrow w - h_{i,\ell}u_{i} \)
4. Compute \( h_{p+\ell,\ell} := \|w\|_2 \) and \( u_{p+\ell} := w/h_{p+\ell,\ell} \)

It should be noted; our aim is in contrast to the goals stated in the dissertation of Loher [15], in which the author extended the work of Aliaga et al [11] to a fully block nonsymmetric Lanczos-based method, preferring the flexibility offered by a block method, e.g., with regard to look-ahead and deflation. Furthermore, our approach can be considered as an alternative to the fully block approach of O’Leary [10]. Schmelzer analyzed fully block MINRES and SYMMLQ in [23]. The strategy advocated in the present work was commented upon in [15] as an alternative strategy one could pursue. The flexibility of the fully block methods with regard to breakdown comes at the price of a more complicated implementation. Here, we sacrifice some of this flexibility in exchange for some simplicity of implementation.

We end by describing some nomenclature and notation. We call a vector with multiple columns, such as \( B \) when \( p > 1 \), a block vector. Boldface, upper-case letters are used to denote matrices, including block vectors. Boldface lower-case letters will denote column vectors. We denote the Euclidean norm by \( \|\cdot\| \). For a square, non-singular matrix \( A \), we will denote the condition number associated with the 2-norm, \( \kappa(A) = \|A\|\|A^{-1}\| \). When identifying an equation as a QR-factorization, we will use the convention that the right-hand side of the equation is the QR-factorization of the left-hand side of the equation. We denote the \( k \times k \) identity matrix \( I_k \). We also use the Matlab indexing notation to indicate a range of rows or columns of a matrix, e.g., \( M(i:j,:) \) is the submatrix containing rows \( i \) to \( j \) and all columns of \( M \). To indicate the same submatrix, but for a product of matrices (where the Matlab notation would be ambiguous) we use the similar notation \( (MN)_{i:j,:} \). For a matrix \( M \), we denote its range (i.e., the span of the columns) by \( R(M) \).

Since the word deflation has more than one meaning in our community, we will refer to the process of removing dependent vectors to maintain a linearly independent basis in a block Krylov subspace method simply as removal of dependent vectors.

3. **A Block Minimum Residual Method.** We derive a minimal residual algorithm based on the banded Lanczos process. If we begin with an initial guess \( X_0 \), at the \( j \)th step the following method will produce an approximation \( X_j \in \mathbb{R}^{n \times p} \) such that for each \( 0 < i \leq p \), the residual \( \|b^{(i)} - AX_j^{(i)}\| \) is minimized over the space \( \mathbb{R}_{k,m}(A,F_0) \), where \( X_j^{(i)} \) is the \( i \)th column of \( X_j \), and \( F_0 = B - AX_0 \) is the initial
At step $j$, we minimize each column of the block residual

$$ F_j = B - AX_j $$

over $\mathbb{K}_{k,m}(A,F_0)$. Following the development of MINRES presented in [10], we can derive a block MINRES algorithm based on the banded Lanczos process. Let $E_1^{(j)} \in \mathbb{R}^{(j+p)\times p}$ be the matrix containing the first $p$ columns of $I_{j+p}$. Observe that

$$ E_1^{(j)} = \begin{bmatrix} E_1^{(j-1)} \\ 0_{1\times p} \end{bmatrix}. $$

(3.1)

Given $F_0$, we can normalize it by computing the QR factorization

$$ F_0 = U_p S. $$

(3.2)

At step $j$ of banded Lanczos process, we have the QR factorization $H_j = Q_j R_j$ such that $Q_j \in \mathbb{R}^{(j+p)\times (j+p)}$ is unitary, and $R_j \in \mathbb{R}^{(j+p)\times j}$ is upper triangular. The matrix $R_j$ has a simple block structure,

$$ R_j = \begin{bmatrix} R_j \\ 0_{p\times j} \end{bmatrix}, $$

where $R_j$ is a square, upper triangular, $j \times j$ matrix. Let $f_j^{(i)}$ be the $i$th column of $F_j$, the $j$th block residual. The minimization of $\|f_j^{(i)}\|$ can be rewritten as

$$ \|f_j^{(i)}\| = \min_{x \in \mathbb{K}_{k,m}(A,B)} \|f_0^{(i)} - AX\| $$

$$ = \min_{y \in \mathbb{R}_j} \|U_p S e_j^{(i)} - AU_j y\| $$

$$ = \min_{y \in \mathbb{R}_j} \|U_{j+p} E_1^{(i)} S e_j^{(i)} - U_{j+p} \Pi_j y\| $$

$$ = \min_{y \in \mathbb{R}_j} \|E_1^{(i)} S e_j^{(i)} - \Pi_j y\| $$

$$ = \min_{y \in \mathbb{R}_j} \|Q_j E_1^{(i)} S e_j^{(i)} - R_j y\|. $$

(3.3)

We remind the reader that the upper triangular matrix $S$ coming from (3.2) serves the same role as the norm of the initial residual in single-vector Krylov methods.

We can solve the normal equations individually for each right-hand side, or we can solve for all right-hand sides simultaneously, i.e.,

$$ Y_j = R_j^{-1}(Q_j E_1^{(j)} S)_{1:j,:} \quad \text{with} \quad X_j = X_0 + U_j Y_j $$

Similar to the development of MINRES for one right-hand side in [17], we define

$$ Z_j = Q_j E_1^{(j)} S, $$
whose first $j$ rows define the coefficients of the correction in the basis of search directions, defined by

$$M_j = U_j R_j^{-1}.$$  

Observe that the columns of $M_j$ successively span the same subspaces as the columns of $U_j$ due to the upper triangular structure of $R_j^{-1}$. We denote block vector of search direction coordinates

$$Z_j = Z_j(1:j).$$

The block minimum residual approximation at step $j$ is

$$X_j = X_0 + U_j Y_j$$

$$= X_0 + U_j R_j^{-1} (Q_j^* E_1^{(j)} S_{1:j})$$

$$= X_0 + M_j Z_j.$$

(3.4)

It remains to show that, as in the case of MINRES, this indeed leads to a progressive formulation. As in the single right-hand side case, a computed residual (also sometimes called the recursive residual) is available,

$$\| f_j^{(i)} \| = \| Z_j^{(i)} (j + 1 : j + p) \|,$$

(3.5)

where $Z_j^{(i)}$ is the $i$th column of $Z_j^{(i)}$. This can be derived from (3.3), which also can be written as

$$\| f_j^{(i)} \| = \min_{y \in \mathbb{R}} \| Z_j^{(i)} - \tilde{R}_j y \|.$$

(3.6)

As we assume here that there has been no breakdown in the banded Lanczos process, $R_j$ is nonsingular. Thus, (3.6) can be satisfied exactly in the first $j$ rows. Due to the structure of $\tilde{R}_j$, we have that the residual is simply the norm of the last $p$ entries of $Z_j^{(i)}$, i.e., (3.5).

