RAL-TR-2000-011

The design of a block rational Lanczos code with partial reorthogonalization and implicit restarting

Karl Meerbergen Jennifer Scott

ABSTRACT

We discuss the design and development of a new Fortran code `EA16` for the computation of selected eigenvalues and eigenvectors of large-scale real symmetric eigenvalue problems. `EA16` can be used for either the standard or the generalized eigenvalue problem. The underlying method used by `EA16` is the block Lanczos method with partial reorthogonalization plus implicit restarting, combined with purging and locking of converged Ritz pairs. A spectral transformation may optionally be used. The code allows a change of pole via the rational Lanczos method. Particular attention is paid to the solution of generalized eigenvalue problems with a singular mass matrix.

Keywords: eigenvalue problem, real symmetric, generalized, large-scale, software, Lanczos method, rational Lanczos method

---

Computational Science and Engineering Department
Atlas Centre
Rutherford Appleton Laboratory
Oxon OX11 0QX

April 19, 2000

Contents

1 Introduction

This paper discusses the design and development of a new Fortran 77 code `EA16` from the Harwell Subroutine Library (HSL 2000) for the computation of a number of selected eigenvalues and corresponding eigenvectors of either the large scale standard eigenvalue problem

$$Ax = \lambda x \ , \tag{1}$$

where A is an $n \times n$ real symmetric matrix, or the generalized eigenvalue problem

$$Kx = \lambda M x \ , \tag{2}$$

where K and M are $n \times n$ real symmetric matrices, and either M or K is positive (semi) definite. If K is positive (semi) definite the generalized eigenvalue problem (2) is known as the *buckling problem* (Grimes, Lewis and Simon 1994). We call λ an eigenvalue, x a corresponding eigenvector and (λ, x) an eigenpair. Applications of the form (1) arise in quantum physics and chemistry while (2) arises in structural analysis (Grimes, Lewis and Simon 1986, Grimes et al. 1994), acoustics (Pierce 1981), and the stability analysis of Stokes problems (Malkus 1981).

The underlying method used by `EA16` is the block Lanczos method equipped with implicit restarting (Sorensen 1992), partial and external selective reorthogonalization (Grimes et al. 1994), a spectral transformation (Ericsson and Ruhe 1980), and rational Krylov (Ruhe 1998, Meerbergen 1999). Most software packages offer only a few of these features. In this respect, `EA16` is a general purpose state of the art eigensolver.

As we will see in §4, the solution of specialized applications sometimes require a number options to be specified when using `EA16`. We will later provide drivers for a range of applications that will set many of these options, thus providing the user with a more straightforward interface to `EA16`.

The software uses reverse communication for the action of the matrices A, K and M, or the spectral transformations $(A \Leftrightarrow \sigma I)^{-1}$, $(K \Leftrightarrow \sigma M)^{-1}M$, and $(K \Leftrightarrow \sigma M)^{-1}K$, on sets of vectors. `EA16` is particularly effective when the user is able to provide an efficient code for the product of one or more of these matrices with a number of vectors equal to the blocksize of the Lanczos method. In particular, the user may be able to take advantage of BLAS 3 kernels (Dongarra, DuCroz, Duff and Hammarling 1990) to significantly enhance the performance when using a blocksize greater than 1.

The paper is organized as follows. In §2, we describe the theory behind the Lanczos algorithm, including locking of converged eigenvectors, purging of unwanted eigenvectors, implicit restarting, and partial and external selective reorthogonalization. Concepts such as purification and implicit filtering are introduced in order to improve the robustness of the Lanczos method for problems with a singular mass matrix. In §3, we compare `EA16` with existing Lanczos codes and discuss the user interface and options offered by `EA16`. §4 looks at a number of applications, their properties and how `EA16` can be used for their solution.

We complete this section by introducing notation that will be used throughout this paper. We use lower case symbols to denote scalars and vectors while upper case is reserved for matrices.

n	Dimension of the eigenvalue problem.
A, K, M	Matrices from the eigenvalue problem.
λ	Eigenvalue of original problem.
	Also used to denote computed Ritz value of original problem.
x	Eigenvector. Also used to denote computed Ritz vector

r	Residual vector.
ν, $\nu(A)$	The number of negative eigenvalues of A.
S	Shift-invert transformation $S = (A \Leftrightarrow \sigma I)^{-1}$ or $(K \Leftrightarrow \sigma M)^{-1}M$.
S_B	Buckling transformation $S_B = (K \Leftrightarrow \sigma M)^{-1}K$.
θ	Eigenvalue or Ritz value of shift-invert transformation or buckling transformation.
σ	Pole of the shift-invert or buckling transformation.
$x^T y$	Standard inner product.
$x^T M y$	M inner product.
$\langle x, y \rangle$	Abstract notation for inner product. Depending on the context $\langle x, y \rangle = x^T y$ or $x^T M y$.
$\langle X, Y \rangle$	Abstract notation for 'inner' product of two matrices $X \in \mathbf{R}^{n \times k}$ and $Y \in \mathbf{R}^{n \times l}$; $\langle X, Y \rangle \in \mathbf{R}^{k \times l}$. Depending on the context $\langle X, Y \rangle = X^T Y$ or $X^T M Y$. Note that $\langle XP, YQ \rangle = P^T \langle X, Y \rangle Q$
$\|x\|$, $\|x\|_2$	2-norm of x : $\sqrt{x^T x}$. The subscript is omitted when there is no ambiguity.
$\|x\|_M$	M-norm of x : $\sqrt{x^T M x}$.
$\|X\|$	2-norm of matrix X : largest singular value of X.
$\|X\|_F$	Frobenius norm of X.
u	Machine precision.
b	The blocksize used by the Lanczos method.
T_k	$kb \times kb$ tridiagonal matrix for the block Lanczos method with block size b.
$\underline{T}_k$	$kb + b \times kb$ tridiagonal matrix with the residual terms included.
B_j	jth $b \times b$ subdiagonal block in the tridiagonal matrix T_k.
C_j	jth $b \times b$ diagonal block in the tridiagonal matrix T_k.
V_j	$n \times b$ block of Lanczos vectors computed at iteration $j \Leftrightarrow 1$.
$\mathcal{V}_j$	$n \times jb$ matrix $[V_1, \ldots, V_j]$.
$.^+$	The superscript $.^+$ is used after an implicit restart or change of pole.
$\perp$	Orthogonality : $x \perp y \Leftrightarrow \langle x, y \rangle = 0$.
I	Identity matrix. Dimension clear from the context.
$\underline{I}_k$	$kb + b \times kb$ matrix with 1's on main diagonal and all other entries 0.
E_k	$kb \times b$ matrix with zero except ones on the main diagonal in the lower $b \times b$ block.

2 Basic theory

In this section, we present the major theory upon which the block Lanczos code `EA16` is built. We introduce the block Lanczos process in §2.1 and in §2.2, we discuss the use of purging to push unwanted eigenvectors out of the Krylov space and the locking of converged eigenvectors to prevent sought-after eigenpairs disappearing from the Lanczos basis due to round-off errors.

The Lanczos method converges quickly to well-separated extremal eigenvalues, that is, to eigenvalues lying at both ends of the spectrum. In many applications, the sought-after eigenvalues are neither extremal nor well-separated. Working with a large Krylov space is not an option for large-scale problems because storing all the basis vectors requires too much memory. An efficient way of dealing with this problem is to use implicit restarting (§2.3). Another option is to try and improve the speed of convergence by using a spectral transformation (§2.4). `EA16` uses implicit restarting optionally combined with a spectral transformation.

Some codes (see §3.1) do not store the Lanczos basis vectors and do not use

reorthogonalization. The advantages of this are that large Krylov subspaces can be built, avoiding the need for restarting. However, the disadvantages are that spurious eigenvalues may appear and multiplicities may not be found correctly. Such codes have tests for the detection of spurious eigenvalues.

In §2.5, we briefly discuss the need to use harmonic Ritz values. The numerically stable orthogonalization of the Lanczos basis vectors is vital for obtaining orthogonal eigenvector estimates and for finding multiple eigenvalues; this is considered in §2.6. In §2.7, we look at how `EA16` attempts to avoid breakdown when the mass matrix is singular.

