Scalable Library
for Eigenvalue Problem
Computations

SLEPc Technical Report **STR-8**
Available at http://slepc.upv.es

Restarted Lanczos Bidiagonalization for the SVD in SLEPc

V. Hernández
J. E. Román
A. Tomás

Last update: June, 2007 (SLEPc 2.3.3)
Previous updates: –

About SLEPc Technical Reports: These reports are part of the documentation of SLEPc, the *Scalable Library for Eigenvalue Problem Computations*. They are intended to complement the Users Guide by providing technical details that normal users typically do not need to know but may be of interest for more advanced users.

Contents

1 Introduction

The singular value decomposition (SVD) of an $m \times n$ complex matrix A can be written as

$$A = U\Sigma V^*, \tag{1}$$

where $U = [u_1, \ldots, u_m]$ is an $m \times m$ unitary matrix ($U^*U = I$), $V = [v_1, \ldots, v_n]$ is an $n \times n$ unitary matrix ($V^*V = I$), and Σ is an $m \times n$ diagonal matrix with nonnegative real diagonal entries $\Sigma_{ii} = \sigma_i$ for $i = 1, \ldots, \min\{m, n\}$. If A is real, U and V are real and orthogonal. The vectors u_i are called the left singular vectors, the v_i are the right singular vectors, and the σ_i are the singular values. In this report, we will assume without loss of generality that $m \geq n$.

It is possible to formulate the problem of computing the singular triplets (σ_i, u_i, v_i) of A as an eigenvalue problem involving a Hermitian matrix related to A. There are two possible ways of achieving this:

1. With the *cross product* matrix, A^*A.
2. With the *cyclic* matrix, $H(A) = \left[\begin{smallmatrix} 0 & A \\ A^* & 0 \end{smallmatrix}\right]$.

The singular values are the nonnegative square roots of the eigenvalues of the cross product matrix. This approach may imply a severe loss of accuracy in the smallest singular values. The cyclic matrix approach is an alternative that avoids this problem, but at the expense of significantly increasing the cost of the computation. Additional details can be found in chapter 4 of the SLEPc Users Manual.

Computing the cross product matrix explicitly is not recommended, especially in the case of sparse A. Bidiagonalization was proposed by Golub and Kahan [1965] as a way of tridiagonalizing the cross product matrix without forming it explicitly. Consider the following decomposition

$$A = PBQ^*, \tag{2}$$

where P and Q are unitary matrices and B is an $m \times n$ upper bidiagonal matrix. Then the tridiagonal matrix B^*B is unitarily similar to A^*A. Additionally, specific methods exist (e.g. [Demmel and Kahan, 1990]) that compute the singular values of B without forming B^*B. Therefore, after computing the SVD of B,

$$B = X\Sigma Y^*, \tag{3}$$

it only remains to combine Eqs. 3 and 2 to get the solution of the original problem, Eq. 1, with $U = PX$ and $V = QY$.

Bidiagonalization can be accomplished by means of Householder transformations or alternatively via Lanczos recurrences. The latter approach is more appropriate for sparse matrix computations and was already proposed in the Golub and Kahan paper, hence it is sometimes referred to as Golub-Kahan-Lanczos bidiagonalization. This report focuses on this technique for computing a partial SVD.

Lanczos bidiagonalization inherits the good properties as well as the implementation difficulties present in Lanczos-based eigensolvers (see the SLEPc Technical Report STR-5, "Lanczos Methods in SLEPc" for complementary information). It is possible to stop after a few Lanczos steps, in which case we obtain Rayleigh-Ritz approximations of the singular triplets. On the other hand, loss of orthogonality among Lanczos vectors has to be dealt with, either by full reorthogonalization or by a cheaper alternative. Block variants of the method can be proposed, as in [Golub *et al.*, 1981]. Also, in the case of slow convergence, restarting techniques become very important, as will be emphasized in this report.

Section 2 provides a general description of the bidiagonalization method, including important aspects such as reorthogonalization and restart. Then section 3 gives some details that are particular to the SLEPc implementation.

2 Description of the Method

This section provides a detailed description of the three main ingredients of the method: Lanczos bidiagonalization, mechanisms for treating loss of orthogonality, and restarting.

