UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Scalable Library for Eigenvalue Problem Computations SLEPc

SLEPc Technical Report **STR-5**
Available at http://slepc.upv.es

Lanczos Methods in SLEPc

V. Hernández
J. E. Román
A. Tomás
V. Vidal

Last update: October, 2006 (SLEPc 2.3.2)
Previous updates: SLEPc 2.3.1

Contents

1 Introduction

The method of Lanczos [1950] has become one of the most successful methods for approximating a few eigenvalues of a real symmetric (or complex Hermitian) matrix. Initially, the method did not receive much attention because it was perceived as a method for tridiagonalizing a matrix, a task that was better achieved by the Givens and Householder methods. To compete in accuracy, the Lanczos method had to be supplemented with the explicit orthogonalization of the computed vectors, which in exact arithmetic would be orthogonal automatically. Many years later, a renewed interest in the Lanczos method was caused by Paige's work, which led to a series of important contributions by many authors, resulting in a better understanding of the method and widening its usability. Some of these contributions are described in this report (section 2), in order to motivate the particular variants implemented in SLEPc (section 3).

The Lanczos method is related to the Arnoldi method in the sense that Lanczos can be seen as a particular case of Arnoldi when the matrix is symmetric. For this reason, both methods have some common aspects. The reader is referred to SLEPc Technical Report STR-4, "Arnoldi Methods in SLEPc", for a detailed description of the Arnoldi method.

This report does not include material regarding the non-symmetric version of the Lanczos method. Eigensolvers based on this method could be included in future versions of SLEPc.

2 Description of the Method

This section provides an overview of the Lanczos method and some of its variations, including techniques for avoiding loss of orthogonality. For more detailed background material the reader is referred to [Stewart, 2001b], [Saad, 1992], [Parlett, 1980], [Bai *et al.*, 2000] or [Komzsik, 2003]. See also [Meurant and Strakos, 2006] for details about finite precision issues.

2.1 Basic Lanczos Algorithm

The Lanczos method can be derived from different points of view. One such perspective is the reduction of an $n \times n$ symmetric matrix A to tridiagonal form by means of a three-term recurrence formula. Given a unit-norm initial vector, v_1, and taking $\beta_1 = 0$, the following recurrence

$$\beta_{j+1}v_{j+1} = Av_j - \alpha_j v_j - \beta_j v_{j-1}\,, \tag{1}$$

where $\alpha_j = v_j^* A v_j$ and $\beta_{j+1} = v_{j+1}^* A v_j$, generates an orthonormal set of Lanczos vectors, v_j, and a tridiagonal matrix defined as

$$T = \begin{bmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \beta_3 & \alpha_3 & \ddots & \\ & & \ddots & \ddots & \beta_n \\ & & & \beta_n & \alpha_n \end{bmatrix}. \tag{2}$$

It can be shown that vector v_{n+1} is zero, and that the following relation holds

$$AV - VT = 0\,, \tag{3}$$

where $V = [v_1, v_2, \ldots, v_n]$. That is, the Lanczos recursion computes a tridiagonal matrix which is orthogonally similar to A.

When describing an algorithm for computing the Lanczos recurrence, two observations can be taken into account. The first one is that β_{j+1} can be computed as $\|Av_j - \alpha_j v_j - \beta_j v_{j-1}\|_2$ since v_{j+1} has unit norm. A numerical analysis by Paige [1972] shows that this alternative improves numerical stability when implementing the recurrence in finite precision arithmetic. The second observation is that α_j can be computed as $v_j^*(Av_j - \beta_j v_{j-1})$ since v_j and v_{j-1} are orthogonal by construction. This alternative is also advocated by Paige [1980]. With these observations, the basic Lanczos algorithm can be written as in Algorithm 1.

Algorithm 1 (Basic Lanczos – recurrence view)

```
Choose a unit-norm vector $v_1$
Set $\beta_1 = 0$
For $j = 1, 2, \ldots$
    $u_{j+1} = Av_j - \beta_j v_{j-1}$
    $\alpha_j = v_j^* u_{j+1}$
    $u_{j+1} = u_{j+1} - \alpha_j v_j$
    $\beta_{j+1} = \|u_{j+1}\|_2$
```

```
        If $\beta_{j+1} = 0$, stop
        $v_{j+1} = u_{j+1}/\beta_{j+1}$
    end
```

Of course, the Lanczos algorithm is most useful when not all n vectors are computed, which is not viable in the context of very large matrices. If only m Lanczos steps are carried out, then instead of Eq. 3 the following relation holds

$$AV_m - V_m T_m = \beta_{m+1} v_{m+1} e_m^*, \tag{4}$$

where T_m is the leading $m \times m$ submatrix of T and $V_m = [v_1, v_2, \ldots, v_m]$. Eq. 4 is the same that describes the residual of an m-step Arnoldi factorization. That is, the Lanczos process can also be seen as the computation of the orthogonal projection of matrix A onto the Krylov subspace $\mathcal{K}_m(A, v_1)$. From this perspective, the Lanczos method is equivalent to the Arnoldi method, see Algorithm 2.