2.1 The block Lanczos process

In this section, we present a block Lanczos method for the standard eigenvalue problem (1) and show how to compute eigenvalues and eigenvectors. The Lanczos method was originally proposed by Lanczos (1950). The block Lanczos method was developed for computing clustered eigenvalues (Cullum and Donath 1974, Golub and Underwood 1977, Scott 1979, Cullum and Willoughby 1985*a*, Cullum and Willoughby 1985*b*, Grimes et al. 1986, Grimes et al. 1994).

Starting from an orthonormal set of b initial vectors $V_1 \in \mathbf{R}^{n\times b}$, the Lanczos method builds an orthonormal basis for the Krylov space

$$\mathcal{K}_{k,b} = \text{span}\{V_1, AV_1, A^2V_1, \ldots, A^{k-1}V_1\} \ .$$

We call b the blocksize. The following algorithm shows how to compute the basis.

Algorithm 2.1 (Block Lanczos method)

1. *Let $V_1 = [v_1, \ldots, v_b]$ be a given set of initial vectors with $\langle V_1, V_1\rangle = I$.*
   Formally let $B_0 = 0$ and $V_0 = 0$.
2. *For $j = 1 : k$ do*
   - 2.1. *Form $W_j = AV_j$.*
   - 2.2. *Form $W_j' = W_j \Leftrightarrow V_{j-1}B_{j-1}^T$.*
   - 2.3. *Form $C_j = \langle V_j, W_j'\rangle$.*
   - 2.4. *Form $W_j'' = W_j' \Leftrightarrow V_jC_j$.*
   - 2.5. *Factorize $W_j'' = V_{j+1}B_j$ so that $\langle V_{j+1}, V_{j+1}\rangle = I$ and B_j is upper triangular.*

Here $\langle\cdot,\cdot\rangle$ denotes the standard inner product. Steps 2.2–2.5 form a block modified Gram-Schmidt orthogonalization of W_j against V_{j-1} and V_j. Note that only C_j and B_j are explicitly computed, since the coefficient B_{j-1} is computed in iteration $j \Leftrightarrow 1$ using the QR factorization of W_j''. In order to improve the numerical stability, reorthogonalization of the Lanczos vectors is required. This is discussed in §2.6.

Eliminating W_j, W_j' and W_j'' from Steps 2.1–2.5, we obtain the recurrence relation

$$AV_j = V_{j+1}B_j + V_jC_j + V_{j-1}B_{j-1}^T \ . \tag{3}$$

Setting $\mathcal{V}_j = [V_1, \ldots, V_j]$ and collecting all iterations in one matrix gives the recurrence relation of order k, defined by

$$A\mathcal{V}_k = \mathcal{V}_{k+1}\underline{T}_k, \tag{4}$$

where $\langle \mathcal{V}_{k+1}, \mathcal{V}_{k+1} \rangle = I$ and $\underline{T}_k$ is the $kb + b \times kb$ block tridiagonal Lanczos matrix

$$\underline{T}_k = \begin{bmatrix} C_1 & B_1^T & & & & & \\ B_1 & C_2 & B_2^T & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \ddots & C_{k-1} & B_{k-1}^T & \\ & & & & B_{k-1} & C_k & \\ & & & & & B_k & \end{bmatrix} .$$

Since the B_j, $j = 1, \ldots, k$ are upper triangular, $\underline{T}_k$ is a band matrix with half bandwidth $b+1$.

An approximate eigenvector $x = \mathcal{V}_k z$ is defined by a Galerkin condition so that the residual satisfies

$$Ax \Leftrightarrow \lambda x \perp \text{Range}(\mathcal{V}_k) ,$$

which is also called the Rayleigh-Ritz projection for the symmetric eigenvalue problem. This leads to the small scale eigenvalue problem

$$T_k z = \lambda z,$$

where T_k is the matrix $\underline{T}_k$ without the final b rows. The pair (λ, x) is called a Ritz pair, because it results from the Rayleigh-Ritz projection on the Krylov space. The value λ is called a Ritz value and x is a corresponding Ritz vector. The residual has the form

$$r = Ax \Leftrightarrow \lambda x = V_{k+1} B_{k+1} E_k^T z ,$$

where E_k is a $kb \times b$ matrix with zeros everywhere except ones on the main diagonal in the lower $b \times b$ block. E_k may be regarded as a generalization of the kth identity vector. The residual norm is thus given by $\|B_{k+1} E_k^T z\|$.

2.2 Purging and locking

When eigenpairs have been computed sufficiently accurately, they are removed from the recurrence relation. They are no longer improved or modified and the Lanczos method continues with a smaller tridiagonal matrix after their removal. This is called 'locking'. Unwanted Ritz pairs on the other hand can be removed from the subspace by purging.

The recurrence relation (4) is the major characteristic of the Lanczos method. It describes the action of the matrix A on the first kb Lanczos vectors. Another orthogonal basis for the Krylov space is the basis of Ritz vectors. Recall that the Ritz values and vectors are computed from $T_k Z_k = Z_k \Lambda_k$, with $\Lambda_k = \text{diag}(\lambda_1, \ldots, \lambda_k)$. Multiplying (4) on the right by Z_k, we have

$$A\mathcal{V}_k Z_k = V_{k+1} \begin{bmatrix} Z_k & \\ & I \end{bmatrix} \begin{bmatrix} Z_k & \\ & I \end{bmatrix}^T \underline{T}_k Z_k .$$

Setting $\mathcal{X}_k = \mathcal{V}_k Z_k$, we obtain

$$\begin{aligned} A\mathcal{X}_k &= [\mathcal{X}_k \;\; V_{k+1}] \mathcal{S}_k \\ \mathcal{S}_k &= \begin{bmatrix} \Lambda_k \\ B_{k+1} E_k^T Z_k \end{bmatrix} . \end{aligned} \tag{5}$$

The last b rows of $\mathcal{S}_k$ represent the residuals of the Ritz pairs. Note that $[\mathcal{X}_k \;\; V_{k+1}]$ is orthogonal.

2.2.1 Purging

Suppose we want to remove $(k \Leftrightarrow p)b$ unwanted Ritz vectors from the Krylov subspace. By ordering the entries of Λ_k and the columns of $\mathcal{X}_k = [x_1, \ldots, x_{kb}]$ so that the pb wanted Ritz pairs appear first, we can decompose (5) in the form

$$A[\mathcal{X}_p \;\; \mathcal{X}_{k-p}] = [\mathcal{X}_p \;\; \mathcal{X}_{k-p} \;\; V_{k+1}] \begin{bmatrix} \Lambda_p & 0 \\ 0 & \Lambda_{k-p} \\ B_{k+1}E_k^T Z_p & B_{k+1}E_k^T Z_{k-p} \end{bmatrix},$$

from which we have

$$A\mathcal{X}_p = [\mathcal{X}_p \;\; V_{k+1}] \begin{bmatrix} \Lambda_p \\ B_{k+1}E_k^T Z_p \end{bmatrix}. \tag{6}$$

This is a new recurrence relation of order p expressed in terms of Ritz vectors and Ritz values. This operation is called purging because $(k \Leftrightarrow p)b$ Ritz vectors are removed from the Krylov space. Equation (6) can be expressed in the form

$$A\mathcal{V}_p = \mathcal{V}_{p+1}\underline{T}_p$$

by making the transformations

$$\mathcal{V}_{p+1} = [\mathcal{X}_p Y_p \;\; V_{k+1}] \quad \text{and} \quad \underline{T}_p = \begin{bmatrix} Y_p^T & \\ & I \end{bmatrix} \begin{bmatrix} \Lambda_p \\ B_{k+1}E_k^T Z_p \end{bmatrix} Y_p,$$

where Y_p is a unitary matrix that is chosen so that $\underline{T}_p$ is a band matrix with half bandwidth $b+1$. Hence we arrive at a Lanczos recurrence relation of order p.