2.1 Lanczos Bidiagonalization

The Lanczos bidiagonalization technique can be derived from several equivalent perspectives. We start by setting up the notation. Consider a trimmed version of Eq. 2

$$A = P_n B_n Q_n^*, \tag{4}$$

where the zero rows of the bidiagonal matrix have been removed and, therefore, P_n is now an $m \times n$ matrix with orthonormal columns, Q_n is a unitary matrix of order n, and B_n is the

following square matrix of order n

$$B_n = P_n^* A Q_n = \begin{bmatrix} \alpha_1 & \beta_1 & & & & \\ & \alpha_2 & \beta_2 & & & \\ & & \alpha_3 & \beta_3 & & \\ & & & \ddots & \ddots & \\ & & & & \alpha_{n-1} & \beta_{n-1} \\ & & & & & \alpha_n \end{bmatrix}. \tag{5}$$

The coefficients of this matrix are real and given by $\alpha_j = p_j^* A q_j$ and $\beta_j = p_j^* A q_{j+1}$, where p_j and q_j are the columns of P_n and Q_n, respectively. It is possible to derive a double recurrence to compute these coefficients together with the vectors p_j and q_j, since after choosing q_1 as an arbitrary unit vector, the other columns of P_n and Q_n are determined uniquely (assuming A has full rank).

Pre-multiplying Eq. 5 by P_n, we have the relation $AQ_n = P_n B_n$. Also, if we transpose both sides of Eq. 5 and pre-multiply by Q_n, we obtain $A^* P_n = Q_n B_n^*$. Equating the first k columns of both relations results in

$$AQ_k = P_k B_k, \tag{6}$$
$$A^* P_k = Q_k B_k^* + \beta_k q_{k+1} e_k^*, \tag{7}$$

where B_k denotes the $k \times k$ leading principal submatrix of B_n. Analogous expressions can be written in vector form by equating the jth column only,

$$Aq_j = \beta_{j-1} p_{j-1} + \alpha_j p_j, \tag{8}$$
$$A^* p_j = \alpha_j q_j + \beta_j q_{j+1}. \tag{9}$$

These expressions directly yield the double recursion

$$\alpha_j p_j = Aq_j - \beta_{j-1} p_{j-1}, \tag{10}$$
$$\beta_j q_{j+1} = A^* p_j - \alpha_j q_j, \tag{11}$$

with $\alpha_j = \|Aq_j - \beta_{j-1} p_{j-1}\|_2$ and $\beta_j = \|A^* p_j - \alpha_j q_j\|_2$ since the columns of P_n and Q_n are normalized.

Algorithm 1 (Golub-Kahan-Lanczos Bidiagonalization)

Choose a unit-norm vector q_1
Set $\beta_0 = 0$
For $j = 1, 2, \ldots, k$
  $p_j = Aq_j - \beta_{j-1} p_{j-1}$
  $\alpha_j = \|p_j\|_2$
  $p_j = p_j / \alpha_j$

$q_{j+1} = A^* p_j - \alpha_j q_j$
$\beta_j = \|q_{j+1}\|_2$
$q_{j+1} = q_{j+1}/\beta_j$
end

Eqs. 6 and 7 can be combined by pre-multiplying the first one by A^*, resulting in

$$A^* A Q_k = Q_k B_k^* B_k + \alpha_k \beta_k q_{k+1} e_k^*. \tag{12}$$

The matrix $B_k^* B_k$ is symmetric positive definite and tridiagonal. The conclusion is that Algorithm 1 computes the same information as the Lanczos tridiagonalization algorithm applied to the Hermitian matrix A^*A. In particular, the right Lanczos vectors q_j computed by Algorithm 1 constitute an orthonormal basis of the following Krylov subspace

$$\mathcal{K}_k(A^*A, q_1) = \text{span}\{q_1, A^*Aq_1, \ldots, (A^*A)^{k-1} q_1\}. \tag{13}$$

Another way of combining Eqs. 6 and 7 is by pre-multiplying the second one by A, resulting in this case the following equation

$$AA^* P_k = P_k B_k B_k^* + \beta_k A q_{k+1} e_k^*. \tag{14}$$

Apparently, this equation is not a Lanczos decomposition, as Eq. 12, because vector Aq_{k+1} is not orthogonal to P_k, in general. However, using Eq. 8 we get

$$\begin{aligned} AA^* P_k &= P_k B_k B_k^* + \beta_k^2 p_k e_k^* + \beta_k \alpha_{k+1} p_{k+1} e_k^* & (15)\\ &= P_k (B_k B_k^* + \beta_k^2 e_k e_k^*) + \beta_k \alpha_{k+1} p_{k+1} e_k^*, & (16) \end{aligned}$$

where matrix $B_k B_k^* + \beta_k^2 e_k e_k^*$ is also symmetric positive definite and tridiagonal. Thus, a similar conclusion can be drawn for matrix AA^*, and the left Lanczos vectors p_j span the Krylov subspace $\mathcal{K}_k(AA^*, p_1)$.