### 4. Block MINRES for Symmetric Linear Systems.

To obtain a storage-efficient block MINRES algorithm based on the banded Lanczos method, we must discuss the structure of $R_j$. This matrix is the upper $j \times j$ block of $\tilde{R}_j$, which is obtained from the QR-factorization of $H_j$. As the lower subdiagonal of $H_j$ has $p$ nonzero entries, we obtain this factorization using Householder reflection, as opposed to Givens rotations. While we can still use Givens rotations to annihilate each subdiagonal element one-at-a-time, this would be less efficient than applying one Householder reflection per column. However, before we can generate a new reflection, we must apply all previously generated reflections to the newest column. Applications of previous reflections to the new column adds, at most, $p$ new nonzero superdiagonal entries. As a result, the upper triangular $R_j$ has at most $2p$ superdiagonal entries per column.

We have that $M_j = U_j R_j^{-1}$ so that $M_j R_j = U_j$; and this gives us the relationship
between the banded Lanczos vectors and the search directions,
\[
\begin{align*}
  r_{1,1} m_1 &= u_1 \\
  r_{1,2} m_1 + r_{2,2} m_2 &= u_2 \\
  &\vdots \\
  r_{1,2p+1} m_1 + r_{2,2p+1} m_2 + \cdots + r_{2p+1,2p+1} m_{2p+1} &= u_{2p+1} \\
  r_{2,2p+2} m_2 + r_{3,2p+2} m_3 + \cdots + r_{2p+2,2p+2} m_{2p+2} &= u_{2p+2} \\
  &\vdots \\
  r_{j-2p,j} m_{j-2p} + r_{j-2p+1,j} m_{j-2p+1} + \cdots + r_{j,j} m_j &= u_j.
\end{align*}
\]

Thus, to compute \( m_j \) we need \( u_j \) and the \( 2p \) previous search directions.

The Householder reflections must also be applied to \( E_1^{(j)} S \). Let \( H_1^{(j)} \in \mathbb{R}^{\ell \times \ell} \) be the Householder reflection constructed to annihilate the entries in the \( j \)th subdiagonal, embedded in an \( \ell \times \ell \) matrix. At step \( j \), the \( j \)th Householder reflection is applied to \( H_{j-1}^{(j)} \cdots H_1^{(j)} E_1^{(j)} S \). From \( (3.1) \), we see that \( E_1^{(j-1)} S \) is a submatrix of \( E_1^{(j)} S \). Furthermore, \( H_{j-1}^{(j-1)} \cdots H_1^{(j-1)} E_1^{(j-1)} S \) is contained as the upper block in \( H_{j-1}^{(j)} \cdots H_1^{(j)} E_1^{(j)} S \). The reflection \( H_1^{(j)} \) only affects rows \( j \) to \( j+p \) of \( H_{j-1}^{(j)} \cdots H_1^{(j)} E_1^{(j)} S \). Thus we have the relation \( Z_j = \begin{bmatrix} Z_{j-1} & \mathbf{t}_j \end{bmatrix}^T \) where \( \mathbf{t}_j \in \mathbb{R}^p \), and we can update \( X_j \) progressively as an update of \( X_{j-1} \).

\[
X_j = X_{j-1} + m_j Z_j^T.
\]

Rather than individually storing the Householder reflection from the most recent \( 2p \) columns, one can employ the idea presented in \[11\]. The authors suggested that one can store the actions of the Householder reflections for a block of columns as a single matrix for the purpose of applying them at future iterations. They showed that performance gains are possible without discernible loss of accuracy due to floating point errors caused by accumulating the action of multiple Householder reflections in one matrix. For block-size \( p \), \( p \) applications of previous Householder reflections to \( p \) columns of \( \mathbf{H}_j \) is replaced with applying them at once as a dense matrix-matrix multiplication. This dense matrix-matrix multiplication can be performed as a level-3 BLAS operation. Since this is a progressive formulation, we must take care to describe this strategy in that context. If we accumulate the actions of the Householder reflections for every \( p \) columns of \( \mathbf{H}_j \), we can represent this cumulative action as the action of a \( 2p \times 2p \) matrix, which must be applied to the appropriate \( 2p \times 2p \) subvector of each future column. Due to the band structure, we only require the action of the two most recently generated accumulated Householder matrices.

5. Removal of Dependent Lanczos Vectors. In a single-vector Krylov subspace method, it may happen that at step \( j \), we have that \( A \mathbf{v}_j \in \mathbb{K}_j(A, \mathbf{r}_0) \). This implies that the grade of \( A \) with respect to \( \mathbf{r}_0 \) is \( j \), which we denote,
\[
\nu(A, \mathbf{r}_0) = j.
\]

In other words, when the process creates a dependent vector, the grade has been achieved. Since \( \mathbf{r}_0 \) is the initial residual, we have that the approximation
\[
\mathbf{x}_j = \mathbf{x}_0 + \mathbf{t}_j \quad \text{with} \quad \mathbf{t}_j \in \mathbb{K}_j(A, \mathbf{r}_0)
\]
is the exact solution, for any Krylov subspace method (derived through a Petrov-Galerkin condition). This situation is referred to as happy breakdown since it means that the true solution is contained in the existing Krylov subspace, see, e.g., [21 Section 6.5.4].

The notion of Krylov subspace grade has been extended to the block Krylov subspace setting [12], where we denote $\nu(A,F_0)$ the block grade of $A$ with respect to $F_0$ as the smallest integer such that

$$K_{\nu(A,F_0)}(A,F_0) = K_{\nu(A,F_0)+1}(A,F_0)$$

As in the single vector case, if the block grade is achieved during the iteration of a block Krylov subspace method, then the method converges (if the initial block residual is used to generate the subspace). However, if we encounter a dependent vector, this does not signify that we have converged. The process may generate dependent vectors without convergence any of the systems [12]. In the case of algorithms that are built upon the symmetric or nonsymmetric block/banded Lanczos methods, this dependence of the Lanczos vectors can lead to unstable algorithms if not properly handled; see, e.g., [9, 16].

Various strategies have been proposed to mitigate the dependence problem. For block-level algorithms, one must first compute or estimate the range of the block Krylov subspace basis to detect rank deficiency. For symmetric Lanczos-based methods, O'Leary [16] advocates removal of the dependent vector, reducing the block size. The update procedures for the systems not associated with the removed vector do not change, and a progressive update formula can be derived for the systems associated with removed right-hand sides. Baglama [2] suggests that instead of simply removing the dependent vector and reducing block size, one can instead replace the dependent vector with a random vector which has been orthogonalized against all previous Lanczos vectors and continue unabated. For nonsymmetric Lanczos-based block QMR, Aliaga et al. [1] propose to remove basis vectors before exact dependence is detected. Due to issues of stability in block nonsymmetric Lanczos based methods, the authors advocate defining a tolerance $d_{tol} > 0$. After a vector $v$ has been biorthogonalized, we have $\|v\| < d_{tol}$. We then consider $v$ as almost being dependent, and it is removed from the basis. In [1], a bookkeeping scheme is presented to keep track of such removals so that the block QMR algorithm can be adjusted accordingly. Recently, this technique was extended to a block conjugate gradient method for shifted linear systems [4]. The book keeping scheme allows for the dependent vectors to be removed from the process but temporarily retained in memory for the purposes of orthogonalization.