2.2.2 Locking

Let the Ritz pairs be ordered so that the first qb residual norms satisfy $\|B_{k+1}E_k^T z_j\| \leq \text{TOL}$ $(j = 1, \ldots, qb)$, where TOL is some user-defined convergence tolerance. Ritz pairs with a residual norm smaller than TOL are locked. We lock them in the recurrence relation by setting the elements $1, \ldots, q$ in the $(kb+1)$ st to $(kb+b)$ th rows of $\mathcal{S}_k$ equal to zero (see (5)). If we assume that the locked Ritz pairs are exact eigenpairs, (5) takes the form

$$A[\mathcal{X}_q \;\; \mathcal{X}_{k-q}] = [\mathcal{X}_q \;\; \mathcal{X}_{k-q} \;\; V_{k+1}] \begin{bmatrix} \Lambda_q & 0 \\ 0 & \Lambda_{k-q} \\ 0 & B_{k+1}E_k^T Z_{k-q} \end{bmatrix},$$

which can be transformed into

$$A \begin{bmatrix} \mathcal{X}_q & \tilde{V}_{p-q} \end{bmatrix} \Leftrightarrow \begin{bmatrix} \mathcal{X}_q & \tilde{V}_{p-q+1} \end{bmatrix} \begin{bmatrix} \Lambda_q & 0 \\ 0 & \underline{\tilde{T}}_{k-q} \end{bmatrix} = 0 \tag{7}$$

where $\underline{\tilde{T}}_{k-q}$ is a $(kb \Leftrightarrow qb+b) \times (kb \Leftrightarrow qb)$ block tridiagonal matrix. The Lanczos method can be continued from this point.

Locking and purging reduce the dimension of the Krylov subspace and the tridiagonal matrix which makes the manipulation of the Lanczos vectors and the tridiagonal matrix cheaper. Because the Ritz pairs are not exact eigenpairs, the right-hand side of (7) is nonzero : the error in the recurrence relation is of the order of the residual tolerance TOL.

2.3 Implicitly restarted Lanczos

When k is very large, the storage of the basis vectors can become prohibitive and it may be necessary to restart the Lanczos method. An elegant restarting algorithm is the implicitly restarted Lanczos (or Arnoldi) method (Sorensen 1992, Calvetti, Reichel and Sorensen 1994). This method compresses the Lanczos basis into one of smaller dimension by throwing away a part of the subspace that is unlikely to make a significant contribution to the convergence of the wanted eigenvalues. After the compression, the Lanczos method can add new Lanczos vectors to the basis. Note that purging (§2.2) can be used for this purpose, but the approach discussed here relies on the implicitly shifted QR method.

2.3.1 Implicitly shifted QR

The following theorem shows the major implication of applying an implicit QR step to the block tridiagonal matrix $\underline{T}_k$.

Theorem 2.1 (Sorensen 1992) *Consider the Lanczos recurrence relation*

$$A\mathcal{V}_k = \mathcal{V}_{k+1}\underline{T}_k \tag{8}$$

and the QR factorization

$$\underline{Q}_k R_k = \underline{T}_k \Leftrightarrow \sigma \underline{I} \ ,$$

where $\underline{Q}_k$ is a $kb+b \times kb$ unitary matrix and R_k is a $kb \times kb$ upper triangular matrix. Let $\underline{Q}_{k-1}$ be the matrix $\underline{Q}_k$ without the final b rows and columns. Define $\mathcal{V}_k^+ = \mathcal{V}_{k+1}\underline{Q}_k$, and $\underline{T}_{k-1}^+ = R_k\underline{Q}_{k-1} + \sigma\underline{I}$. Then

1. *$\mathcal{V}_k^+$ is unitary.*
2. *If $\underline{T}_k \Leftrightarrow \sigma\underline{I}$ is nonsingular, then* $\mathrm{Range}(\mathcal{V}_j^+) = \mathrm{Range}((A \Leftrightarrow \sigma I)\mathcal{V}_j)$ *for $j = 1, \ldots, k$.*
3. *$A\mathcal{V}_{k-1}^+ = \mathcal{V}_k^+\underline{T}_{k-1}^+$.*

Proof Shifting the recurrence relation and using the QR factorization we have

$$(A \Leftrightarrow \sigma I)\mathcal{V}_k = \mathcal{V}_{k+1}(\underline{T}_k \Leftrightarrow \sigma\underline{I}) = \underline{Q}_k R_k \ .$$

The matrix $\underline{Q}_{k-1}$ that is obtained from $\underline{Q}_k$ by removing the final b rows and columns is unitary because $\underline{Q}_k$ is block tridiagonal and has dimensions $kb \times kb \Leftrightarrow b$. Thus

$$\begin{aligned} (A \Leftrightarrow \sigma I)\mathcal{V}_k &= \mathcal{V}_{k+1}\underline{Q}_k R_k \\ (A \Leftrightarrow \sigma I)\mathcal{V}_k\underline{Q}_{k-1} &= \mathcal{V}_{k+1}\underline{Q}_k R_k \underline{Q}_{k-1} \\ A\mathcal{V}_k\underline{Q}_{k-1} &= \mathcal{V}_{k+1}\underline{Q}_k(R_k\underline{Q}_{k-1} + \sigma\underline{I}) \ . \end{aligned} \tag{9}$$

Define $\mathcal{V}_k^+ = \mathcal{V}_{k+1}\underline{Q}_k$ and $\underline{T}_{k-1}^+ = R_k\underline{Q}_{k-1} + \sigma\underline{I}$. Since $\underline{T}_{k-1}^+$ arises from a shifted QR step, $\underline{T}_{k-1}^+$ is block tridiagonal and has dimensions $kb \times kb \Leftrightarrow b$. We thus obtain a new recurrence relation

$$A\mathcal{V}_{k-1}^+ = \mathcal{V}_k^+\underline{T}_{k-1}^+ \ ,$$

which proves Statement 3. Statement 1 follows from the fact that $\mathcal{V}_{k+1}$ and $\underline{Q}_k$ are unitary. Statement 2 follows from (9) after multiplication on the right by R_k^{-1}. □

The first major conclusion of this theorem is that applying an implicit QR step to the tridiagonal matrix leads to a new Lanczos recurrence relation of order $k \Leftrightarrow 1$. Thus implicit QR steps may be used to reduce the order of the Lanczos recurrence relation, as purging also does. The second major conclusion is that the Lanczos vectors and matrix may be considered as having been computed by a Lanczos process restarted with vectors

$$V_1^+ = (A \Leftrightarrow \sigma I)V_1 \ .$$

This is why the process is called an implicit restart. Thirdly, Statement 2 shows that the new Lanczos vectors result from the application of a polynomial of degree one to the old Lanczos vectors.

The result can be generalized to more than one implicit QR step as follows. When p implicit QR steps are performed — we say that p implicit restart steps in the Lanczos method are performed — we have the following result.

- $\mathrm{Range}(V_1^+) = \mathrm{Range}(\prod_{j=1}^{p}(A \Leftrightarrow \sigma_j I)V_1)$
- $\mathrm{Range}(\mathcal{V}_{k-p+1}^+) = \mathrm{Range}(\prod_{j=1}^{p}(A \Leftrightarrow \sigma_j I)\mathcal{V}_{k-p+1})$

It follows that by choosing $k \Leftrightarrow p$ 'shifts' σ_j $(j = 1, \ldots, k \Leftrightarrow p)$, we reduce the dimension of the Krylov subspace from $kb + b$ to $pb + b$ by applying a polynomial filter with the shifts as zeros. This enables us to reduce the dimension of the subspace by filtering away unwanted directions.

By an (implicit) restart we mean the set of $k \Leftrightarrow p$ implicit restart steps to reduce a Lanczos recurrence relation of order k to one of order p.

In practice, we use the following repeated implicitly restarted Lanczos algorithm.

Algorithm 2.2 (Implicitly restarted Lanczos algorithm)

1. Choose initial vectors V_1. Choose p and k $(k > p)$.
2. Perform p steps of the Lanczos method.
3. For $l = 1 : n_{\mathrm{it}}$ do
  3.1. Perform $k \Leftrightarrow p$ steps of the Lanczos method to expand the Lanczos recurrence of order p to order k.
  3.2. Compute Ritz values, vectors and residual norms.
  3.3. Select $k \Leftrightarrow p$ shifts $\sigma_j^{(l)}$ $(j = 1, \ldots, k \Leftrightarrow p)$.
  3.4. Perform an implicit restart with these $k \Leftrightarrow p$ shifts.

In this way, the Lanczos basis is expanded and compressed until the basis converges to an invariant subspace. The Lanczos process may be considered to have been applied to the initial vector

$$\prod_{l=1}^{n_{\mathrm{it}}} \left(\prod_{j=1}^{k-p} (A \Leftrightarrow \sigma_j^{(l)} I) \right) V_1 \ . \tag{10}$$

The shifts should be chosen to ensure rapid convergence of the wanted eigenvalues. In the literature various shift strategies have been proposed. In Sorensen's original paper (Sorensen 1992), exact shifts and Chebyshev shifts are suggested. Leja shifts were introduced in order to improve the convergence speed (Calvetti et al. 1994, Baglama, Calvetti and Reichel 1998*a*), and refined shifts based on the refined Arnoldi method are used by Jia (1998). We now briefly discuss three shift choices offered by `EA16`.