Algorithm 2 (Basic Lanczos – projection view)

```
Input: Matrix $A$, number of steps $m$, and initial vector $v_1$ of norm 1
Output: $(V_m, T_m, v_{m+1}, \beta_{m+1})$ so that $AV_m - V_m T_m = \beta_{m+1} v_{m+1} e_m^*$
    For $j = 1, 2, \ldots, m$
        $u_{j+1} = Av_j$
        Orthogonalize $u_{j+1}$ with respect to $V_j$ (obtaining $\alpha_j$)
        $\beta_{j+1} = \|u_{j+1}\|_2$
        If $\beta_{j+1} = 0$, stop
        $v_{j+1} = u_{j+1}/\beta_{j+1}$
    end
```

In Algorithm 2, the second line in the loop performs a Gram-Schmidt process in order to orthogonalize vector u_{j+1} with respect to the columns of V_j, that is, the vectors $v_1, v_2, \ldots, v_j$ (see SLEPc Technical Report STR-1, "Orthogonalization Routines in SLEPc", for details about Gram-Schmidt). In this operation, j Fourier coefficients are computed. In exact arithmetic, the first $j-2$ coefficients are zero, and therefore the corresponding operations need not be carried out (orthogonality with respect to the first $j-2$ vectors is automatic). The other two coefficients are β_j and α_j. Note that, according to Paige, the β_j computed in this operation should be discarded and, instead, use the value $\|u_j\|_2$ computed in the previous iteration. From the point of view of orthogonalization, Algorithm 1 performs a Modified Gram-Schmidt step with just vectors v_{j-1} and v_j, while computing α_j as $v_j^* A v_j$ would correspond to Classical Gram-Schmidt. In the following, we will center our discussion on Algorithm 2 because orthogonalization will be a key aspect of robust Lanczos variants that cope with loss of orthogonality.

As in the case of Arnoldi, since $V_m^* v_{m+1} = 0$ by construction, then by premultiplying Eq. 4 by V_m^*

$$V_m^* A V_m = T_m, \tag{5}$$

that is, matrix T_m represents the orthogonal projection of A onto the Krylov subspace, and this fact allows us to compute Rayleigh-Ritz approximations of the eigenpairs of A. Let (λ_i, y_i) be an eigenpair of matrix T_m, then the Ritz value, λ_i, and the Ritz vector, $x_i = V_m y_i$, can be taken as approximations of an eigenpair of A. Typically, only a small percentage of the m approximations are good. This can be assessed by means of the residual norm for the Ritz pair, which turns out to be very easy to compute:

$$\|Ax_i - \lambda_i x_i\|_2 = \|AV_m y_i - \lambda_i V_m y_i\|_2 = \|(AV_m - V_m T_m) y_i\|_2 = \beta_{m+1} |e_m^* y_i|. \tag{6}$$

The only difference with respect to Arnoldi is that, in Lanczos, T_m is a symmetric tridiagonal matrix and, therefore, there exist more possible methods for computing its eigenpairs.

2.2 Lanczos in Finite Precision Arithmetic

When implemented in finite precision arithmetic, the Lanczos algorithm does not behave as expected. The eigenvalues of the tridiagonal matrix T_j (the Ritz values) converge very rapidly to well-separated eigenvalues of matrix A, typically those in the extreme of the spectrum. However, if enough iterations of the algorithm are carried out, then multiple copies of these Ritz values appear, beyond the multiplicity of the corresponding eigenvalue in A. In addition, the process gives wrong Ritz values as converged, which are usually called spurious eigenvalues. It can be easily seen that this unwanted behaviour appears at the same time that the Lanczos vectors start to lose mutual orthogonality. Lanczos was already aware of this problem and suggested to explicitly reorthogonalize the new Lanczos vector with respect to all the previous ones at each step. Although effective, this costly operation seems to invalidate all the appealing properties of the algorithm. Other alternatives, discussed below, have been proposed in order to be able to deal with loss of orthogonality at less cost.