There is an alternative way of deriving Algorithm 1, which further displays the intimate relation between Lanczos bidiagonalization and the usual three-term Lanczos tridiagonalization. The idea is to apply the standard Lanczos algorithm to the cyclic matrix, $H(A)$, with the following special initial vector

$$z_1 = \begin{bmatrix} 0 \\ q_1 \end{bmatrix}. \tag{17}$$

It can be shown that the generated Lanczos vectors are the following

$$z_{2j-1} = \begin{bmatrix} 0 \\ q_j \end{bmatrix} \quad \text{and} \quad z_{2j} = \begin{bmatrix} p_j \\ 0 \end{bmatrix}, \tag{18}$$

and that the projected matrix after $2k$ Lanczos steps is

$$T_{2k} = \begin{bmatrix} 0 & \alpha_1 & & & & & \\ \alpha_1 & 0 & \beta_1 & & & & \\ & \beta_1 & 0 & \alpha_2 & & & \\ & & \alpha_2 & 0 & \beta_2 & & \\ & & & \beta_2 & 0 & \ddots & \\ & & & & \ddots & \ddots & \alpha_k \\ & & & & & \alpha_k & 0 \end{bmatrix}. \tag{19}$$

That is, two steps of this procedure compute the same information as one step of Algorithm 1. It is easy to show also that there is a simple permutation (odd-even) that maps T_{2k} into the cyclic matrix $H(B_k)$. Note that, in a computer implementation, this procedure would require about twice as much storage as Algorithm 1.

Due to these equivalences, all the properties and implementation considerations of Lanczos tridiagonalization (see SLEPc Technical Report STR-5) carry over to Algorithm 1. In particular, error bounds for Ritz approximations can be computed very easily. After k Lanczos steps, the Ritz values $\tilde{\sigma}_i$ (approximate singular values of A) are equal to the computed singular values of B_k, and the Ritz vectors are

$$\tilde{u}_i = P_k x_i, \qquad \tilde{v}_i = Q_k y_i, \tag{20}$$

where x_i and y_i are the left and right singular vectors of B_k. With these definitions, and Eqs. 6–7, it is easy to show that

$$A\tilde{v}_i = \tilde{\sigma}_i \tilde{u}_i, \qquad A^*\tilde{u}_i = \tilde{\sigma}_i \tilde{v}_i + \beta_k q_{k+1} e_k^* x_i. \tag{21}$$

If we define the residual norm associated to the Ritz singular triplet, $(\tilde{\sigma}_i, \tilde{u}_i, \tilde{v}_i)$, as

$$\|r_i\|_2 = \left(\|A\tilde{v}_i - \tilde{\sigma}_i \tilde{u}_i\|_2^2 + \|A^*\tilde{u}_i - \tilde{\sigma}_i \tilde{v}_i\|_2^2 \right)^{\frac{1}{2}}, \tag{22}$$

then it can be cheaply computed as

$$\|r_i\|_2 = \beta_k |e_k^* x_i|. \tag{23}$$

2.2 Dealing with Loss of Orthogonality

As in the case of the standard Lanczos tridiagonalization algorithm, Algorithm 1 diverts from the expected behavior when run in finite precision arithmetic. In particular, after a sufficient number of steps the Lanczos vectors start to lose their mutual orthogonality, and this happens together with the appearance in the spectrum of B_j of repeated and spurious Ritz values.

The simplest cure for this loss of orthogonality is full orthogonalization. In Lanczos bidiagonalization, two sets of Lanczos vectors are computed, so full orthogonalization amounts to orthogonalizing vector p_j explicitly with respect to all the previously computed left Lanczos

vectors, and orthogonalizing vector q_{j+1} explicitly with respect to all the previously computed right Lanczos vectors. Algorithm 2 shows the full orthogonalization variant. Note that in the computation of p_j it is no longer necessary to subtract the term $\beta_{j-1}p_{j-1}$, since this is already done in the orthogonalization step, and analogously in the computation of q_{j+1}.