2.3.2 Exact shifts

Morgan (1996) showed that when a Ritz value is used as a shift, an implicit restart step is equivalent to the purging of the corresponding Ritz vector from the Krylov space. In the early days of the implicitly restarted Arnoldi and Lanczos methods (Sorensen 1992) this choice of 'exact' shifts was advocated. To reduce the dimension of the subspace, $k \Leftrightarrow p$ 'unwanted' Ritz values are selected, usually from amongst those that lie away from the desired part of the spectrum. For example, when the eigenvalues of largest modulus are wanted, the exact shifts are chosen to be the Ritz values with smallest modulus.

2.3.3 Chebyshev shifts

If computing the eigenvalues that lie outside an interval $[\alpha, \beta]$ is of interest, the shifts can be selected so that this interval is filtered out by the polynomial. The polynomial that is smallest in the interval $[\Leftrightarrow 1, 1]$ and largest outside is the Chebyshev polynomial of the second kind. Chebyshev shifts are the zeros of the Chebyshev polynomial of degree $k \Leftrightarrow p$ shifted and scaled so that its zeros are in $[\alpha, \beta]$.

When Chebyshev shifts are used in Algorithm 2.2, (10) is the $n_{\rm it}$ th power of a Chebyshev polynomial. This polynomial is not the best polynomial of degree $n_{\rm it}(k \Leftrightarrow p)$ for filtering the interval. A better choice would be to select the shifts from amongst the zeros of the Chebyshev polynomial of degree $n_{\rm it}(k \Leftrightarrow p)$. The problem is that $n_{\rm it}$ is usually not known beforehand. Leja shifts (see below) are more appropriate since the shifts are chosen with respect to shifts chosen in previous restarts.

2.3.4 Leja shifts

Exact and Chebyshev shifts may not be optimal when the eigenvalues outside an interval are required (Calvetti et al. 1994). Leja shifts are chosen from a sequence of Leja points z_j, where $z_1 = \alpha$, $z_2 = \beta$ and z_j for $j > 2$ satisfies

$$\prod_{k=1}^{j-1} |z_j \Leftrightarrow z_k| = \max_{z \in [\alpha, \beta]} |z \Leftrightarrow z_k| \ . \tag{11}$$

The Leja points have the same limit distribution as the zeros of the Chebyshev polynomials. The difference is that when the Leja points $z_1, \ldots, z_{k-p}$ are used to restart the Lanczos recurrence relation, the points $z_{k-p+1}, \ldots, z_{2(k-p)}$ are used as shifts in the next restart. In practice, fast Leja shifts are used, which differ only from (11) in that the maximum is taken over a set of candidate Leja points. Further details are given by Baglama et al. (1998*a*).

2.4 Spectral transformation

As already noted, the Lanczos method is often the method of choice when the wanted eigenvalues are extremal and well-separated but, for many applications, the Lanczos converges slowly because other wanted eigenvalues are sought. A particular case is when interior eigenvalues are required. Furthermore, the Lanczos method cannot be used for solving generalized eigenvalue problems of the form (2) (although Scott 1981 did develop a method for solving this problem using the Lanczos process, but because it converges relatively slowly it is not used in practice).

The convergence of the Lanczos method can be substantially improved by using the *spectral transformation*

$$S = (A \Leftrightarrow \sigma I)^{-1} \ ,$$

where σ is called the *pole*. In this paper, we refer to the Lanczos method applied to A as the *regular* mode and when S is used we call it the *shift-invert* mode. The matrix S has the same eigenvectors as A, but the eigenvalues are $\theta = (\lambda \Leftrightarrow \sigma)^{-1}$. When σ lies close to the wanted eigenvalue(s), the corresponding θ's are well-separated and extremal, hence easy to compute by the Lanczos method. Once θ has been computed, λ is recovered as $\lambda = \sigma + \theta^{-1}$.

The shift-invert mode requires matrix-vector products of the form $W = (A \Leftrightarrow \sigma I)^{-1}V$ to be performed. In practice, $(A \Leftrightarrow \sigma I)^{-1}$ is never formed explicitly but W is computed by solving the linear system $(A \Leftrightarrow \sigma I)W = V$. Usually, a direct method is used as this allows $A \Leftrightarrow \sigma I$ can be factorized once and k linear solvers using the factors then needed to form the Lanczos basis vectors. Iterative methods can be used and may be essential for very large problems, but the linear systems need to be solved more accurately than the desired accuracy of the eigensolution. For recent work on the use of iterative linear solvers in eigenvalue solvers, we refer to Morgan and Scott (1993), Crouzeix, Philippe and Sadkane (1994), Sleijpen and van der Vorst (1996), Meerbergen and Roose (1997), Morgan and Meerbergen (2000).

2.4.1 The generalized eigenvalue problem

A spectral transformation is particularly useful for solving the generalized eigenvalue problem (2), since in this case linear solves cannot be avoided. Ericsson and Ruhe (1980) propose the spectral transformation

$$L^T(K \Leftrightarrow \sigma M)^{-1}L$$

with $M = LL^T$ (L lower triangular). The problem with this transformation is that the factorization of M is required.

Let the recurrence relation for the spectral transformation be

$$L^T(K \Leftrightarrow \sigma M)^{-1}L\mathcal{Q}_k = \mathcal{Q}_{k+1}T_k$$

with $\mathcal{Q}_{k+1}^T\mathcal{Q}_{k+1} = I$. With the change of basis into $\mathcal{V}_{k+1} = L^{-T}\mathcal{Q}_{k+1}$, we can rewrite this as

$$(K \Leftrightarrow \sigma M)^{-1}M\mathcal{V}_k = \mathcal{V}_{k+1}\underline{T}_k$$

with $\mathcal{Q}_{k+1}^T\mathcal{Q}_{k+1} = \mathcal{V}_{k+1}^TM\mathcal{V}_{k+1} = I$. When the Lanczos basis is M orthogonal, the Lanczos method can thus be applied to the non-symmetric shift-invert transformation $S = (K \Leftrightarrow \sigma M)^{-1}M$ (Ericsson 1986, Nour-Omid, Parlett, Ericsson and Jensen 1987). The factorization $M = LL^T$ is not required. Another way to understand this is by noting $x^TM(Sy) = (Sx)^TMy$, so that S is self-adjoint with respect to the M inner product x^TMy. In other words, the Lanczos method can be used for computing eigenpairs of S when the M inner product is used for the construction of the tridiagonal matrix. As a result, the Lanczos basis $\mathcal{V}_{k+1}$ is M orthogonal, i.e. $\mathcal{V}_{k+1}^TM\mathcal{V}_{k+1} = I$. The M orthogonal Lanczos method is given by the following algorithm.

Algorithm 2.3 (M orthogonal block Lanczos method)

1. *Let $V_1 = [v_1, \ldots, v_b]$ be a given set of initial vectors with $\langle V_1, V_1\rangle = I$*
   Let $B_0 = 0$ and $V_0 = 0$.
2. *For $j = 1 : k$ do*
   2.1.a. Form $U_j = MV_j$.
   2.1.b. Solve $(K \Leftrightarrow \sigma M)W_j = U_j$ for W_j.
   2.2. Form $W_j' = W_j \Leftrightarrow V_{j-1}B_{j-1}^T$.
   2.3. Form $C_j = \langle V_j, W_j'\rangle$.
   2.4. Form $W_j'' = W_j' \Leftrightarrow V_jC_j$.
   2.5. Factorize $W_j'' = V_{j+1}B_j$ so that $\langle V_{j+1}, V_{j+1}\rangle = I$ and B_j is upper triangular.

Note that, apart from Step 2.1, this is the same as Algorithm 2.1 but here $\langle X, Y \rangle$ denotes the M inner product $X^T M Y$.

When K is positive (semi) definite and M is indefinite the shift-invert transformation cannot be used since $x^T M y$ cannot serve as an inner product. Instead, we can use the so-called *buckling transformation* $S_B = (K \Leftrightarrow \sigma M)^{-1} K$ with the K inner product because S_B is self-adjoint with respect to this inner product.