Dubrulle [6] proposes an alternative to the removal of dependent or near-dependent vectors, for use in a block conjugate gradient algorithm. He proposes to use a change-of-basis strategy for the block descent directions and other algorithmic changes to avoid the problem long before near-rank deficiency of the block basis vectors occurs. This additionally avoids the need for basis rank estimation.

In [19], following from [14], that authors suggest that removing nearly dependent directions could represent an unacceptable loss of information. They recommend instead to reintroduce the dependent directions at the next iteration. They also consider some different methods of defining and detecting near breakdown.

For our version of the block MINRES algorithm, we define dependency, as in [1, 4], with the candidate vector being considered dependent if $h_{j+p,j} < \gamma$ for a pre-chosen $0 < \gamma \ll 1$. One of the characteristics of a block Krylov method built from the banded Lanczos method is that there is no need for any basis rank estimation.
we construct only one banded Lanczos vector at a time, we simply need to compute $h_{j+p,j}$, i.e., compute the norm of the newest basis vector after orthogonalization via the banded Lanczos process. This has been observed previously (in the context of eigenvalue computations) [2]. Baglama presents two options for dealing with linear dependence. One option is to reduce block size by one and adjust short-term recurrences accordingly. The other is to generate a random vector and orthogonalize it with respect to all previous Lanczos vectors. This normalized vector is then put in the place of the dependent Lanczos vector. We show that either option results in minimal changes to the algorithm. We discuss the algorithmic modifications required to incorporate both options into our block MINRES algorithm, but we choose the latter to maintain the larger block size. For illustration, we present examples for a particular block size, for ease of discussion; however, it is clear that the simplifications presented do not change if the block size increases.

Let us assume for simplicity that we only remove truly dependent vectors, i.e., $\gamma = 0$. We discuss the effects of employing the first option: block size reduction. Suppose we have block size $p = 2$, and we run ten iterations of the banded Lanczos algorithm, such that no dependent vectors arise. We have generated the block Krylov subspace $K_{6,0}(A, U_2)$. Implicitly, we have also generated

$$
\Pi_{10} = \begin{bmatrix}
    h_{1,1} & h_{1,2} & h_{1,3} \\
    h_{2,1} & h_{2,2} & h_{2,3} & h_{2,4} \\
    h_{3,1} & h_{3,2} & h_{3,3} & h_{3,4} & h_{3,5} \\
    h_{4,2} & h_{4,3} & h_{4,4} & h_{4,5} & h_{4,6} \\
    h_{5,3} & h_{5,4} & h_{5,5} & h_{5,6} & h_{5,7} \\
    h_{6,4} & h_{6,5} & h_{6,6} & h_{6,7} & h_{6,8} \\
    h_{7,5} & h_{7,6} & h_{7,7} & h_{7,8} & h_{7,9} \\
    h_{8,6} & h_{8,7} & h_{8,8} & h_{8,9} & h_{8,10} \\
    h_{9,7} & h_{9,8} & h_{9,9} & h_{9,10} & h_{9,11} \\
    h_{10,8} & h_{10,9} & h_{10,10} & h_{10,11} & h_{10,12} \\
    h_{11,9} & h_{11,10} & h_{11,11} & h_{11,12} & h_{11,13} \\
    h_{12,10} & h_{12,11} & h_{12,12} & h_{12,13} & h_{12,14}
\end{bmatrix}
$$

(5.1)

where $\Pi_{10}$ is symmetric, and we have $AU_{10} = U_{12}\Pi_{10}$. Now, suppose that instead $Au_5 \in K_{3,0}(A, U_2)$. This means that

$$
Au_5 = h_{7,5}u_7 + h_{6,5}u_6 + h_{5,5}u_5 + h_{4,5}u_4 + h_{3,5}u_3
$$

(5.2)

with $h_{7,5} = 0$. Then we take $u_7 = 0$, do not add it to our basis, and assign $h_{7,5} = 0$. Furthermore, since this implies that $Au_7 = 0$, we have that $u_9 = 0$, $u_{11} = 0$, etc. Thus, certain entries of $\Pi_{10}$ will be annihilated. Specifically, columns seven and nine are now zero. By symmetry, nonzero entries in rows seven and nine are also annihilated, and the same is true for row eleven. With this in mind, we can actually construct a smaller, banded matrix and write a more compact banded Lanczos relation which ignores the eliminated Lanczos vectors. Note that for the purposes of this description, we do not renumber the banded Lanczos vectors when one is eliminated due to the removal of a dependent vector. We maintain the same indexing as before,
meaning indices associated with annihilated vectors are no longer used. Let

\[
\tilde{\mathbf{H}}_{10} = \begin{bmatrix}
h_{1,1} & h_{1,2} & h_{1,3} \\
h_{2,1} & h_{2,2} & h_{2,3} & h_{2,4} \\
h_{3,1} & h_{3,2} & h_{3,3} & h_{3,4} & h_{3,5} \\
h_{4,2} & h_{4,3} & h_{4,4} & h_{4,5} & h_{4,6} \\
h_{5,3} & h_{5,4} & h_{5,5} & h_{5,6} \\
h_{6,4} & h_{6,5} & h_{6,6} & h_{6,8} \\
h_{8,6} & h_{8,8} & h_{8,10} \\
h_{10,8} & h_{10,10} \\
h_{12,10}
\end{bmatrix}
\]

(5.3)

so that we have the compacted banded Lanczos relation \( \mathbf{A}\tilde{\mathbf{U}}_{10} = \tilde{\mathbf{U}}_{12}\mathbf{H}_{10} \), where

\[
\tilde{\mathbf{U}}_{12} = [\mathbf{u}_1 \hspace{1em} \mathbf{u}_2 \hspace{1em} \cdots \hspace{1em} \mathbf{u}_6 \hspace{1em} \mathbf{u}_8 \hspace{1em} \mathbf{u}_{10} \hspace{1em} \mathbf{u}_{12}],
\]

and \( \tilde{\mathbf{U}}_{10} \) is similarly defined but without the last column. Notice that this removal of one dependent vector results in a reduction of the effective bandwidth by two when we switch from \( \mathbf{H}_{10} \) to \( \mathbf{H}_{10} \). However, this reduction happens in two stages. First, bandwidth reduces once in column five. Then in column eight (actually now the seventh column) there is a second reduction. This can easily be generalized. For block size \( p \), if we observe that \( \mathbf{A}\mathbf{u}_i \) is contained in an old Krylov subspace and therefore \( \mathbf{u}_{i+p} = \mathbf{0} \), then we will see a reduction of bandwidth at column \( i \) and another reduction at column \( i + p \). This yields a reduction in the number of vectors required for storage in the banded Lanczos process. We need two fewer vectors for each block size reduction.