Algorithm 2 (Lanczos Bidiagonalization with Full Orthogonalization)

Choose a unit-norm vector q_1
For $j = 1, 2, \ldots, k$
  $p_j = Aq_j$
  Orthogonalize p_j with respect to P_{j-1}
  $\alpha_j = \|p_j\|_2$
  $p_j = p_j/\alpha_j$
  $q_{j+1} = A^* p_j$
  Orthogonalize q_{j+1} with respect to Q_j
  $\beta_j = \|q_{j+1}\|_2$
  $q_{j+1} = q_{j+1}/\beta_j$
end

This solution was already proposed in the seminal paper by Golub and Kahan [1965], and used in some of the first implementations such as the block version in [Golub *et al.*, 1981]. The main advantage of full orthogonalization is its robustness, since orthogonality is maintained to full machine precision. Note that for this to be true it may be necessary to resort to orthogonalization refinement, as explained in section 3. Its main drawback is the high computational cost, which grows as the iteration proceeds. However, this problem is less important in the context of restarted variants, discussed in subsection 2.3.

An alternative to full orthogonalization is to simply ignore loss of orthogonality and perform only local orthogonalization at every Lanczos step. This technique has to carry out a post-process of matrix T_{2k} in order to determine the correct multiplicity of the computed singular values as well as to discard the spurious ones. See [Cullum *et al.*, 1983] for further details.

Semiorthogonal techniques try to find a compromise between full and local orthogonalization. One of such techniques is partial reorthogonalization (see SLEPc Technical Report STR-5 for details), which uses a cheap recurrence to estimate the level of orthogonality, and when it drops below a certain threshold some corrective measures are applied. This technique has been adapted by Larsen [1998] to the particular case of Lanczos bidiagonalization. In this case, two recurrences are necessary, one for monitoring loss of orthogonality among right Lanczos vectors, and the other for left Lanczos vectors.

One-sided Variant. There is a variation of Algorithm 2 that maintains the effectiveness of full reorthogonalization but with a considerably reduced cost. This technique was proposed by Simon and Zha [2000]. The idea comes from the observation that, in the Lanczos bidiagonalization procedure without reorthogonalization, the level of orthogonality of left and right

Lanczos vectors go hand in hand. If we quantify the level of orthogonality of the Lanczos vectors computed in finite precision arithmetic, $\hat{P}_j$ and $\hat{Q}_j$, as

$$\eta(\hat{P}_j) = \|I - \hat{P}_j^* \hat{P}_j\|_2, \quad \eta(\hat{Q}_j) = \|I - \hat{Q}_j^* \hat{Q}_j\|_2, \tag{24}$$

then it can be observed that at a given Lanczos step, j, $\eta(\hat{P}_j)$ and $\eta(\hat{Q}_j)$ differ in no more than an order of magnitude, except maybe in the case that B_j becomes very ill-conditioned. This observation led to Simon and Zha to propose what they called the one-sided version, shown in Algorithm 3.

Algorithm 3 (One-Sided Lanczos Bidiagonalization)

Choose a unit-norm vector q_1
Set $\beta_0 = 0$
For $j = 1, 2, \ldots, k$
  $p_j = Aq_j - \beta_{j-1} p_{j-1}$
  $\alpha_j = \|p_j\|_2$
  $p_j = p_j / \alpha_j$
  $q_{j+1} = A^* p_j$
  Orthogonalize q_{j+1} with respect to Q_j
  $\beta_j = \|q_{j+1}\|_2$
  $q_{j+1} = q_{j+1} / \beta_j$
end

Note that the only difference of Algorithm 3 with respect to Algorithm 2 is that p_j is no longer orthogonalized explicitly. Still, numerical experiments carried out by Simon and Zha show that the computed $\hat{P}_j$ vectors maintain a similar level of orthogonality as $\hat{Q}_j$. The same behavior has been observed in the context of the SLEPc implementation.

Apart from the substantial reduction of computational cost achieved by the one-sided variant, another very important practical consideration is that the $\hat{P}_j$ vectors need no longer be stored (apart from p_j and p_{j-1}) because (1) they are not involved in the reorthogonalization, and (2) left singular vectors, if required, can be computed by the simple relation $Av_i = \sigma_i u_i$. Note that, in some applications, this may represent a huge savings in memory requirements, especially for the case of very skinny matrix A with $m \gg n$.

2.3 Restarted Bidiagonalization

Restarting is a key aspect in the efficient implementation of projection eigensolvers. This issue is treated in SLEPc reports STR-4, STR-5 and STR-7, in the context of Krylov-type eigensolvers. In this section, we adapt the discussion to the context of Lanczos bidiagonalization.