For the generalized problem, we refer to the Lanczos method applied to $M^{-1}K$ as the *regular* mode, when $S = (K \Leftrightarrow \sigma M)^{-1} M$ is used with the M inner product we call it the *shift-invert* mode, and when $S_B = (K \Leftrightarrow \sigma M)^{-1} K$ is used with the K inner product we call it the *buckling* mode.

2.4.2 Trust intervals and pole selection

Consider two real symmetric matrices C and D such that $C = L^T D L$ with L nonsingular. Sylvester's Inertia Theorem states that C and D have the same number of positive, zero, and negative eigenvalues. These three numbers are called the *inertia* (Parlett 1980, Ericsson and Ruhe 1980, Grimes et al. 1986). We want to use the inertia or, more specifically, the number ν of negative eigenvalues of $A \Leftrightarrow \sigma I$ (or $K \Leftrightarrow \sigma M$ for the generalized problem) to help with the robust selection of poles and with the computation of trust intervals (Grimes et al. 1994).

Let ν_1 and ν_2 denote the number of negative eigenvalues of $A \Leftrightarrow \sigma_1 I$ and $A \Leftrightarrow \sigma_2 I$, respectively. Then, provided $\sigma_1 < \sigma_2$, the number of eigenvalues in the interval $[\sigma_1, \sigma_2]$ is $\nu_2 \Leftrightarrow \nu_1$. When the number of locked Ritz values in the interval is equal to $\nu_2 \Leftrightarrow \nu_1$, the interval $[\sigma_1, \sigma_2]$ is called a *trust interval*. We want our pole selection strategy to establish a trust interval around the sought-after eigenvalues.

For the generalized problem, it is clear that $\nu(K \Leftrightarrow \sigma_1 M) \Leftrightarrow \nu(K \Leftrightarrow \sigma_2 M)$ is the number of eigenvalues in the interval $[\sigma_1, \sigma_2]$ $(\sigma_1 < \sigma_2)$, and so, for a given σ, we need to be able to compute the number of negative eigenvalues of $K \Leftrightarrow \sigma M$. If M is positive (semi) definite there are a number of sparse direct solvers, including the routines `MA27` and `MA57` from the Harwell Subroutine Library (HSL 2000), that compute a factorization of the form $K \Leftrightarrow \sigma M = LDL^T$ (L unit lower triangular and D diagonal, possibly with 2×2 blocks on the diagonal) and return $\nu(K \Leftrightarrow \sigma M)$ to the user.

For the buckling problem, if K is positive definite, we can consider $K = LL^T$ and compute the number of negative eigenvalues of $L^{-1}(K \Leftrightarrow \sigma M) L^{-T}$ by factorizing $(K \Leftrightarrow \sigma M)$. The number of negative eigenvalues of $L^{-1}(K \Leftrightarrow \sigma M) L^{-T}$ is the number of negative eigenvalues of $(K \Leftrightarrow \sigma M)x = \theta K x$, which is equal to the number of eigenvalues of $Kx = \lambda M x$ between 0 and σ.

In finite precision arithmetic, the factorization $K \Leftrightarrow \sigma M = LDL^T$ is exact for the matrix $K \Leftrightarrow \sigma M + E$, where E the backward error with norm of the order of magnitude of $\|K \Leftrightarrow \sigma M\| u$ (u is the machine precision). It follows that the inertia is for $K \Leftrightarrow \sigma M + E$, rather than $K \Leftrightarrow \sigma M$. Since the eigenvalues of $K \Leftrightarrow \sigma M + E$ are perturbations of the order of at most $\|E\|$, the inertia may be wrong if $K \Leftrightarrow \sigma M$ has an eigenvalue of modulus smaller than $\|E\|$. Therefore, it is important to ensure poles are selected away from the eigenvalues of $Kx = \lambda M x$. The following lemma suggests a minimum distance between the poles and the eigenvalues.

Lemma 2.2 *Let $\lambda_{\min}$ and $\lambda_{\max}$ be the minimum and maximum eigenvalues of $Kx = \lambda M x$ respectively. If*

$$\min_{\lambda(K,M)} |\sigma \Leftrightarrow \lambda| > (\lambda_{\max} \Leftrightarrow \lambda_{\min}) \frac{u}{1 \Leftrightarrow u}$$

$\|E\| = \|K \Leftrightarrow \sigma M\| u$ and M is nonsingular, then $K \Leftrightarrow \sigma M$ and $K \Leftrightarrow \sigma M + E$ have the same number of positive eigenvalues and the same number of negative eigenvalues.

Proof Let η be an eigenvalue of $K \Leftrightarrow \sigma M$. If $\min_\eta |\eta| > \|E\|$ then $K \Leftrightarrow \sigma M$ and $K \Leftrightarrow \sigma M + E$ have the same number of positive and negative eigenvalues. The proof follows by replacing E by ϵE and letting ϵ evolve from 0 to 1.

Let η be the eigenvalue of $K \Leftrightarrow \sigma M$ with smallest modulus, then

$$\begin{aligned} \eta &= \min_{\|x\|=1} \|(K \Leftrightarrow \sigma M)x\| \\ &\geq \min_{\lambda(K,M)} |\lambda \Leftrightarrow \sigma| / \|M^{-1}\| \ . \end{aligned}$$

We want $\eta > \|E\|$, which is satisfied when $\min_{\lambda(K,M)} |\lambda \Leftrightarrow \sigma| / \|M^{-1}\| > \|E\| \equiv \|K \Leftrightarrow \sigma M\| u$. It is also satisfied provided

$$\min_{\lambda(K,M)} |\lambda \Leftrightarrow \sigma| / \|M^{-1}\| > \max_{\lambda(K,M)} |\lambda \Leftrightarrow \sigma| / \|M^{-1}\| u \ .$$

Since

$$\max_{\lambda(K,M)} |\lambda \Leftrightarrow \sigma| < |\lambda_{\max} \Leftrightarrow \lambda_{\min}| + \min_{\lambda(K,M)} |\lambda \Leftrightarrow \sigma| \ ,$$

we have $\eta > \|E\|$ when

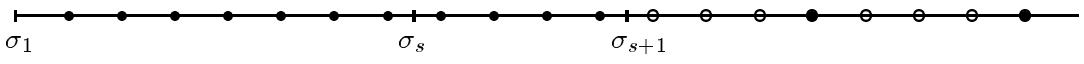
$$\min_{\lambda(K,M)} |\lambda \Leftrightarrow \sigma| (1 \Leftrightarrow u) > |\lambda_{\max} \Leftrightarrow \lambda_{\min}| u \ ,$$

from which the proof follows. □

Our new Lanczos code `EA16` allows the pole σ in the spectral transformation to be either fixed or to vary as the computation proceeds. To illustrate how `EA16` selects appropriate poles, we consider the computation of eigenvalues to the right of a point τ. The idea is to start with an empty trust interval $[\tau, \tau]$ and gradually expand it as Ritz values are locked. The initial pole is taken to be $\sigma = \tau$, which ensures eigenvalues close to τ converge rapidly. When at least one Ritz value has converged, the pole is moved to $\sigma = \sigma_2$, where σ_2 is chosen on the right of $\sigma_1 = \tau$. The Lanczos method continues with the new pole until $\nu_2 \Leftrightarrow \nu_1$ Ritz values lying in $[\sigma_1, \sigma_2]$ have converged, where ν_1 and ν_2 are the number of negative eigenvalues of $K \Leftrightarrow \sigma_1 M$ and $K \Leftrightarrow \sigma_2 M$, respectively. $[\sigma_1, \sigma_2]$ is now a trust interval.

Once a trust interval has been found, a new pole is chosen to the right of the trust interval since we want to expand the interval in that direction. The new pole is chosen to lie between a converged (locked) Ritz value and the left-most Ritz value that has not yet converged, as is shown in Figure 1. The idea is that eigenvalues to the right of the trust interval will converge in the subsequent iterations of the Lanczos method. The pole is chosen to lie between clusters of eigenvalues. If the distance between the pole and the left-most unconverged Ritz value is too small, the pole is chosen between the left-most and the next unconverged Ritz value.

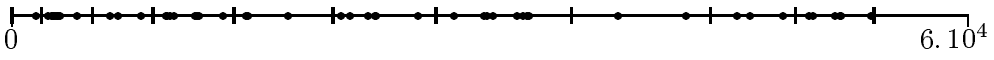
Figure 1: Expansion of the trust interval $[\sigma_1, \sigma_s]$ into $[\sigma_1, \sigma_{s+1}]$. Bullets denote converged (locked) Ritz values and circles unconverged Ritz values. The new pole is chosen between the old trust interval and the left-most unconverged Ritz value.