This change in bandwidth is reflected in the upper triangular bands of \( \mathbf{R}_{10} \), the upper triangular matrix defined by the QR factorization \( \mathbf{H}_{10} = \mathbf{Q}_{10}\mathbf{R}_{10} \). We have the structure

\[
\tilde{\mathbf{R}}_{10} = \begin{bmatrix}
r_{1,1} & r_{1,2} & r_{1,3} & r_{1,4} & r_{1,5} & 0 \\
r_{2,2} & r_{2,3} & r_{2,4} & r_{2,5} & 0 \\
r_{3,3} & r_{3,4} & r_{3,5} & r_{3,6} \\
r_{4,4} & r_{4,5} & r_{4,6} & r_{4,8} & 0 \\
r_{5,5} & r_{5,6} & r_{5,8} & 0 \\
r_{6,6} & r_{6,8} & r_{7,10} \\
r_{8,8} & r_{8,10} \\
r_{10,10}
\end{bmatrix}
\]

(5.4)

with bold-face zeros indicating where an entry was annihilated by the reduction in block size. Recalling (4.11), this indicates that our storage requirements for constructing the search directions will change as the bandwidth changes. To construct \( \mathbf{m}_5 \), we need \( \mathbf{m}_4, \mathbf{m}_3, \mathbf{m}_2, \) and \( \mathbf{m}_1 \) along with \( \mathbf{u}_5 \). However, to construct \( \mathbf{m}_6 \), we only need four vectors: \( \mathbf{m}_5, \mathbf{m}_4, \mathbf{m}_3, \) and \( \mathbf{u}_6 \). For \( \mathbf{m}_{10} \), we only require three vectors: \( \mathbf{m}_8, \mathbf{m}_7, \) and \( \mathbf{u}_{10} \). This can easily be generalized for block size \( p \). For each dependent vector removed, we will have a two-vector reduction in storage requirements for the construction of the search directions. In total, for each block size reduction, we have a four-vector reduction in storage requirements.

We now discuss how employing the second option (inserting a random, orthogonalized vector into the basis) affects the algorithm. We begin by describing the
replacement procedure in more detail. At iteration \( j \), we compute \( A u_j \). After orthogonalization, we see that \( h_{j+p,j} = 0 \). Thus, \( A u_j \) is in the range of the previous Lanczos vectors. Let \( \hat{u}_{j+p} \) be a vector constructed by taking a random vector \( \hat{w} \), orthogonalizing \( \hat{w} \) with respect to all previous Lanczos vectors, and setting \( \hat{u}_{j+p} = \hat{w} / \| \hat{w} \| \).

Then the algorithm continues as before with this modified block Lanczos basis.

This strategy allows us to maintain the block size \( p \) when a loss of independence is encountered. We advocate this policy, specifically in the context of high-performance computing applications. Of course, this must be weighted against the costs of maintaining a larger block size. If we generate the block Krylov subspace for \( p \) initial residuals to solve (1.1), \( K_{k,m}(A,F_0) \) will contain sufficient information to construct high-quality solutions for all right-hand sides for large enough values of \( j = (k-1)p + m \), in theory [12]. However, in practice, exact convergence in this scenario would not occur. Maintaining the large block size allow us to build a larger constraint space for each block matrix-vector product executed. In the high-performance computing setting, the low costs of this strategy make them worthy of consideration.

What modifications must be made to the block MINRES algorithm to accommodate this strategy? It turns out, very few. Of course, we do not store the complete Lanczos basis, as this would defeat the purpose of developing a method for symmetric systems. However, we need to orthogonalize the random vector against the entire basis. As a work-around, we can generate a random vector at the start of the iteration and simply orthogonalize against each Lanczos vector as it is created. This would require only one additional vector of storage and an additional orthogonalization per iteration. If we are solving a problem in which we expect there to be more than one occurrence of loss of linear independence, we can generate more than one random vector, balancing between increasing the storage requirements and insuring against the dependence problem. This strategy does entail additional computational cost, but it allows us to achieve the goal of maintaining block size \( p \) when breakdown occurs, and it is desirable to maintain the larger block size for the data movement efficiencies previous discussed.

One might be concerned that introducing a vector not created by the banded Lanczos process will destroy the short-term recurrences which make symmetric Lanczos methods so attractive. However, this is not the case. Suppose that after iteration \( j \), we continue the banded Lanczos process with the modified basis. Let \( \tilde{U}_{j+p} \in \mathbb{R}^{n \times (j+p)} \) be the matrix containing the banded Lanczos vectors but with \( \hat{u}_{j+p} \) as its last column. Observe that the matrix \( \tilde{U}_{j+p}^* A \tilde{U}_{j+p} \) is symmetric; inserting the new basis vector does not affect this. Thus, the banded structure of \( \tilde{H}_j \) defined by \( A \tilde{U}_j = \tilde{U}_{j+p} \tilde{H}_j \) is the same as that of \( \tilde{H}_j \). The only change is that we now have zero entries at \( h_{j+p,j} \) and \( h_{j,j+p} \). This, in turn, gives a slight change in structure to \( \tilde{R}_j \), the upper triangular factor in the QR-factorization of \( \tilde{H}_j \).

As an example, suppose \( p = 2 \) and that \( A v_5 \) is in the span of the existing banded Lanczos vectors, as in the last example. If we continue the banded Lanczos process with the modified basis, we have the following structures for \( \tilde{H}_8 \), and
\( \tilde{R}_8 \in \mathbb{R}^{10 \times 8} \)

\[
\tilde{H}_8 = \begin{bmatrix}
h_{1,1} & h_{1,2} & h_{1,3} & h_{2,4} & h_{3,5} & h_{4,6} & 0 & h_{6,7} & h_{6,8} \\
h_{2,1} & h_{2,2} & h_{2,3} & h_{3,4} & h_{4,5} & h_{5,6} & h_{6,8} & h_{7,8} & h_{7,9} \\
h_{3,1} & h_{3,2} & h_{3,3} & h_{4,4} & h_{5,5} & h_{6,5} & h_{6,7} & h_{7,9} & h_{8,9} \\
h_{4,1} & h_{4,2} & h_{4,3} & h_{5,4} & h_{6,4} & h_{6,6} & h_{7,8} & h_{7,9} & h_{8,10} \\
h_{5,1} & h_{5,2} & h_{5,3} & h_{6,4} & h_{6,5} & h_{6,6} & h_{6,8} & h_{8,9} & h_{8,10} \\
h_{6,1} & h_{6,2} & h_{6,3} & h_{6,4} & h_{6,5} & h_{6,6} & h_{7,7} & h_{9,9} & h_{9,10} \\
h_{7,1} & h_{7,2} & h_{7,3} & h_{7,4} & h_{7,5} & h_{7,6} & h_{8,8} & h_{9,9} & h_{10,9} \\
h_{8,1} & h_{8,2} & h_{8,3} & h_{8,4} & h_{8,5} & h_{8,6} & h_{8,7} & h_{9,10} & h_{10,10} \\
h_{9,1} & h_{9,2} & h_{9,3} & h_{9,4} & h_{9,5} & h_{9,6} & h_{9,7} & h_{10,10} & h_{11,10} \\
h_{10,1} & h_{10,2} & h_{10,3} & h_{10,4} & h_{10,5} & h_{10,6} & h_{10,7} & h_{10,10} & h_{12,10}
\end{bmatrix}
\]