The number of iterations required in the Lanczos bidiagonalization algorithm (the value of k) can be quite high, if many singular triplets are requested, and also depending on the distribution of the spectrum (convergence is slow in the presence of clustered singular values). Increasing k too much may not be acceptable, since this implies a growth in storage requirements and, more

importantly, a growth of computational cost per iteration (in the case of full orthogonalization). To avoid this problem, restarted variants limit the maximum number of Lanczos steps to a fixed value, k, and when this value is reached the computation is re-initiated. This can be done in different ways.

Explicit restart consists in rerunning the algorithm with a "better" initial vector. In Algorithms 1–3, the initial vector is q_1, so the easiest strategy is to replace q_1 with the (right) Ritz vector associated to the (approximate) dominant singular value. A block equivalent of this technique was employed in [Golub *et al.*, 1981]. In the case that many singular triplets are to be computed, it is not evident how to build the new q_1. One possibility is to compute q_1 as a linear combination of a subset of the computed Ritz vectors, possibly applying a polynomial filter to remove components in unwanted directions.

Implicit restart is a much better alternative that eliminates the need to explicitly compute a new start vector q_1. It consists in combining the Lanczos bidiagonalization process with the implicitly shifted QR algorithm. The k-step Lanczos relations described in Eqs. 6–7 are compacted into order ℓ, and then extended again to order k. The procedure allows the small-size equations to retain the relevant spectral information of the full-size relations. A detailed description of this technique can be found in [Björck *et al.*, 1994], [Larsen, 2001], [Jia and Niu, 2003] and [Kokiopoulou *et al.*, 2004].

An equivalent yet easier to implement alternative to implicit restart is the so-called thick restart, originally proposed in the context of Lanczos tridiagonalization [Wu and Simon, 2000]. We next describe how this method can be adapted to Lanczos bidiagonalization, as proposed in [Baglama and Reichel, 2005].

The main idea of thick-restarted Lanczos bidiagonalization is to reduce the full k-step Lanczos bidiagonalization, Eqs. 6–7, to the following one

$$A\tilde{Q}_{\ell+1} = \tilde{P}_{\ell+1}\tilde{B}_{\ell+1}, \tag{25}$$

$$A^*\tilde{P}_{\ell+1} = \tilde{Q}_{\ell+1}\tilde{B}^*_{\ell+1} + \tilde{\beta}_{\ell+1}\tilde{q}_{\ell+2}e^*_{k+1}, \tag{26}$$

where the value of $\ell < k$ could be for instance the number of wanted singular values. The key point here is to build the decomposition of Eqs. 25–26 in such a way that it keeps the relevant spectral information contained in the full decomposition. This is achieved directly by setting the first ℓ columns of $\tilde{Q}_{\ell+1}$ to be the wanted approximate right singular vectors, and analogously in $\tilde{P}_{\ell+1}$ the corresponding approximate left singular vectors. It can be shown [Baglama and Reichel, 2005] that it is possible to easily build a decomposition that satisfies these requirements, as described next.

We start by defining $\tilde{Q}_{\ell+1}$ as

$$\tilde{Q}_{\ell+1} = [\tilde{v}_1, \tilde{v}_2, \ldots, \tilde{v}_\ell, q_{k+1}], \tag{27}$$

that is, the Ritz vectors $\tilde{v}_i = Q_k y_i$ together with the last Lanczos vector generated by Algorithm 1. Note that this matrix has orthonormal columns because $Q_k^* q_{k+1} = 0$ by construction. Similarly, define $\tilde{P}_{\ell+1}$ as

$$\tilde{P}_{\ell+1} = [\tilde{u}_1, \tilde{u}_2, \ldots, \tilde{u}_\ell, \tilde{p}_{\ell+1}], \tag{28}$$

with $\tilde{u}_i = P_k x_i$, and $\tilde{p}_{\ell+1}$ a unit-norm vector computed as $\tilde{p}_{\ell+1} = f/\|f\|$, where f is the vector resulting from orthogonalizing Aq_{k+1} with respect to the first ℓ left Ritz vectors, $\tilde{u}_i$,

$$f = Aq_{k+1} - \sum_{i=1}^{\ell} \tilde{\rho}_i \tilde{u}_i. \tag{29}$$

It can be shown that the orthogonalization coefficients can be easily computed as $\tilde{\rho}_i = \beta_k e_k^* x_i$ (note that these values are similar to the residual bounds in Eq. 23, but here the sign is relevant). The new projected matrix is