When the number of eigenvalues $\nu_2 \Leftrightarrow \nu_1$ is larger than the Krylov subspace dimension, convergence of all the eigenvalues in the interval $[\sigma_1, \sigma_2]$ may be slow. In this situation, `EA16` tries to pick a new pole lying between σ_1 and σ_2.

Example 2.1 We used `EA16` to compute 50 eigenvalues lying to the right of 0 for the structural problem `BCSST13` from the Harwell Boeing Collection (Duff, Grimes and Lewis 1992). This is a generalized eigenvalue problem of dimension 2004. Figure 2 shows the computed eigenvalues and the selected poles when 20 Lanczos vectors are used. Each selection of a pole corresponds to an expansion of the trust interval. The first pole is 0. The last pole is selected to verify the first 50 eigenvalues to the right of 0 have been found.

Figure 2: Computed Ritz values (dots) and selected poles (ticks) for problem `BCSST13`.



2.4.3 Rational Lanczos

Changing the pole in the spectral transformation Lanczos method requires building an entirely new Krylov space. This implies that the Lanczos vectors and the tidiagonal matrix built with the old pole are completely lost. It is possible to change the pole without throwing away the subspace by the use of the rational Krylov (Ruhe 1984, Ruhe 1998) or rational Lanczos method (Meerbergen 1999). Suppose we want to change the pole from σ to σ^+. Using $(A \Leftrightarrow \sigma I)^{-1}\mathcal{V}_k = \mathcal{V}_{k+1}\underline{T}_k$, we derive

$$\begin{aligned}(A \Leftrightarrow \sigma I)\mathcal{V}_{k+1}\underline{T}_k &= \mathcal{V}_k \\ (A \Leftrightarrow \sigma^+ I)\mathcal{V}_{k+1}\underline{T}_k &= \mathcal{V}_{k+1}(\underline{I} + (\sigma \Leftrightarrow \sigma^+)\underline{T}_k)\ .\end{aligned}$$

Consider the QR factorization $\left[\begin{array}{cc} \underline{Q} & q \end{array}\right]\left[\begin{array}{c} R \\ 0 \end{array}\right] = \underline{Q}R = \underline{H}_k = \underline{I} + (\sigma \Leftrightarrow \sigma^+)\underline{T}_k$, where $\underline{Q}$ is a $kb+b \times kb$ orthogonal matrix and $Q = \left[\begin{array}{cc} \underline{Q} & q \end{array}\right]$ is a $kb+b \times kb+b$ orthogonal matrix and R is $kb \times kb$ upper triangular. Final manipulations give

$$(A \Leftrightarrow \sigma^+ I)^{-1}\mathcal{V}_{k+1}\underline{Q} = \mathcal{V}_{k+1}QQ^T\underline{T}_k R^{-1}\ . \tag{12}$$

With $\underline{T}_k^+ = Q^T\underline{T}_k R^{-1}$ and $\mathcal{V}_{k+1}^+ = \mathcal{V}_{k+1}Q$, we derive the following algorithm :

Algorithm 2.4 (Rational Lanczos change of pole)

1. *Form* $\underline{H}_k = \underline{I}_k + (\sigma \Leftrightarrow \sigma^+)\underline{T}_k$.
2. *Factorize* $\underline{H}_k = Q\left[\begin{array}{c} R \\ 0 \end{array}\right]$.
3. *Compute* $\underline{T}_k^+ = Q^T\underline{T}_k R^{-1} = \frac{1}{\sigma-\sigma^+}\left(I \Leftrightarrow Q^T\left[\begin{array}{c} R^{-1} \\ 0 \end{array}\right]\right)$.
4. *Apply the orthogonal transformation* P *such that* $\left[\begin{array}{cc} P & 0 \\ 0 & 1 \end{array}\right]^T \underline{T}_k^+ P$ *is tridiagonal.*
5. *Update* $\mathcal{V}_{k+1}^+ = V_{k+1}Q\left[\begin{array}{cc} P & 0 \\ 0 & 1 \end{array}\right]$.

As discussed in §2.4.2, we pick the poles to lie between clusters of eigenvalues. Following the analysis given in Meerbergen (1999), a change of pole changes the error in the Lanczos

recurrence relation. Fortunately, this error is small when the poles are chosen away from the Ritz values. We choose the new pole so that

$$\max_{\lambda} |\lambda \Leftrightarrow \sigma| / |\lambda \Leftrightarrow \sigma^+| \leq \zeta \tag{13}$$

where $\theta = (\lambda \Leftrightarrow \sigma)^{-1}$ is a Ritz value and ζ is a growth factor. In `EA16`, this is controlled by the variable `CNTL(4)` and has default value 50. This condition forces the distance between a Ritz value and the old pole to be at most ζ times the distance between the Ritz value and the new pole. This prevents the new pole from being chosen close to a Ritz value and maintains stability of the rational Lanczos method.

Changing the pole also changes the residual norm of the locked Ritz vectors. Consider the standard eigenvalue problem and let (θ, x) with $\theta = (\lambda \Leftrightarrow \sigma)^{-1}$ be a locked Ritz pair for $S = (A \Leftrightarrow \sigma I)^{-1}$ with residual $r = Sx \Leftrightarrow \theta x$. In addition, let (θ^+, x) with $\theta^+ = (\lambda \Leftrightarrow \sigma^+)^{-1}$ be the corresponding Ritz pair for $S^+ = (A \Leftrightarrow \sigma^+ I)^{-1}$. Then

$$r^+ = S^+ x \Leftrightarrow \theta^+ x = \frac{\lambda \Leftrightarrow \sigma}{\lambda \Leftrightarrow \sigma^+} (A \Leftrightarrow \sigma^+ I)^{-1} (A \Leftrightarrow \sigma I) r \ .$$

In `EA16`, we assume that $\|(A \Leftrightarrow \sigma^+ I)^{-1}(A \Leftrightarrow \sigma I) r\| \simeq \|r\|$ and we estimate $\|r^+\| \simeq |\lambda \Leftrightarrow \sigma| / |\lambda \Leftrightarrow \sigma^+| \|r\|$. This is a reasonable assumption because most eigenvalues of $(A \Leftrightarrow \sigma^+ I)^{-1}(A \Leftrightarrow \sigma I)$ lie close to 1 and r is often poor in the dominant Ritz vectors of $(A \Leftrightarrow \sigma^+ I)^{-1}(A \Leftrightarrow \sigma I)$. In addition, if (13) holds, $\|(A \Leftrightarrow \sigma^+ I)^{-1}(A \Leftrightarrow \sigma I)\| \leq \zeta$. This implies that $\|r^+\| / |\theta^+| \simeq \|r\| / |\theta|$, so the relative residual norm does not change very much. If

$$S \mathcal{V}_k \Leftrightarrow \mathcal{V}_{k+1} \underline{T}_k = E$$

with $\|E\| \ll \underline{T}_k$, then

$$S^+ \mathcal{V}_k \Leftrightarrow \mathcal{V}^+_{k+1} \underline{T}^+_k = E^+ \equiv (A \Leftrightarrow \sigma^+ I)^{-1} (A \Leftrightarrow \sigma I) E R^{-1} \ .$$

We want $\|E\| / \|\underline{T}_k\| \simeq \|E^+\| / \|\underline{T}^+_k\|$. Since $\|(A \Leftrightarrow \sigma^+ I)^{-1}(A \Leftrightarrow \sigma I)\| \leq \zeta$ and $\|R^{-1}\| \leq \zeta$, $\|E^+\|$ is bounded from above by $\|E\| \zeta^2$.

For the generalized eigenvalue problem with the shift-invert transformation $S = (K \Leftrightarrow \sigma M)^{-1} M$, we can again use Algorithm 2.4 to change the pole. For the buckling transformation $S_B = (K \Leftrightarrow \sigma M)^{-1} K$, a change of pole may be carried out using Algorithm 2.4 with the following modification to Step 3.

3. Compute $\underline{T}^+_k = \sigma Q^T \underline{T}_k R^{-1} = \frac{\sigma}{\sigma - \sigma^+} \left(I \Leftrightarrow Q^T \begin{bmatrix} R^{-1} \\ 0 \end{bmatrix} \right)$.