and

\[
\tilde{R}_8 = \begin{bmatrix}
r_{1,1} & r_{1,2} & r_{1,3} & r_{1,4} & r_{1,5} & 0 & 0 & 0 & 0 \\
r_{2,1} & r_{2,2} & r_{2,3} & r_{2,4} & r_{2,5} & r_{2,6} & 0 & 0 & 0 \\
r_{3,1} & r_{3,2} & r_{3,3} & r_{3,4} & r_{3,5} & r_{3,6} & 0 & 0 & 0 \\
r_{4,1} & r_{4,2} & r_{4,3} & r_{4,4} & r_{4,5} & r_{4,6} & r_{4,7} & r_{4,8} & 0 \\
r_{5,1} & r_{5,2} & r_{5,3} & r_{5,4} & r_{5,5} & r_{5,6} & r_{5,7} & r_{5,8} & 0 \\
r_{6,1} & r_{6,2} & r_{6,3} & r_{6,4} & r_{6,5} & r_{6,6} & r_{6,7} & r_{6,8} & r_{6,9} \\
r_{7,1} & r_{7,2} & r_{7,3} & r_{7,4} & r_{7,5} & r_{7,6} & r_{7,7} & r_{7,8} & r_{7,9} \\
r_{8,1} & r_{8,2} & r_{8,3} & r_{8,4} & r_{8,5} & r_{8,6} & r_{8,7} & r_{8,8} & r_{8,9} \\
r_{9,1} & r_{9,2} & r_{9,3} & r_{9,4} & r_{9,5} & r_{9,6} & r_{9,7} & r_{9,8} & r_{9,9} \\
r_{10,1} & r_{10,2} & r_{10,3} & r_{10,4} & r_{10,5} & r_{10,6} & r_{10,7} & r_{10,8} & r_{10,9}
\end{bmatrix}
\]

This indicates that the final effects of replacing the dependent vector with a random one are minimal. The two zeros are introduced into upper Hessenberg matrix, but the bandwidth and symmetry properties remain unchanged. The introduction of a zero in the seventh column of \( R_8 \) and another in the ninth simply means that the seventh and ninth banded Lanczos vectors are linear combinations of the previous four rather than the previous five search directions, recalling the construction of the search directions \([4,1] \).

In our discussion of both techniques for handling the removal of a dependent vector, we have assumed exact deflation, i.e., the newest candidate Lanczos vector is exactly in the span of the previous vectors. In practice, we want to remove a generated vector when it is “nearly” dependent, i.e., reject \( v_j \) when \( h_{j+p,j} < \gamma \) where \( \gamma < 1 \) is some dependence tolerance constant sufficiently far from zero, as in \([4,8]\). This is especially true in block Krylov method formulations relying on short-term recurrences, which are formulated with a progressive update of the solution at each iteration. In our code, any removed vector is held in storage and new Lanczos vectors are still orthogonalized with respect to it until the removed vector would naturally have been dropped due to the banded Lanczos relation. It should be noted; removing basis vectors in this way no longer follows the mathematical derivation, and we must understand the effect of this strategy on convergence, choosing \( \gamma \) in a way that balances our need for stability with any delay in convergence this strategy might cause.

13
6. Convergence Theory. Theoretically, block MINRES is a version of block GMRES for symmetric systems. Simoncini and Gallopoulos discussed the convergence properties of block GMRES [24], including a result by Vital [26]. We can describe the quality of the residual produced at iteration $j$. For $b^{(i)}$, the $i$th column of the right-hand side $B$, Algorithm 7.1 minimizes the $i$th column of the residual $f_j^{(i)}$ over the subspace $K_{k,m}(A,F_0)$. Thus, we can expect $\|f_j^{(i)}\|$ to be at least as good as the norm of the residual produced by running $J$ steps of MINRES with $b^{(i)}$ as the single right-hand side, where

$$J = \begin{cases} k + 1 & \text{if } i \leq m \\ k & \text{if } i > m \end{cases}.$$  

This easily can be understood by recalling the definition of $K_{k,m}(A,F_0)$ in (2.4). We observe that having a larger subspace over which to minimize is not guaranteed to give improvements in convergence. The additional information contained in $K_{k,m}(A,F_0)$ may not be helpful in the minimization process. For specially related right-hand sides, though, we may have a great boost in performance.

In theory, a block Krylov subspace iterative method may terminate before the subspace becomes the full space $\mathbb{R}^n$. As in the single vector case, achievement of the block grade implies that the exact solution correction is in that subspace. For a block Arnoldi (or Lanczos)-based method, if $\nu(A,F_0)$ is the block grade, then

$$X = X_0 + T_{\nu(A,F_0)} \text{ for } T_{\nu(A,F_0)} \in \mathbb{K}_{\nu(A,F_0)}(A,F_0)$$

is the exact solution, where the correction is generated by any Petrov-Galerkin condition (since $F_0$ is the initial residual). Using the equivalence between the block Arnoldi and banded Arnoldi bases at specific iterations [25], we can extend this notion of block grade to a block Krylov subspace generated by a banded rather than truly block process. From [25] and the definition of block grade, we have the following straightforward identity.

**Proposition 6.1.** Let $\nu(A,F_0)$ be the block grade of $A$ and $F_0$, as defined in [12]. Then we have

$$\mathbb{K}_{\nu(A,F_0)}(A,F_0) = \mathbb{K}_{\nu(A,F_0),0}(A,F_0)$$

where the banded Krylov subspace satisfies the invariance

$$\mathbb{K}_{\nu(A,F_0),0}(A,F_0) = \mathbb{K}_{\nu(A,F_0)+1,0}(A,F_0). \quad (6.1)$$

We must be careful when describing the notion of grade for a banded Krylov method. Depending on the ordering of the right-hand sides, we will have that at some iteration $K$, there will be no further increase of the banded Krylov subspace dimension, where

$$K \in \{(\nu(A,F_0) - 1)p + 1, (\nu(A,F_0) - 1)p + 2, \ldots, (\nu(A,F_0)p)\}.$$  

However, the exact value of $K$ depends on the ordering of the right-hand sides, since each has an associate grade with respect to the matrix $A$, and from [12] Lemma 6] these individual grades can be related to the block grade by

$$\mathbb{K}_{\nu(A,F_0),0}(A,F_0) = \sum_{i=1}^{p} \mathbb{K}_{\nu_i}(A,f_0^{(i)}) \text{ where } \nu_i = \nu(A,f_0^{(i)}).$$
Thus, to describe the notion of grade in an unambiguous way, we must make one assumption. Without loss of generality, we assume that the columns of \( F_0 \) are ordered such that the single-vector Krylov subspace grades with respect to each right-hand side satisfy

\[
\nu(A, f_0^{(1)}) \leq \nu(A, f_0^{(2)}) \leq \cdots \leq \nu(A, f_0^{(p)}).
\]

This determines the order in which the individual grades are achieved in (2.4) as we iterate. By fixing the ordering of the columns of \( F_0 \) as in (6.2), the iteration \( K \) at which we have achieved the largest possible block Krylov subspace dimension (constructed by a banded Arnoldi-based method) can be unambiguously defined as the \textit{banded grade} with respect to \( A \) and \( F_0 \). Thus, the notion of block grade described in [12] can be translated unambiguously to a banded-Arnoldi based method.