Full Orthogonalization. The simplest cure for loss of orthogonality is to orthogonalize vector u_{j+1} explicitly with respect to all the previously computed Lanczos vectors. That is, as shown in Algorithm 2, performing the computation for all vectors, including the first $j-2$ ones for which the Fourier coefficient is zero in exact arithmetic.

The main advantage of full orthogonalization is its robustness, since orthogonality is maintained to full machine accuracy. (Note that for this to be true it may be necessary to resort to double orthogonalization, see report STR-1 for details.) Two drawbacks of this technique are that all Lanczos vectors need to be kept in main memory (not in secondary storage as some legacy implementations did) and that the cost of orthogonalization is high and increases as more Lanczos steps are carried out. These reasons call for a restarted version, in which the number of Lanczos vectors is bounded, see section 2.3. Another drawback appears when doing a parallel implementation of the method, in which case performance would be bad if a modified Gram-Schmidt variant is employed for the orthogonalization (again, see report STR-1 for details).

Local Orthogonalization. The quest for more efficient solutions to the problem of loss of orthogonality started with a better theoretical understanding of the Lanczos process in finite precision arithmetic, unveiled by Paige's work [1972; 1976; 1980]. One key aspect of Paige's analysis is that Lanczos vectors start to lose orthogonality as soon as an eigenvalue of T_j stabilizes or, in other words, when a Ritz value is close to convergence, causing the subsequent Lanczos vectors to contain a non-negligible component in the direction of the corresponding Ritz vector. A remarkable aspect of the analysis is that, until this situation occurs, the Lanczos algorithm with local orthogonalization (that is, if vector u_{j+1} is orthogonalized only with respect to v_j and v_{j-1}) computes the same quantities as the variant with full orthogonalization. This fact suggests different strategies for avoiding loss of orthogonality:

- To proceed with local orthogonalization until an eigenvalue of T_j has stabilized, then start a new Lanczos process with a different initial vector. This strategy was pointed out by Paige, but without specifying practical ways of doing this automatically.

- To proceed with local orthogonalization until an eigenvalue of T_j has stabilized, then continue the Lanczos process with the introduction of some kind of reorthogonalization. This gave way to the development of semiorthogonal Lanczos methods, discussed below.

A completely different approach is to simply ignore loss of orthogonality and perform only local orthogonalization at every Lanczos step. This technique is obviously the cheapest one, but has several important drawbacks. For one thing, convergence of new Ritz values is much slower since multiple copies of already converged ones keep on appearing again and again. This makes it necessary to carry out many Lanczos steps to obtain the desired eigenvalues, in some cases n steps or even many more depending on the number of wanted eigenvalues, where n is the order of matrix A. On the other hand, there is the problem of determining the correct multiplicity of the computed eigenvalues as well as discarding those which are spurious. A clever technique for doing this was proposed by Cullum and Willoughby [1985]. An eigenvalue of T_j is identified as being spurious if it is also an eigenvalue of the matrix T_j', which is constructed by deleting the first row and column of T_j. Furthermore, good eigenvalues are accepted only after they have been replicated at least once. Finally, another disadvantage of this method is that, if eigenvectors are also required, then they have to be computed afterwards with techniques such as the inverse iteration, since the Lanczos vectors are not available.

Although the above technique may seem less powerful than more elaborate alternatives, in some applications it may still be competitive, see for instance [Elsner *et al.*, 1999].

Semiorthogonal Techniques. As mentioned above, the idea of these techniques is to proceed with local orthogonalization until an eigenvalue of T_j has stabilized, then continue the Lanczos process with the introduction of some kind of reorthogonalization. Two aspects are basic in this context:

1. How to carry out the orthogonalization so that the overall cost is less than in the full orthogonalization variant.

2. How to determine when an eigenvalue has stabilized or, alternatively, how to monitor loss of orthogonality, without incurring a high cost.

With respect to the first aspect, several different approaches have been proposed: selective [Parlett and Scott, 1979], periodic [Grcar, 1981], and partial [Simon, 1984b] reorthogonalization. In brief, they consist, respectively, in the following:

- (selective) orthogonalize every Lanczos vectors with respect to all nearly converged Ritz vectors;
- (periodic) orthogonalize u_{j+1} and u_{j+2} with respect to all the Lanczos vectors; and
- (partial) orthogonalize u_{j+1} and u_{j+2} with respect to a subset of the Lanczos vectors.