For the standard eigenvalue problem, it can be convenient to start the computation using the regular mode and then to accelerate convergence by switching to the shift-invert mode. Suppose we have $A \mathcal{V}_k = \mathcal{V}_{k+1} \underline{T}_k$ and $\langle \mathcal{V}_{k+1}, \mathcal{V}_{k+1} \rangle = 1$. We can transform this into $(A \Leftrightarrow \sigma^+ I)^{-1} \mathcal{V}^+_k = \mathcal{V}^+_{k+1} \underline{T}^+_k$ with $\langle \mathcal{V}^+_{k+1}, \mathcal{V}^+_{k+1} \rangle = I$ using the following algorithm :

Algorithm 2.5 (Change from regular to shift-invert mode)

1. *Form* $\underline{H}_k = \underline{T}_k \Leftrightarrow \sigma^+ \underline{I}_k$.
2. *Factorize* $\underline{H}_k = Q \begin{bmatrix} R \\ 0 \end{bmatrix}$.
3. *Compute* $\underline{T}_k^+ = Q^T \begin{bmatrix} R^{-1} \\ 0 \end{bmatrix}$.
4. *Apply the orthogonal transformation* P *such that* $\begin{bmatrix} P & 0 \\ 0 & 1 \end{bmatrix}^T \underline{T}_k^+ P$ *is tridiagonal.*
4. *Update* $\mathcal{V}_{k+1}^+ = V_{k+1} Q \begin{bmatrix} P & 0 \\ 0 & 1 \end{bmatrix}$.

Algorithm 2.5 may also be used to change from the regular mode to the shift-invert mode for the generalized eigenvalue problem.

2.5 A note on harmonic Ritz values

The Ritz values computed by the Lanczos algorithm are bound by the smallest and largest eigenvalues $\lambda_{\min}$ and $\lambda_{\max}$. When spectral transformations are used, the Ritz values are similarly bounded by $\theta_{\min}$ and $\theta_{\max}$, where $\theta = (\lambda \Leftrightarrow \sigma)^{-1}$ for the shift-invert mode and $\theta = \lambda/(\lambda \Leftrightarrow \sigma)$ for the buckling mode. The computed θ's may give corresponding λ's that lie outside $[\lambda_{\min}, \lambda_{\max}]$. This can lead to poor pole selection when extremal (smallest or largest) eigenvalues are wanted. To prevent this, harmonic Ritz values may be used. The shift-invert transformation produces $\mathcal{V}_{k+1}$ and $\underline{T}_k$ for which $(A \Leftrightarrow \sigma I)^{-1} \mathcal{V}_k = \mathcal{V}_{k+1} \underline{T}_k$ where the Ritz values are computed from $T_k = \langle \mathcal{V}_k, (A \Leftrightarrow \sigma I)^{-1} \mathcal{V}_k \rangle$. We can transform this into $A\mathcal{V}_k^+ = \mathcal{V}_{k+1}^+ \underline{T}_k^+$ where $\underline{T}_k^+$ and $\mathcal{V}_{k+1}^+$ are computed from the QR decomposition

$$Q \begin{bmatrix} R \\ 0 \end{bmatrix} = \underline{T}_k$$

as $\mathcal{V}_{k+1}^+ = \mathcal{V}_{k+1} Q$ and $\underline{T}_k^+ = Q^T \underline{I}_k R^{-1} + \sigma \underline{I}_k$. Note that $T_k^+ = \langle \mathcal{V}_k^+, A\mathcal{V}_k^+ \rangle$ with $\langle \mathcal{V}_k^+, \mathcal{V}_k^+ \rangle = I$ so the spectrum of T_k^+ is bounded by $\lambda_{\min}$ and $\lambda_{\max}$. Note that $\mathrm{Range}(\mathcal{V}_k^+) = \mathrm{Range}((A \Leftrightarrow \sigma I)^{-1} \mathcal{V}_k)$ so the extremal eigenvalues of A are filtered away from $\mathcal{V}_k$. This can lead to a loss of efficiency when σ is far away from the desired extreme eigenvalue. `EA16` computes harmonic Ritz values when either the shift-invert or buckling mode is used and left-most or right-most eigenvalues are sought.

2.6 Block Gram-Schmidt orthogonalization

Steps 2.2–2.5 of Algorithms 2.1 and 2.3 orthonormalize the new Lanczos vectors V_{j+1} against the columns of $\mathcal{V}_j$. The Lanczos method has the nice property that $\langle V_{j+1}, \mathcal{V}_{j-2} \rangle = 0$ and $B_j^T = \langle AV_j, V_{j+1} \rangle$ is known from the previous iteration. The only Gram-Schmidt coefficients to be computed are thus C_{j+1} and B_{j+1}^T.

Unfortunately, in exact arithmetic, the Lanczos vectors may lose orthogonality. The Lanczos method may then produce spurious eigenvalues. Consequently, either reorthogonalization or some mechanism for detecting spurious eigenvalues is required. An alternative to Algorithm 2.1 is Arnoldi's method, i.e. to force orthogonality against all computed Lanczos vectors by the use of Gram-Schmidt orthogonalization instead of using the properties of the Lanczos recurrence relation. However, in exact arithmetic, the Lanczos vectors may still lose orthogonality. Thus, even if full Gram-Schmidt orthogonalization is used, reorthogonalization is required (Daniel,

Gragg, Kaufman and Stewart 1976, Sorensen 1992). In addition, for large k, the cost of full orthogonalization is much larger than for the Lanczos method.

One possible solution to restore orthogonality is so-called partial reorthogonalization. This is usually cheap, because it is only performed when a significant loss of orthogonality may occur. Reorthogonalization schemes are discussed by Parlett (1980) for the Lanczos method and by Grimes et al. (1994) for the block Lanczos method. In our implementation, we follow the partial reorthogonalization scheme of Grimes et al. (1994).

In this section, we use the notation $\omega_{j,l} = \|\langle V_j, V_l\rangle\|$ and $\epsilon = u\|\underline{T}_k\|$ (u machine precision). We present the analysis for the standard eigenvalue problem $Ax = \lambda x$, but by replacing A by $M^{-1}K$, $(K \Leftrightarrow \sigma M)^{-1}M$ or $(K \Leftrightarrow \sigma M)^{-1}K$, the conclusions may also be applied to the generalized problem.

2.6.1 Partial reorthogonalization

The idea is to explicitly reorthogonalize the Lanczos vectors when they lose orthogonality without checking this property explicitly. The orthogonality of the Lanczos vectors is modelled as follows. In finite precision arithmetic, the recurrence relation (3) becomes

$$AV_j \Leftrightarrow V_{j+1}B_j \Leftrightarrow V_jC_j \Leftrightarrow V_{j-1}B_j^T = F_j \ ,$$

where F_j represents the rounding error. Premultiplying by V_l^T (or $V_l^T M$ for the generalized problem or $V_l^T K$ for the buckling problem),

$$\langle V_l, AV_j\rangle \Leftrightarrow \langle V_l, V_{j+1}\rangle B_j \Leftrightarrow \langle V_l, V_j\rangle C_j \Leftrightarrow \langle V_l, V_{j-1}\rangle B_{j-1}^T = \langle V_l, F_j\rangle \ .$$

Similarly, we have

$$\langle V_j, AV_l\rangle \Leftrightarrow \langle V_j, V_{l+1}\rangle B_l \Leftrightarrow \langle V_j, V_l\rangle C_l \Leftrightarrow \langle V_j, V_{l-1}\rangle B_{l-1}^T = \langle V_j, F_l\rangle \ .$$

Eliminating $\langle V_j, AV_l\rangle = \langle V_l, AV_j\rangle^T$ from these equations we obtain for $l = 1, \ldots, j \Leftrightarrow 2$,

$$\omega_{j+1,l} \le \|B_j^{-1}\|(\|B_l\|\omega_{j,l+1} + \|B_{l-1}\|\omega_{j,l-1} + \|B_{j-1}\|\omega_{j-1,l} + (\|C_j\| + \|C_l\|)\omega_{j,l} + \epsilon), \qquad (14)$$

where we assume that $\|\langle V_l, F_j\rangle + \langle V_j, F_l\rangle\| \le \epsilon = u\|\underline{T}_k\|$. Since V_{j+1} is orthogonalized against V_j using Gram-Schmidt orthogonalization with reorthogonalization (also called iterative Gram-Schmidt) by computing the coefficient C_j, we let $\omega_{j+1,j} = \omega_{\min} := ub\sqrt{n}$.