7. A Note on Implementation. We conclude our description with some notes about practical implementation details. In order to achieve the data movement benefits of block operations, we apply the operator to a block of \( p \) vectors every \( p \) iterations. We must store \( 2p \) Lanczos vectors, \( 2p \) search directions, \( 2p \) Householder reflections (or sets of Givens rotations), the lower subdiagonal entries of \( p \) previous columns of \( H_j \), and the \( j \)th column of \( H_j \). We also may store some nearly dependent vectors for orthogonalization and some random vectors used to replace dependent vectors.

If we use the strategy of accumulating the Householder reflections, as described in [11], then we must store two \( 2p \times 2p \) matrices instead of \( 2p \) individual Householder reflections (represented compactly as vectors in \( \mathbb{R}^{p+1} \)). The purpose of accumulating the action of the Householder reflections in this way is so that the accumulated action can be applied onto a block of \( p \) columns. Thus, the triangularization of \( H_j \) is performed using BLAS-3 dense matrix operations. To achieve this in the framework of a progressive implementation of our block MINRES method, we would need to adjust the order of our algorithmic steps. The new order of operations would be:

1. \( A \) is applied to a block of \( p \) vectors
2. The banded Lanczos process is applied to the \( p \) resulting vectors, producing \( p \) new columns of \( H_j \). Any breakdown is detected here.
3. The actions of previous Householder reflections are applied to the \( p \) new columns of \( H_j \).
4. The rest of the algorithm continues as previously described.

While the symmetry of \( A \) allows for a fixed storage requirement, we must take care with how we store the Lanczos vectors and search directions. Our primary goal in describing storage layout is to show how the method can be implemented without much need for tracking of indices. For simplicity of implementation, we advocate that the Lanczos vectors and search directions be stored in a first-in-first-out (FIFO) queue, holding \( 2p \) vectors. This results in the most recently generated vector will be in the last position in the queue, and when a new vector is created, the oldest vector will automatically be overwritten.

The full matrix \( H_j \) need not be stored, but the lower subdiagonal entries are needed for the block Lanczos process (as they are orthogonalization coefficients in future iterations due to symmetry). The subdiagonal entries from the \( p \) most recent columns of \( H_j \) can be stored in a FIFO queue (or in a \( p \times p \) matrix called \( C_{p \times p} \) behaving as a queue with the newest entries inserted into the last column). Storing

\[\footnote{This accumulation of the actions of the Householder reflections is not currently implemented in our code.}\]
the entries in this manner results in the nonzero superdiagonal entries of the current column of $\mathbf{H}_j$ being available as the nonzero antidiagonal entries of $C_{p \times p}$. This allows us to obtain the super diagonal entries of the current column without computing the associated inner products.

For block size $p = 5$ at iteration $j = 7$ of the banded Lanczos process, we have

$$
\mathbf{H}_7(:, 7) = \begin{bmatrix}
0 \\
h_{2,7} \\
h_{3,7} \\
h_{4,7} \\
h_{5,7} \\
h_{6,7} \\
h_{7,7} \\
h_{8,7} \\
h_{9,7} \\
h_{10,7} \\
h_{11,7} \\
h_{12,7} \\
h_{13,7} \\
h_{14,7}
\end{bmatrix}
$$

and

$$
C_{p \times p} = \begin{bmatrix}
h_{3,2} & h_{4,2} & h_{5,3} & h_{6,3} & h_{7,5} \\
h_{4,2} & h_{5,4} & h_{6,4} & h_{7,6} \\
h_{5,2} & h_{6,2} & h_{7,4} & h_{8,5} & h_{9,6} \\
h_{6,2} & h_{7,2} & h_{8,4} & h_{9,5} & h_{10,6} \\
h_{7,2} & h_{8,3} & h_{9,4} & h_{10,5} & h_{11,6}
\end{bmatrix}.
$$

(7.1)

In (7.1), note the correspondence between bold entries in $\mathbf{H}_7(:, 7)$ and the antidiagonal entries of $C_{p \times p}$, computed at previous iterations.

8. **Numerical Results.** We present numerical experiments to demonstrate the effectiveness and behavior of Algorithm [7.1]. In all experiments, we compared the performance of block MINRES with sequential applications of Matlab’s MINRES function. We compared performance using iteration counts and sometimes CPU timings. However, note that if we measure the cost of an iteration according to a data movement metric, the cost of the iteration would be dominated by the block matrix-vector product executed every $p$ iterations, amortized over the subsequent $p$ iterations. The block matrix-vector product does not cost (in data movement) $p$ times as much as $p$ single matrix-vector products [13]. In this metric, an iteration of our method and a sequential MINRES iteration are not equivalent.

All tests were performed on a Macbook Pro containing a 2.3 GHz Intel Core i5 processor with 8 GB of 1333MHz DDR3 main memory running the 64-bit version of Matlab R2011b. In any experiment involving the generation of random vectors, we used Matlab’s mt19937ar random number generator, with seed 0, which was initialized at the beginning of each experiment. The tests were performed for a model shifted Laplacian problem. Let $L \in \mathbb{R}^{n_1 \times n_1}$, with $n_1 = 40000$, be the discretization of the Laplacian operator on a $200 \times 200$ regular grid using central differences, constructed by setting $T = \text{tridiag}(1, -2, 1)$ and $L = h^{-2}(I \oplus T + T \oplus I)$ where $h = 1/199$. This matrix is negative-definite. Let $A = -L - 200I$. Due to the eigenvalue distribution of $L$, we have that $A$ is indefinite. In all experiments, we precondition with the incomplete Cholesky factors of $-L$ constructed using Matlab’s ichol() function with the default settings.

We begin by demonstrating the performance of the algorithm on the shifted Laplacian system with ten randomly generate right-hand sides. In Figure 8.1, we see that for these right-hand sides, the block MINRES algorithm converges in fewer iterations and less time.
Algorithm 7.1: Block MINRES (Banded Lanczos Version) with no Breakdown

**Input:** $A \in \mathbb{R}^{n \times n}$ Symmetric, $B \in \mathbb{R}^{n \times p}$, $X_0 = 0$, $\epsilon > 0$, $M \in \mathbb{N}$

**Output:** $X \in \mathbb{R}^{n \times p}$ such that

$\|B(:, j) - AX(:, j)\| / \|B(:, j) - AX_0(:, j)\| < \epsilon \ \forall j \leq p$

1. Compute the QR-Factorization $BV_p S$
2. $\hat{S} \leftarrow SE^{(1)}$
3. $X \leftarrow X_0$
4. $R \leftarrow B - AX$

5. while $\max_{0 < i < p} \left\{ \left\| (Z_j)_{ij+1:ij+i} \right\| \right\} < \epsilon \left\| b^{(i)} \right\|$ and $j \leq M$ do