$$\tilde{B}_{\ell+1} = \begin{bmatrix} \tilde{\sigma}_1 & & & & \tilde{\rho}_1 \\ & \tilde{\sigma}_2 & & & \tilde{\rho}_2 \\ & & \ddots & & \vdots \\ & & & \tilde{\sigma}_\ell & \tilde{\rho}_\ell \\ & & & & \tilde{\alpha}_{\ell+1} \end{bmatrix}, \tag{30}$$

where $\tilde{\alpha}_{\ell+1} = \|f\|$ so that Eq. 25 holds. To complete the form of a Lanczos bidiagonalization, it only remains to define $\tilde{\beta}_{\ell+1}$ and $\tilde{q}_{\ell+2}$ in Eq. 26, which turn out to be $\tilde{\beta}_{\ell+1} = \|g\|$ and $\tilde{q}_{\ell+2} = g/\|g\|$, where $g = A^*\tilde{p}_{\ell+1} - \tilde{\alpha}_{\ell+1} q_{k+1}$.

It is shown in [Baglama and Reichel, 2005] that the Lanczos bidiagonalization relation is maintained if Algorithm 1 is run for $j = \ell+2, \ldots, k$ starting from the values of $\tilde{\beta}_{\ell+1}$ and $\tilde{q}_{\ell+2}$ indicated above, thus obtaining a new full-size decomposition. In this case, the projected matrix is no longer bidiagonal,

$$\tilde{B}_k = \begin{bmatrix} \tilde{\sigma}_1 & & & \tilde{\rho}_1 & & & \\ & \tilde{\sigma}_2 & & \tilde{\rho}_2 & & & \\ & & \ddots & \vdots & & & \\ & & & \tilde{\sigma}_\ell & \tilde{\rho}_\ell & & \\ & & & & \tilde{\alpha}_{\ell+1} & \beta_{\ell+1} & \\ & & & & & \ddots & \ddots & \\ & & & & & & \alpha_{k-1} & \beta_{k-1} \\ & & & & & & & \alpha_k \end{bmatrix}, \tag{31}$$

where the values without tilde are computed in the usual way with Algorithm 1. When carried out in an iterative fashion, the above procedure gets the form shown in Algorithm 4.

Algorithm 4 (Thick-restart Lanczos Bidiagonalization)

Input: Matrix A, initial unit-norm vector q_1, and number of steps k
Output: $\ell \leq k$ Ritz triplets

1. Build an initial Lanczos bidiagonalization of order k
2. Compute Ritz approximations of the singular triplets

3. Truncate to a Lanczos bidiagonalization of order ℓ
4. Extend to a Lanczos bidiagonalization of order k
5. If not satisfied, go to step 2

Note that in step 2, it is necessary to use a general dense solver for the computation of the singular triplets of matrix $\tilde{B}_k$, since the bidiagonal structure cannot be exploited.

Step 4 can be carried out by a variation of e.g. Algorithm 3, as illustrated in Algorithm 5. Starting from the new initial vector, $q_{\ell+1}$, this algorithm first computes the corresponding left initial vector, $p_{\ell+1}$, then iterates normally.

Algorithm 5 (One-Sided Lanczos Bidiagonalization – restarted)

$p_{\ell+1} = Aq_{\ell+1}$
For $i = 1, 2, \ldots, \ell$
    $p_{\ell+1} = p_{\ell+1} - B_{i,\ell+1} p_i$
end
$\alpha_{\ell+1} = \|p_{\ell+1}\|_2$
$p_{\ell+1} = p_{\ell+1}/\alpha_{\ell+1}$
For $j = \ell+1, \ell+2, \ldots, k$
    $q_{j+1} = A^* p_j$
    Orthogonalize q_{j+1} with respect to Q_j
    $\beta_j = \|q_{j+1}\|_2$
    $q_{j+1} = q_{j+1}/\beta_j$
    If $j < k$
        $p_{j+1} = Aq_{j+1} - \beta_j p_j$
        $\alpha_{j+1} = \|p_{j+1}\|_2$
        $p_{j+1} = p_{j+1}/\alpha_{j+1}$
    end
end

2.4 Available Implementations

In this subsection, we cite software packages that provide implementations of algorithms discussed in this report. For additional information, see SLEPc Technical Report STR-6, "A Survey of Software for Sparse Eigenvalue Problems".

SVDPACK [Berry, 1992b,a] is a Fortran77 software library that provides four alternative methods for the computation of a partial SVD. Three of these methods are eigensolvers that work with the equivalent eigenproblem associated to the cross product matrix or the cyclic matrix. The fourth method performs a block Lanczos bidiagonalization with full reorthogonalization, in the spirit of [Golub *et al.*, 1981].