The second aspect can be addressed in several ways. One of them is to compute the error bounds associated to the Ritz pairs at each Lanczos step. This operation is quite costly because all the eigenvectors of T_j have to be computed from scratch every time. Methods have been proposed for cheaply updating the eigenvalues of T_j in order to detect their stabilization, [Parlett and Reid, 1981; Parlett and Nour-Omid, 1985]. Another alternative is to use a recurrence for estimating a bound of the level of orthogonality. If we define the level of orthogonality at the j-th Lanczos step as

$$\omega_j \equiv \max_{1 \leq k < j} |\omega_{j,k}| \,, \quad \text{with} \quad \omega_{j,k} \equiv v_j^* v_k \,, \tag{7}$$

then the full reorthogonalization technique keeps it at roundoff level in each step, $\omega_j \approx \epsilon$. However, all that effort is not necessary since, as shown in [Simon, 1984b,a], maintaining semiorthogonality, i.e. $\omega_j \approx \sqrt{\epsilon}$, is sufficient so that properties of the Rayleigh-Ritz projection are still valid.

Among the different proposed recurrences, the most appropriate seems to be the following one used in [Simon, 1984b] to estimate the values of $\omega_{j,k}$ for $k = 1, 2, \ldots, j$:

$$\omega_{k,k} = 1 \,, \tag{8}$$

$$\omega_{k,k-1} = \psi_k \,, \quad \text{with} \;\; \omega_{k,0} \equiv 0 \,, \tag{9}$$

$$\omega_{j+1,k} = \frac{1}{\beta_{j+1}} \left[\beta_{k+1}\omega_{j,k+1} + (\alpha_k - \alpha_j)\omega_{j,k} + \beta_k \omega_{j,k-1} - \beta_j \omega_{j-1,k} \right] + \vartheta_{j,k} \,, \tag{10}$$

where ψ_k and $\vartheta_{j,k}$ represent roundoff errors.

2.3 Explicit Restart

As mentioned above, restarting is intended for reducing the storage requirements and, more importantly, reducing the computational cost of orthogonalization, which grows as more Lanczos vectors become available.

Restart can be accomplished in several ways. The idea of explicit restart is to iteratively compute different m-step Lanczos factorizations (Eq. 4) with successively "better" initial vectors. The initial vector for the next Lanczos run is computed from the information available in the most recent factorization. The simplest way to select the new initial vector is to take the Ritz vector associated to the first wanted, non-converged Ritz value. This strategy is described below whereas more sophisticated approaches are postponed until next subsection.

In order for a restarted method to be effective in computing more than one eigenpair, it is necessary to keep track of already converged eigenpairs and perform some form of deflation. This is done by a technique usually called *locking*, in which vectors associated to converged eigenvalues are not modified in successive runs. Suppose that after certain Lanczos run, the first k eigenpairs have already converged to the desired accuracy, and write V_m as

$$V_m = \left[\ V_{1:k}^{(l)} \ \middle|\ V_{k+1:m}^{(a)}\ \right], \tag{11}$$

where the (l) superscript indicates locked vectors and the (a) superscript indicates active vectors. In the next Lanczos run, only $m-k$ Lanczos vectors must be computed, the active ones, and in doing this the first k vectors have to be deflated. This can be done simply by orthogonalizing every new Lanczos vector also with respect to the locked ones, as illustrated in Algorithm 3.

Algorithm 3 (Lanczos with Deflation)

```
Input: Matrix $A$, number of steps $m$, $V_k$, $T_k$ with $k < m$, and initial vector $v_{k+1}$ of norm 1
Output: $(V_m, T_m, v_{m+1}, \beta_{m+1})$ so that $AV_m - V_mT_m = \beta_{m+1}v_{m+1}e_m^*$
    For $j = k+1, \ldots, m$
        $u_{j+1} = Av_j$
        Orthogonalize $u_{j+1}$ with respect to $V_j$ (obtaining $\alpha_j$)
        $\beta_{j+1} = \|u_{j+1}\|_2$
        If $\beta_{j+1} = 0$, stop
        $v_{j+1} = u_{j+1}/\beta_{j+1}$
    end
```

Note that Algorithm 3 only computes the last $m-k$ columns of V_m and T_m, the active part of the factorization. Note also that the initial vector is v_{k+1} in this case. The operations carried out in the loop are essentially the same as in Algorithm 2, with the difference that the orthogonalization has to include necessarily the locked Lanczos vectors (deflation).