6. if $j \mod p = 1$ then

7. $W \leftarrow A [v_j \ v_{j+1} \ \cdots \ v_{j+p}]$

8. else

9. if $j \mod p \neq 0$ then

10. $\ell = j \mod p$

11. else

12. $\ell = 12$

13. $w \leftarrow W(:, \ell)$

14. if $j > 1$ then

15. for $i = j - p : j - 1$ do

16. $h_{i,j} = h_{j,i}$

17. $w \leftarrow h_{i,j}w$

18. for $i = j : j + p - 1$ do

19. $h_{i,j} = v_i^*w$

20. $w \leftarrow h_{i,j}v_i$

21. $h_{j+p,j} = \|w\|$ if $j > 1$ then

22. $r_j^{(j)} \leftarrow \mathcal{H}_j^{(j)} \cdots \mathcal{H}_{j-2p}^{(j)} \mathcal{H}_{j-1}^{(j)}$

24. Generate Householder reflection $j$th column of $\overline{H}_j$

25. $r_j^{(j)} \leftarrow \mathcal{H}_j^{(j)}r_j^{(j)}$

26. if $m = 1$ then

27. $m_1 = v_1/\overline{R}_1(1, 1)$

28. else

29. $w \leftarrow v_j$

30. for $i = j - 2p : j - 1$ do

31. $w \leftarrow w - \overline{R}_j(i, j)m_i$

32. $m_j = w/\overline{R}_j(j, j)$

33. $z^T \leftarrow \hat{S}(j,:)$

34. $X \leftarrow X + m_jz^T$

35. $\hat{S} \leftarrow \left[ \hat{S} \ S_{01 \times p} \right]$

36. $j \leftarrow j + 1$
We can also compare performance of our method versus sequential applications of MINRES for varying numbers of right-hand sides. We take as our first right-hand side the vector of all ones. If we have $p$ total right-hand sides, we take the remaining $p - 1$ to be the first $p - 1$ columns of the $I_{n_1}$. In Figure 8.2, we plot for various $p$, the ratio between the iteration count of our method and the total iteration count for $p$ sequential applications of Matlab’s MINRES. For this experiment, we see a reduction in the ratio as $p$ increase, but the marginal benefit of adding each additional right-hand side diminishes for larger numbers of right-hand sides.

We demonstrate that our removal of dependent vectors works as described. Of course, it is difficult to choose a pair of right-hand sides for which dependence will occur in later iterations. Thus, as a simple, easy-to-construct test, we chose the first right-hand side $e_1$, as the first canonical basis vector. The second right-hand side is $Ae_1$, the image of the first canonical basis vector, i.e., the first column of our coefficient matrix. This will result in dependence at the first iteration of our algorithm. As is shown in Figure 8.3, this leads to immediate convergence for that system when running block MINRES. Of course, this example is not likely to occur in practice. It merely demonstrates that the algorithm can handle dependence gracefully.

We demonstrate how the relationship between the right-hand sides can affect the performance of block MINRES. We compared the performance of our block MINRES implementation with that of sequential runs of Matlab’s MINRES for $A$ with three pairs of right-hand sides. For the first pair, let $b_1 = e^{(1)}_{n_1}$ and $b_2 = 1$, the vector of all ones. For second pair of right-hand sides, we let $b_1 = b_2$ but change the second right-hand side by letting $b_2 = e^{(2)}_{n_1}$. In Figure 8.4, we show a comparison of convergence curves for these pairs of right-hand sides. We observe that exchanging $b_2$ for $\hat{b}_2$ degrades the performance of our Block MINRES implementation. Recall that the convergence of a Krylov subspace method for a symmetric system is completely determined by its eigenvalues and the decomposition of the initial residual in the
Fig. 8.2. For different numbers of right-hand sides, we plot the ratio between the number of iterations required by our block MINRES method and the sequential MINRES method to solve the system with multiple right-hand sides to a tolerance of $10^{-8}$. As the number of right-hand sides increases, the ratio decreases, i.e., our method requires fewer iterations than sequential MINRES.

Fig. 8.3. Demonstration of the algorithm’s performance in the case that it encounters dependence. In this case, with the right-hand sides $e_1$ and $Ae_1$, dependence occurs at the first iteration. Since the first right-hand side is the solution to the second system, we get immediate convergence for the second system, and block MINRES continues for the other system, replacing the dependent vector with a random one.

eigenbasis. For an indefinite system, the eigenvalues closest to the origin cause a delay in convergence. In Figure 8.5, we decomposed the three right-hand sides in the eigenbasis and plotted the magnitudes of the 200 eigencomponents associated to small eigenvalues. What we see is that almost all the components of $b_1$ and $b_2$ have similar magnitude while those of $\hat{b}_2$ differ, with some being larger and others being smaller. Therefore, we hypothesize that a pair of right-hand sides that have strong components from different parts of the eigenspace might complement each other well.
Fig. 8.4. Performance of block MINRES for different right-hand sides. In the figure on the left, the two right-hand sides are $b_1 = e_n^{(1)}$ and $b_2 = 1$. In the figure on the right, $b_1$ does not change, but $\hat{b}_2 = e_n^{(2)}$.

Fig. 8.5. Magnitude of the components of different right-hand-sides in the eigenspace spanned by the two hundred eigenvectors associated with the smallest magnitude eigenvalues.

We concoct some experiments to explore this line of thinking further. We construct two right-hand sides, each coming from the span of some subset of eigenvectors. We can further specify how many eigenvector components they have in common and see how this affects convergence.

Let $\{q_1, q_2, \ldots, q_n\}$ be the orthonormal eigenvectors of $A$, in ascending order according to the magnitude of their associated eigenvalues. We define the following subspaces,

$$Q_{S_1} = \text{span} \{q_1, \ldots, q_{100}\}$$
$$Q_{S_2} = \text{span} \{q_{101}, \ldots, q_{200}\}$$
$$Q_L = \text{span} \{q_{n-200+1}, \ldots, q_n\}$$
In the first experiment, we construct both right-hand sides from eigenvectors associated only to eigenvalues of smaller magnitude, i.e., \( b_1, b_2 \in Q_{S_1} \oplus Q_{S_2} \), such that a fixed number of eigenvectors are used to construct both vectors. We define the two right-hand sides

\[
b_1 = \sum_{i=1}^{100+m} \alpha_i q_i \quad \text{and} \quad b_2 = \sum_{i=100-m+1}^{200} \beta_i q_i. \tag{8.1}\]

For \( m = 0 \), \( b_1 \) and \( b_2 \) are orthogonal. For \( m = 1 \), they both have components from \( q_{100} \) and \( q_{101} \) but are otherwise orthogonal. For \( m = 100 \), both right-hand sides have components in all 200 basis vectors of \( Q_{S_1} \oplus Q_{S_2} \). For various values of \( m \), we can test the performance of our algorithm. The coefficients \( \{\alpha_i\} \) and \( \{\beta_i\} \) are generated using Matlab’s \texttt{rand()} command. In order to avoid judging performance based on a specific random example (which may be an outlier), for each \( m \) tested, we generated 100 different pairs of right-hand sides. In Figure 8.6 we plot the average iteration counts over the 100 tests for each \( m \). Until \( m = 100 \), we see little change in the iteration counts.