PROPACK [Larsen, 1998] is based on the Lanczos bidiagonalization algorithm with partial reorthogonalization. The Fortran version of PROPACK incorporates implicit restart, thus reducing the storage requirements. PROPACK can be used with either real or complex matrices.

3 The SLEPc Implementation

SLEPc provides the following two implementations of Lanczos bidiagonalization:

- Explicit restart version. The corresponding solver is `SVDLANCZOS` (or `-svd_type lanczos` from the command-line). It uses a simple restart mechanism, in which the most recent approximation of the dominant Ritz vector is used as the new starting vector q_1. Converged triplets are locked and explicitly deflated during the iteration.

- Thick-restart version. The corresponding solver is `SVDTRLANCZOS` (or `-svd_type trlanczos` from the command-line). It uses the restarting strategy discussed at the end of section 2.3. Converged triplets are also locked and deflated during the iteration, that is, the first $\tilde{\rho}_i$ are set to zero and the leading part of $\tilde{B}_k$ in Eq. 31 is not considered in the computation of the SVD, but the Ritz vectors are indeed included in the orthogonalization step.

For practical use, it is almost always recommended to use the thick-restarted version. This implementation is further described next.

3.1 The Algorithm

In addition to the features described in this report, the Lanczos bidiagonalization implemented in SLEPc also incorporates several techniques for efficient orthogonalization of vectors during the computation of Lanczos vectors. For numerical robustness, it is necessary to carry out conditional refinement in the orthogonalization, and this has to be carefully implemented in order to maintain good parallel performance. In particular, estimation of the norm and delayed normalization are used in this context. See [Hernandez *et al.*, 2007] and the SLEPc Technical Report STR-1, "Orthogonalization Routines in SLEPc" for a description of these techniques.

Algorithm 6 represents a detailed version of Algorithm 4, that intends to be closer to the actual SLEPc implementation. It makes use of Algorithm 7, which is a version of Algorithm 5 incorporating the efficient orthogonalization techniques mentioned above.

Algorithm 6 (Thick-restart Lanczos Bidiagonalization)

Input: Matrix A, initial vector q_1, and number of steps k
Output: `nev` Ritz triplets, with `nev` $\leq \ell < k$

```
Normalize q_1
Set ℓ = 0
Restart loop
    Run one-sided Lanczos bidiagonalization (Algorithm 7)
    Compute the SVD of B_k = X_k Σ_k Y_k^*
    Compute residual norm estimates, |ρ_i|, with ρ_i = β_k e_k^* x_i
    Exit if enough converged singular triplets, otherwise lock newly converged triplets
    Update ℓ and set q_{ℓ+1} ← q_{k+1}
    Compute Ritz vectors, Q_ℓ ← Q_k Y_{:,1:ℓ}, P_ℓ ← P_k X_{:,1:ℓ}
```

Insert ρ_i in the appropriate positions of B_k
end

Algorithm 7 (One-Sided Lanczos Bidiag. – restarted, with enhancements)

$p_{\ell+1} = Aq_{\ell+1}$
For $i = 1, 2, \ldots, \ell$
  $p_{\ell+1} = p_{\ell+1} - B_{i,\ell+1} p_i$
end
For $j = \ell+1, \ell+2, \ldots, k$
  $q_{j+1} = A^* p_j$
  $c = Q_j^* q_{j+1}$
  $\rho = \|q_{j+1}\|_2$
  $\alpha_j = \|p_j\|_2$
  $p_j = p_j/\alpha_j$
  $q_{j+1} = q_{j+1}/\alpha_j$
  $c = c/\alpha_j$
  $\rho = \rho/\alpha_j$
  $q_{j+1} = q_{j+1} - Q_j c$
  $\beta_j = \sqrt{\rho^2 - \sum_{i=1}^{j} c_i^2}$
  If $\beta_j < \eta\rho$
    $c = Q_j^* q_{j+1}$
    $\rho = \|q_{j+1}\|_2$
    $q_{j+1} = q_{j+1} - Q_j c$
    $\beta_j = \sqrt{\rho^2 - \sum_{i=1}^{j} c_i^2}$
  end
  $q_{j+1} = q_{j+1}/\beta_j$
  If $j < k$
    $p_{j+1} = Aq_{j+1} - \beta_j p_j$
  end
end

Selection of ℓ. Note that in Algorithm 6 the value of ℓ changes dynamically. Currently, the SLEPc implementation sets this value to somewhere in-between k (the maximum subspace dimension, or `ncv`) and the number of currently converged singular triplets. Its value grows progressively as singular values converge.