Algorithm 4 (Explicitly Restarted Lanczos)

```
Input: Matrix $A$, initial vector $v_1$, and dimension of the subspace $m$
Output: A partial eigendecomposition $AV_k = V_k\Theta_k$, with $\Theta_k = \mathrm{diag}(\theta_1, \ldots, \theta_k)$
    Normalize $v_1$
    Initialize $V_m = [v_1]$,    $k = 0$
    Restart loop
        Perform $m-k$ steps of Lanczos with deflation (Algorithm 3)
        Compute eigenpairs of $T_m$, $T_my_i = y_i\theta_i$
        Compute residual norm estimates, $\tau_i = \beta_{m+1}|e_m^*y_i|$
        Lock converged eigenpairs, update the value of $k$
        $V_m = V_m Y$
    end
```

Note that in Algorithm 4 the leading submatrix of T corresponding to locked vectors is diagonal and therefore some of the operations can be skipped for this part.

Loss of Orthogonality in the Context of Restarted Lanczos. In a restarted Lanczos method, it is also necessary to deal with loss of orthogonality. In the particular case of the simple explicit restart scheme (Algorithm 4), it is safe to use any of the techniques described in section 2, since full orthogonality of the Lanczos vectors is not required for the restart to work correctly. Only in the case of local orthogonalization, the following considerations should be made:

- Since the value of m (the largest allowable subspace dimension) is usually very small compared to n (the matrix dimension), then the heuristics suggested by Cullum and Willoughby [1985] cannot be applied. Therefore, another technique should be used in order to discard spurious eigenvalues as well as redundant duplicates. One possible approach is to explicitly compute the residual norm for every converged eigenpair, then from the correct values accept only the first replica (this is explained in more detail in section 3).
- The restart vector has to be explicitly orthogonalized with respect to the locked vectors.

2.4 Other Strategies for Restart

As in the case of the Arnoldi method, the simple restart scheme discussed above is very limited, especially when more than one eigenpair is sought. The problem is that the vector chosen to be the initial vector in the next restart does not represent well the whole currently approximated invariant subspace. Better than this is to employ a *filtering* strategy that incorporates directions of all wanted eigenvectors while trying to eliminate directions in the unwanted ones. This is discussed more in detail in SLEPc Technical Report STR-4, "Arnoldi Methods in SLEPc". Here, we mention some of the techniques proposed for the particular case of symmetric Lanczos.

Polynomial filtering. In contrast to non-symmetric problems, polynomial filtering (i.e. to take the initial vector to be $v_{k+1} = z/\|z\|_2$ where $z = p(A)z_0$, being p a polynomial of degree d and z_0 a linear combination of the approximate eigenvectors) in symmetric problems can be more effective because building optimal Chebyshev polynomials is rather straightforward. However, this technique is generally more expensive than the alternatives mentioned below.

Implicit restart. Implicit restart consists in combining the Lanczos process with the implicitly shifted QR algorithm. An m-step Lanczos factorization is compacted into an $(m-d)$-step Lanczos factorization, which is then extended again to an m-step one. In this process, the relevant eigeninformation from the large factorization is kept in the small factorization. In the case of symmetric problems, it is possible to select the accelerating polynomial by using different values of the shifts in the QR process, for example Leja points. It can be shown that Leja shifts can lead to faster convergence of implicitly restarted Lanczos [Calvetti *et al.*, 1994; Baglama *et al.*, 1996].

Thick restart. Implicit restart may have stability problems unless its implementation incorporates some rather sophisticated numerical techniques. It can be shown that, in the case of symmetric problems, there is a mathematically equivalent yet simpler way to achieve the same effect. The method is called thick-restart Lanczos [Wu and Simon, 2000]. Instead of restarting with a set of modified Lanczos vectors, thick restart works directly with Ritz vectors, resulting in a much simpler implementation.

Krylov-Schur. The Krylov-Schur method [Stewart, 2001a] employs a restarting technique that is very similar to the thick restart idea. The advantage is that it provides a general restarting framework that can be applied also in the non-symmetric case. Since this method is also implemented in SLEPc, its description is provided separately (see SLEPc Technical Report STR-7, "Krylov-Schur Methods in SLEPc").

2.5 Other Variants

This subsection describes very briefly some variations of the algorithm that may be of interest in some situations.