We also performed the same experiment but constructed the two right-hand sides using eigenvectors from different parts of the spectrum. For different values of \( m \), we define

\[
b_1 = \sum_{i=1}^{200} \alpha_i q_i + \sum_{i=n_1-200+1}^{n_1-200+m} \alpha_i q_i, \quad \text{and} \quad b_2 = \sum_{i=201-m}^{200} \beta_i q_i + \sum_{i=n_1-200+1}^{n_1} \beta_i q_i. \tag{8.2}\]

When \( m = 0 \), we have \( b_1 \in Q_{S_1} \oplus Q_{S_2} \) and \( b_2 \in Q_L \) and they are orthogonal. For \( m = 1 \) they share components from two eigenvectors (\( q_{200} \) and \( q_{n_1-200+1} \)). For \( m = 200 \),
both right-hand sides have components from every basis vector of $Q_{S_1} \oplus Q_{S_2} \oplus Q_L$. As in the previous experiment, 100 random pairs of right-hand sides were generated for each $m$, and the results averaged. Average iterations counts are shown in Figure 8.7. We see a quick drop in iterations at $m = 25$ followed by an increase. Overall, mixing eigencomponents in this experiment produces a decrease in iteration counts.

![Figure 8.7](image_url)

**Fig. 8.7.** Experiments constructed in the same as those shown in Figure 8.6 but now with pairs of right-hand sides constructed according to (8.2). Again the average iterations and times from tests for 100 pairs of right-hand sides are given.

This is by no means a rigorous analysis of the convergence of a block method. These experiments only are meant to illustrate the variability of performance of a block method for different right-hand sides and provide some insight into this phenomenon.

9. **Conclusions.** We have presented an implementation of the block MINRES algorithm based on the banded Lanczos process. This version is designed to perform as many operations as possible in a block fashion while maintaining the banded Lanczos method’s easy-to-implement breakdown detection property. We provide not only a theoretical derivation of the algorithm but also a discussion of the practical implementation issues which need to be addressed to fully take advantage of the efficiencies which arise in a block method for symmetric systems. This variant of the block MINRES method handles dependence of block Krylov subspace basis vectors in a more straightforward manner than a block Lanczos-based algorithm. A software implementation in Matlab is provided at [http://math.soodhalter.com/software.php](http://math.soodhalter.com/software.php).

**Acknowledgement.** The author would like to thank Sebastian Birk, Michael Parks, and Daniel Szyld for their constructive editorial comments and suggestions. The author would also like to express gratitude to the two reviewers and editor who offered extensive comments and constructive criticism, which were of great help in improving this manuscript.

**REFERENCES**

[1] José I. Aliaga, Daniel L. Boley, Roland W. Freund, and Vicente Hernández, A Lanczos-type method for multiple starting vectors, Math. Comp., 69 (2000), pp. 1577–1602.
[2] James Baglama, Dealing with linear dependence during the iterations of the restarted block Lanczos methods, Numer. Algorithms, 25 (2000), pp. 23–36.
[3] Allison H. Baker, John M. Dennis, and Elisabeth R. Jessup, On improving linear solver performance: a block variant of GMRES, SIAM J. Sci. Comput., 27 (2006), pp. 1608–1626.
[4] Sebastian Birk and Andreas Frommer, A deflated conjugate gradient method for multiple right-hand sides and multiple shifts, (In preparation).
[5] J. Jack Dongarra and Aad J. van der Sten, High-performance computing systems: status and outlook, Acta Numer., 21 (2012), pp. 379–474.
[6] Augustin A. Dubrulle, Retooling the method of block conjugate gradients, Electron. Trans. Numer. Anal., 12 (2001), pp. 216–233 (electronic).
[7] Roland W. Freund, Computation of matrix Padé approximations of transfer functions via a Lanczos-type process, in Approximation theory VIII, Vol. 1 (College Station, TX, 1995), vol. 6 of Ser. Approx. Decompos., World Sci. Publ., River Edge, NJ, 1995, pp. 215–222.
[8] Roland W. Freund and Manish Malhotra, A block QMR algorithm for non-Hermitian linear systems with multiple right-hand sides, in Proceedings of the Fifth Conference of the International Linear Algebra Society (Atlanta, GA, 1995), vol. 254, 1997, pp. 119–157.
[9] Roland W. Freund and Noël M. Nachtigal, QMR: a quasi-minimal residual method for non-Hermitian linear systems, Numer. Math., 60 (1991), pp. 315–339.
[10] Anne Greenbaum, Iterative Methods for Solving Linear Systems, SIAM, Philadelphia, 1997.
[11] Martin H. Gutknecht and Thomas Schmelzer, Updating the QR decomposition of block tridiagonal and block Hessenberg matrices, Appl. Numer. Math., 58 (2008), pp. 871–883.
[12] The block grade of a block Krylov space, Linear Algebra Appl., 430 (2009), pp. 174–185.
[13] Mark Hoemmen, Communication-avoiding Krylov subspace methods, PhD thesis, University of California Berkeley, 2010.
[14] Julian Langou, Iterative methods for solving linear systems with multiple right-hand sides, PhD thesis, CERFACS, France, 2003.
[15] Damian Loher, Reliable nonsymmetric block Lanczos algorithms, PhD thesis, Diss. no. 16337, ETH Zurich, Zurich, Switzerland, 2006.
[16] Dianne P. O’Leary, The block conjugate gradient algorithm and related methods, Linear Algebra Appl., 29 (1980), pp. 293–322.
[17] Chris C. Paige and Michael A. Saunders, Solutions of sparse indefinite systems of linear equations, SIAM J. Numer. Anal., 12 (1975), pp. 617–629.
[18] Michael L. Parks, Kirk M. Soodhalter, and Daniel B. Szyld, Block Krylov subspace recycling, In Preparation.
[19] Mickaël Robbé and Miloud Sadkane, Exact and inexact breakdowns in the block GMRES method, Linear Algebra Appl., 419 (2006), pp. 265–285.
[20] Axel Ruhe, Implementation aspects of band Lanczos algorithms for computation of eigenvalues of large sparse symmetric matrices, Math. Comp., 33 (1979), pp. 680–687.
[21] Yousef Saad, Iterative methods for sparse linear systems, SIAM, Philadelphia, Second ed., 2003.
[22] Yousef Saad and Martin H. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM Journal on Scientific and Statistical Computing, 7 (1986), pp. 856–869.
[23] Thomas Schmelzer, Block Krylov methods for Hermitian linear systems, master’s thesis, 2004.
[24] Valeria Simoncini and Efstratios Gallopoulos, Convergence properties of block GMRES and matrix polynomials, Linear Algebra Appl., 247 (1996), pp. 97–119.
[25] Valeria Simoncini and Daniel B. Szyld, Recent computational developments in Krylov subspace methods for linear systems, Numer. Linear Algebra Appl., 14 (2007), pp. 1–59.
[26] Brigitte Vital, Étude de quelques méthodes de résolution de problèmes linéaires de grande taille sur multiprocesseur, PhD thesis, Université de Rennes, 1990.