3.2 User Options

Both bidiagonalization methods implemented in SLEPc support one-sided orthogonalization as discussed in section 2.2. By default, these variants are deactivated, in order to avoid numerical problems in the case of very ill-conditioned B_k. However, in most cases it is safe to activate them with

```
SVDLanczosSetOneSide(SVD svd,PetscTruth oneside)
SVDTRLanczosSetOneSide(SVD svd,PetscTruth oneside)
```

or alternatively in the command-line with `-svd_lanczos_oneside` or `-svd_trlanczos_oneside`.

Activating this option may represent a substantial reduction in the computing time. Additionally, there is a significant reduction in memory requirements with the explicit restarted algorithm, which is especially important in very large problems that do not need to compute left singular vectors.

3.3 Known Issues and Applicability

Currently, the computation of small singular values is not very robust, because convergence of the methods discussed in this report is likely to be slow or, in some cases, unattainable. In order to be more likely to get convergence to small singular values, it would be necessary to incorporate techniques such as harmonic or refined Ritz projections. These techniques are discussed in some of the references cited so far, e.g. [Jia and Niu, 2003], [Kokiopoulou *et al.*, 2004], and [Baglama and Reichel, 2005].

Allowed portion of the spectrum	All
Support for complex numbers	Yes

References

Baglama, J. and L. Reichel (2005). Augmented Implicitly Restarted Lanczos Bidiagonalization Methods. *SIAM J. Sci. Comput.*, 27(1):19–42.

Berry, M. W. (1992a). Large Scale Sparse Singular Value Computations. *Int. J. Supercomp. Appl.*, 6:13–49.

Berry, M. W. (1992b). SVDPACK: A Fortran-77 Software Library for the Sparse Singular Value Decomposition. Technical Report UT-CS-92-159, Department of Computer Science, University of Tennessee.

Björck, Å., E. Grimme, and P. Van Dooren (1994). An Implicit Shift Bidiagonalization Algorithm for Ill-posed Systems. *BIT*, 34(4):510–534.

Cullum, J., R. A. Willoughby, and M. Lake (1983). A Lanczos Algorithm for Computing Singular Values and Vectors of Large Matrices. *SIAM J. Sci. Statist. Comput.*, 4:197–215.

Demmel, J. W. and W. Kahan (1990). Accurate Singular Values of Bidiagonal Matrices. *SIAM J. Sci. Statist. Comput.*, 11(5):873–912.

Golub, G. H. and W. Kahan (1965). Calculating the Singular Values and Pseudo-Inverse of a Matrix. *J. Soc. Indust. Appl. Math. Ser. B Numer. Anal.*, 2:205–224.

Golub, G. H., F. T. Luk, and M. L. Overton (1981). A Block Lánczos Method for Computing the Singular Values of Corresponding Singular Vectors of a Matrix. *ACM Trans. Math. Softw.*, 7(2):149–169.

Hernandez, V., J. E. Roman, and A. Tomas (2007). Parallel Arnoldi Eigensolvers with Enhanced Scalability via Global Communications Rearrangement. *Parallel Comput.*, 33(7–8):521–540.

Jia, Z. and D. Niu (2003). An Implicitly Restarted Refined Bidiagonalization Lanczos Method for Computing a Partial Singular Value Decomposition. *SIAM J. Matrix Anal. Appl.*, 25(1):246–265.

Kokiopoulou, E., C. Bekas, and E. Gallopoulos (2004). Computing Smallest Singular Triplets with Implicitly Restarted Lanczos Bidiagonalization. *App. Numer. Math.*, 49(1):39–61.

Larsen, R. M. (1998). Lanczos Bidiagonalization with Partial Reorthogonalization. Technical Report PB-537, Department of Computer Science, University of Aarhus, Aarhus, Denmark. Available at http://www.daimi.au.dk/PB/537.

Larsen, R. M. (2001). Combining Implicit Restart and Partial Reorthogonalization in Lanczos Bidiagonalization. Technical report, SCCM, Stanford University. Available at http://soi.stanford.edu/ rmunk/PROPACK.

Simon, H. D. and H. Zha (2000). Low-Rank Matrix Approximation Using the Lanczos Bidiagonalization Process with Applications. *SIAM J. Sci. Comput.*, 21(6):2257–2274.

Wu, K. and H. Simon (2000). Thick-Restart Lanczos Method for Large Symmetric Eigenvalue Problems. *SIAM J. Matrix Anal. Appl.*, 22(2):602–616.