B-Lanczos. When addressing a generalized eigenvalue problem, $Ax = \lambda Bx$, symmetry is lost because the algorithm has to work with matrix $T_S = B^{-1}A$ or similar expressions such as $T_{SI} = (A - \sigma B)^{-1}B$ in the case of the shift-and-invert spectral transformation.

However, in the case of symmetric positive-definite matrix pairs, symmetry can be recovered by replacing the standard Hermitian inner product, $\langle x, y\rangle = y^*x$, by the B-inner product, $\langle x, y\rangle_B = y^*B\,x$. It can be shown that the operator matrix (T_S or T_{SI}) is self-adjoint with respect to this inner product. When the Lanczos method is applied to the operator T_S (similarly for T_{SI}), the relations shown in section 2 can be rewritten as

$$AV_m - BV_mT_m = \beta_{m+1}Bv_{m+1}e_m^*, \tag{12}$$

and

$$V_m^*AV_m = T_m, \tag{13}$$

assuming that, in this case, the Lanczos vectors are not orthonormal but B-orthonormal, i.e. $V_m^*BV_m = I_m$. Finally, the residual norms satisfy

$$\|Ax_i - \lambda_i Bx_i\|_{B^{-1}} = \beta_{m+1}|e_m^*y_i|. \tag{14}$$

In the context of Algorithm 3, this can be accomplished by doing B-orthogonalization and replacing the 2-norm with the B-norm, $\|w\|_B = \sqrt{\langle w, w\rangle_B}$. This modified algorithm is usually referred to as B-Lanczos. For details, see [Bai *et al.*, 2000, §5.5] or [van der Vorst, 1982].

In the context of the B-Lanczos process, the techniques for maintaining semi-orthogonality described in subsection 2.2 can still be applied, see [Parlett, 1992].

In the particular case that B is singular, special care must be taken in order to prevent eigenvector corruption. This can be avoided with a simple technique called purification, see [Nour-Omid *et al.*, 1987].

Block Methods. A block Krylov method tries to extract approximate spectral information from a block Krylov subspace

$$\mathcal{K}_m(A, V_1) = \text{span}\{V_1, AV_1, A^2V_1, \ldots, A^{m-1}V_1\} \quad , \tag{15}$$

where V_1 has b columns. This kind of method has two main advantages. First, the convergence behavior can be less problematic in cases with multiple or clustered eigenvalues, provided that b is sufficiently large. Second, computational efficiency may be better in some situations, because access to the elements of matrix A is amortized by multiplying several vectors and not just one.

Many variants of block Lanczos methods have been proposed, starting from a straightforward block generalization of the simple algorithm with full orthogonalization [Golub and Underwood, 1977], incorporating some semi-orthogonality technique [Grimes *et al.*, 1994], or even with implicit restart [Baglama *et al.*, 2003]. The main issue with block Lanczos algorithms is how to deflate single vectors within a block. This can be simplified significantly with a different class of variants called band Lanczos [Bai and Freund, 2001].

2.6 Available Implementations

Many implementations of Lanczos are freely available in the form of subroutine libraries. A complete list can be found in SLEPc Technical Report STR-6, “A Survey of Software for Sparse Eigenvalue Problems”, with references and pointers to software download. Of particular interest are those libraries with parallel capabilities, such as ARPACK, BLZPACK, and TRLAN. All these have been integrated in SLEPc by means of a wrapper. A short description of these packages can be found in the SLEPc Users Manual, together with an indication of how to use them from SLEPc.

3 The SLEPc Implementation

A symmetric Lanczos solver is available in SLEPc since version 2.3.0. This implementation is based on Algorithm 4 and incorporates all the different alternatives for dealing with loss of orthogonality presented in subsection 2.2. Also, the B-Lanczos process mentioned in section 2.5 is also incorporated through the general SLEPc framework for spectral transformations, see the SLEPc Users Manual for details.

3.1 The Algorithm

The particular implementation of Lanczos provided by SLEPc is sketched in Algorithm 5. The differences with respect to Algorithm 4 are highlighted with (*).

Algorithm 5 (Explicitly Restarted Lanczos with Different Orthogonalizations)

```
Input: Matrix $A$, initial vector $v_1$, and dimension of the subspace $m$
Output: A partial eigendecomposition $AV_k = V_k\Theta_k$, with $\Theta_k = \mathrm{diag}(\theta_1, \ldots, \theta_k)$
    Normalize $v_1$
    Initialize $V_m = [v_1]$,    $k = 0$
    Restart loop
        For $j = k+1, \ldots, m$
            $u_{j+1} = Av_j$
            (*) Orthogonalize $u_{j+1}$ with respect to $[V_k, v_{j-1}, v_j]$ (obtaining $\alpha_j$)
            (*) Determine a set $S$ of Lanczos vectors
            (*) Orthogonalize $u_{j+1}$ with respect to $S$
            $\beta_{j+1} = \|u_{j+1}\|_2$       (if $\beta_{j+1} = 0$, stop)
            $v_{j+1} = u_{j+1}/\beta_{j+1}$
        end
        Compute eigenpairs of $T_m$, $T_m y_i = y_i \theta_i$
        Compute residual norm estimates, $\tau_i = \beta_{m+1}|e_m^* y_i|$
        (*) Check for spurious eigenvalues
        Lock converged eigenpairs, update the value of $k$
        $V_m = V_m Y$
    end
```

The determination of S differs depending on the orthogonalization strategy (local, full, selective, periodic or partial) selected by the user (subsection 3.2 explains how to do the selection).

- In local orthogonalization, $S = \varnothing$ so the second orthogonalization is not carried out.
- Full orthogonalization is equivalent to $S = [v_{k+1}, v_{k+2}, \ldots, v_{j-2}]$. In the context of parallel execution the two orthogonalizations shown in Algorithm 5 are actually performed as a single operation.
- In the case of periodic and partial orthogonalization, the recurrence for estimating ω_j defined in subsection 2.2 is computed at every Lanczos step. Note that the cost of this computation is negligible.
- In selective orthogonalization, the SLEPc implementation explicitly computes the eigenvalues and eigenvectors of the tridiagonal matrix T_k, together with the associated residual norm estimates, at every Lanczos step. Note that this can be computationally rather expensive for moderately large k.

The check for spurious eigenvalues shown at the end of Algorithm 5 is required only in the case of local orthogonalization. The strategy is the following. Eigenvalues that are repeated in the tridiagonal matrix T_m of the current restart are simply discarded, assuming that if they are genuine repetitions they will arise in the subsequent restart. For the rest of eigenvalues, the associated true residual norm $\|Ax_i - \lambda_i x_i\|_2$ is explicitly computed to guarantee that only those eigenpairs whose norm is within the tolerance are accepted.

For a comprehensive comparison of the different orthogonalization strategies implemented in SLEPc, both in terms of performance and numerical robustness, the reader is referred to Hernandez *et al.* [2007].

3.2 User Options

The only option specific to the Lanczos solver that the user can set is the orthogonalization strategy. The following function allows the programmer to change the selection from the source code of the application:

```
EPSLanczosSetReorthog(EPS eps,EPSLanczosReorthogType reorthog)
```

where the `reorthog` argument is an enumerate that can adopt values indicating the different orthogonalization techniques. The corresponding command-line key is `-eps_lanczos_orthog` with one of the following values: `local`, `full`, `selective`, `periodic`, `partial`, and `delayed`. The last one is similar to `full` but with delayed reorthogonalization activated (for details see SLEPc Technical Report STR-1, "Orthogonalization Routines in SLEPc").

The default value is `local` since it is usually the fastest variant and generally gives good numerical results. In order to maximize the numerical robustness, `full` should be used.

3.3 Known Issues and Applicability

The Lanczos solver in SLEPc is currently only implemented in its symmetric variant and therefore cannot be applied to non-symmetric problems. A non-symmetric Lanczos algorithm could be implemented in the future.

Supported problem types	`EPS_HEP`, `EPS_GHEP`
Allowed portion of the spectrum	All
Support for complex numbers	Yes

References

Baglama, J., D. Calvetti, and L. Reichel (1996). Iterative Methods for the Computation of a Few Eigenvalues of a Large Symmetric Matrix. *BIT*, 36(3):400–421.

Baglama, J., D. Calvetti, and L. Reichel (2003). IRBL: An Implicitly Restarted Block-Lanczos Method for Large-Scale Hermitian Eigenproblems. *SIAM J. Sci. Comput.*, 24(5):1650–1677.

Bai, Z., J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst (eds.) (2000). *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Bai, Z. and R. W. Freund (2001). A Symmetric Band Lanczos Process Based on Coupled Recurrences and Some Applications. *SIAM J. Sci. Comput.*, 23:542–562.

Calvetti, D., L. Reichel, and D. C. Sorensen (1994). An Implicitly Restarted Lanczos Method for Large Symmetric Eigenvalue Problems. *Electron. Trans. Numer. Anal.*, 2:1–21.

Cullum, J. K. and R. A. Willoughby (1985). *Lanczos Algorithms for Large Symmetric Eigenvalue Computations. Vol. 1: Theory*. Birkhaüser, Boston, MA. Reissued by SIAM, Philadelphia, 2002.

Elsner, U., V. Mehrmann, F. Milde, R. A. Römer, and M. Schreiber (1999). The Anderson Model of Localization: A Challenge for Modern Eigenvalue Methods. *SIAM J. Sci. Comput.*, 20(6):2089–2102.

Golub, G. H. and R. Underwood (1977). The Block Lanczos Method for Computing Eigenvalues. In *Mathematical Software III* (edited by J. Rice), pp. 364–377. Academic Press, New York, NY, USA.

Grcar, J. F. (1981). Analyses of the Lanczos Algorithm and of the Approximation Problem in Richardson's Method. Technical Report 1074, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois.

Grimes, R. G., J. G. Lewis, and H. D. Simon (1994). A Shifted Block Lanczos Algorithm for Solving Sparse Symmetric Generalized Eigenproblems. *SIAM J. Matrix Anal. Appl.*, 15(1):228–272.

Hernandez, V., J. E. Roman, and A. Tomas (2007). Evaluation of Several Variants of Explicitly Restarted Lanczos Eigensolvers and their Parallel Implementations. In *High Performance Computing for Computational Science - VECPAR 2006*, volume 4395 of *Lect. Notes Comp. Sci.*, pp. 403–416.

Komzsik, L. (2003). *The Lanczos Method: Evolution and Application*. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Lanczos, C. (1950). An Iteration Method for the Solution of the Eigenvalue Problem of Linear Differential and Integral Operators. *J. Res. Nat. Bur. Standards*, 45:255–282.

Meurant, G. and Z. Strakos (2006). The Lanczos and Conjugate Gradient Algorithms in Finite Precision Arithmetic. *Acta Numerica*, 15:471–542.

Nour-Omid, B., B. N. Parlett, T. Ericsson, and P. S. Jensen (1987). How to Implement the Spectral Transformation. *Math. Comp.*, 48(178):663–673.

Paige, C. C. (1972). Computational Variants of the Lanczos Method for the Eigenproblem. *J. Inst. Math. Appl.*, 10:373–381.

Paige, C. C. (1976). Error Analysis of the Lanczos Algorithm for Tridiagonalizing a Symmetric Matrix. *J. Inst. Math. Appl.*, 18(3):341–349.

Paige, C. C. (1980). Accuracy and Effectiveness of the Lanczos Algorithm for the Symmetric Eigenproblem. *Linear Algebra Appl.*, 34:235–258.

Parlett, B. N. (1980). *The Symmetric Eigenvalue Problem*. Prentice-Hall, Englewood Cliffs, NJ. Reissued with revisions by SIAM, Philadelphia, 1998.

Parlett, B. N. (1992). The Rewards for Maintaining Semi-Orthogonality among Lanczos Vectors. *Numer. Linear Algebra Appl.*, 1(2):243–267.

Parlett, B. N. and B. Nour-Omid (1985). The Use of a Refined Error Bound When Updating Eigenvalues of Tridiagonals. *Linear Algebra Appl.*, 68:179–219.

Parlett, B. N. and J. K. Reid (1981). Tracking the Progress of the Lanczos Algorithm for Large Symmetric Eigenproblems. *IMA J. Numer. Anal.*, 1(2):135–155.

Parlett, B. N. and D. S. Scott (1979). The Lanczos Algorithm with Selective Orthogonalization. *Math. Comp.*, 33:217–238.

Saad, Y. (1992). *Numerical Methods for Large Eigenvalue Problems: Theory and Algorithms*. John Wiley and Sons, New York.

Simon, H. D. (1984a). Analysis of the Symmetric Lanczos Algorithm with Reorthogonalization Methods. *Linear Algebra Appl.*, 61:101–132.

Simon, H. D. (1984b). The Lanczos Algorithm with Partial Reorthogonalization. *Math. Comp.*, 42(165):115–142.

Stewart, G. W. (2001a). A Krylov–Schur Algorithm for Large Eigenproblems. *SIAM J. Matrix Anal. Appl.*, 23(3):601–614.

Stewart, G. W. (2001b). *Matrix Algorithms. Volume II: Eigensystems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

van der Vorst, H. A. (1982). A Generalized Lánczos Scheme. *Math. Comp.*, 39(160):559–561.

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