Given a tolerance ω_{TOL}, in the Lanczos algorithm we use a recurrence relation for modeling $\omega_{j,l}$ as follows.

Algorithm 2.6 (Block Lanczos method with reorthogonalization)

1. *Let $V_1 = [v_1, \ldots, v_b]$ be a given set of initial vectors with $\langle V_1, V_1\rangle = I$.*
   Let $B_0 =$ and $V_0 = 0$.
2. *For $j = 1 : k$ do*
   - 2.1. *Form $W_j = AV_j$.*
   - 2.2. *Form $W_j' = W_j \Leftrightarrow V_{j-1}B_{j-1}^T$.*
   - 2.3. *Compute C_j and W_j'' so that $\langle V_j, W_j''\rangle = 0$ and $C_j = \langle V_j, W_j'\rangle$ using iterative Gram-Schmidt. $\omega_{j+1,j} = \epsilon$.*
   - 2.4. *Form $W_j'' = W_j' \Leftrightarrow V_jC_j$.*
   - 2.5. *Factorize $W_j'' = V_{j+1}B_j$ so that $\langle V_{j+1}, V_{j+1}\rangle = I$.*
   - 2.6. *Update $\omega_{j+1,l}$ for $l = 1, \ldots, j \Leftrightarrow 1$ using the right hand side of (14).*
   - 2.7. *If $\max_l \omega_{j+1,l} > \omega_{\mathrm{TOL}}$ then reorthogonalize V_j and V_{j+1} against $\mathcal{V}_j$ using iterative Gram-Schmidt and set $\omega_{j,l} = \omega_{j+1,l} = \omega_{\min}$, $1 \le l \le j \Leftrightarrow 2$.*

2.6.2 Partial reorthogonalization against locked Ritz vectors

There are two reasons for orthogonalizing Lanczos vectors against locked Ritz vectors. Firstly, the Ritz vectors are not exact eigenvectors, and, secondly, the computations are carried out in finite precision. The propagation of loss of orthogonality is modelled as follows : let (λ, x) be a locked Ritz pair and $\xi_{j+1} = \langle x, V_j \rangle$, then

$$\langle x, AV_j \rangle \Leftrightarrow \langle x, V_{j+1} \rangle B_j \Leftrightarrow \langle x, V_j \rangle C_j \Leftrightarrow \langle x, V_{j-1} \rangle B_{j-1}^T = \langle x, F_j \rangle \ .$$

Using $x^T A = r^T + \lambda x^T$, we find that

$$\xi_{j+1} \leq \|B_j^{-1}\| (\xi_j \|\lambda I \Leftrightarrow C_j\| + \xi_{j-1} \|B_{j-1}\| + \sqrt{\langle r, r \rangle} + \epsilon \ .$$

When ξ_{j+1} is larger than the tolerance ω_{TOL}, V_{j-1} and V_j are orthogonalized for x, and ξ_j and ξ_{j+1} are set to $\omega_{\min}$. In `EA16`, we check the orthogonality against all locked Ritz vectors. This is in contrast to Grimes et al. (1994), who only check the need for reorthogonalization for a selected number of locked Ritz vectors. In `EA16`, orthogonality is checked against all locked Ritz vectors but reorthogonalization is only carried out for the Ritz vectors for which the estimate of $\langle x, AV_j \rangle$ is larger than ω_{TOL}.

2.6.3 Reorthogonalization and implicit restarting

When implicit restarting or purging is performed, the loss of orthogonality may grow for particular Lanczos vectors. In finite precision arithmetic, the Lanczos vectors and the Lanczos matrix satisfy

$$\begin{aligned} A\mathcal{V}_k \Leftrightarrow \mathcal{V}_{k+1} \underline{T}_k &= F_k \\ \langle \mathcal{V}_{k+1}, \mathcal{V}_{k+1} \rangle \Leftrightarrow I &= \Omega_k \ , \end{aligned}$$

where Ω_k is the matrix of ω_{jl}'s for $j, l = 1, \ldots, k+1$. Implicit restarting or purging applies an orthogonal transformation $Q \in \mathbf{R}^{kb+b \times pb+b}$ to map $\mathcal{V}_{k+1} \in \mathbf{R}^{n \times kb+b}$ onto $\mathcal{V}_{p+1}^+ \in \mathbf{R}^{n \times pb+b}$ with $p < k$. Let $\underline{Q}$ be the matrix that contains the first pb columns of Q. With $\mathcal{V}_{p+1} = \mathcal{V}_{k+1} Q$ and $\underline{T}_p^+ = Q^T \underline{T}_k \underline{Q}$, we have

$$\begin{aligned} A\mathcal{V}_p^+ \Leftrightarrow \mathcal{V}_{p+1}^+ \underline{T}_p^+ &= F_k \underline{Q} \\ \langle \mathcal{V}_{p+1}^+, \mathcal{V}_{p+1}^+ \rangle \Leftrightarrow I &= Q^T E_k Q \ . \end{aligned}$$

The Lanczos vectors retain their orthogonality in a global sense, because $\|Q^T E_k Q\| = \|E_k\|$. When $\omega_{\max} \leq \omega_{j,k}$, it is possible that $\|\langle V_j^+, V_k^+ \rangle\| = \omega_{\max}(k+1)$, since V_j^+ is a linear combination of $kb + b$ columns of $\mathcal{V}_{k+1}$. In `EA16`, we estimate that $\|\langle \mathcal{V}_{p+1}^+, \mathcal{V}_{p+1}^+ \rangle\| \sim \omega_{\max}$ so that $\omega_{j,l}^+ = \|\langle V_j^+, V_l^+ \rangle\| \sim \omega_{\max}$.

2.6.4 Reorthogonalization and changing the pole

With the shift-invert transformation we have

$$(A \Leftrightarrow \sigma I)^{-1} V_j \Leftrightarrow V_{j+1} B_j \Leftrightarrow V_j C_j \Leftrightarrow V_{j-1} B_{j-1}^T = F_j \ .$$

The term F_j leads to the term ϵ in (14). After changing the pole σ to σ^+, we have

$$(A \Leftrightarrow \sigma^+ I)^{-1} V_j^+ \Leftrightarrow V_{j+1}^+ B_j^+ \Leftrightarrow V_j^+ C_j^+ \Leftrightarrow V_{j-1}^+ (B_{j-1}^+)^T = F_j^+ \ .$$

Following (12), $\|F_j^+\| \leq \|(A \Leftrightarrow \sigma^+ I)^{-1} (A \Leftrightarrow \sigma I)\| \|E\| \|R^{-1}\|$, which is bounded from above by $\|E\| \zeta^2$ where ζ is the growth factor of the change of pole and $\|E\| \sim \|(A \Leftrightarrow \sigma I)^{-1}\| u$. As in the last section, we can use (14) for estimating the orthogonality of the Lanczos vectors, but we now use $\epsilon = u \|\underline{T}_k\| \zeta^2$.

2.7 Avoiding breakdown for singular mass matrices

The generalized eigenvalue problem $Kx = \lambda Mx$ for the Stokes problem has the form

$$\begin{bmatrix} \tilde{K} & C \\ C^T & 0 \end{bmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \lambda \begin{pmatrix} u \\ v \end{pmatrix} \begin{bmatrix} \tilde{M} & 0 \\ 0 & 0 \end{bmatrix},$$

where $\tilde{K}$ and $\tilde{M}$ are symmetric and $\tilde{M}$ is positive definite. Thus, in this case, M is positive semi-definite. There are also applications in structural analysis where the mass matrix M is not only semi-definite but does not have an explicit zero block. In this case, we can (at least formally), decompose M in the form

$$M = Q \begin{bmatrix} \tilde{M} & 0 \\ 0 & 0 \end{bmatrix} Q^T.$$

Defining $\hat{K} = Q^T K Q$ and $\hat{M} = Q^T M Q$, the spectral transformation has the same block structure as for the Stokes problem, namely

$$S = (\hat{K} \Leftrightarrow \sigma \hat{M})^{-1} \hat{M} = \begin{bmatrix} S_1 & 0 \\ S_2 & 0 \end{bmatrix}.$$

It follows that the spectral transformation has a zero eigenvalue with the nullspace of $\hat{M}$ as the corresponding eigenspace. Ericsson (1986) showed that the zero eigenvalue may be defective with index 2 i.e. there are x for which $Sx \neq 0$ but $S^2 x = 0$. We shall see that the presence of this zero eigenvalue may lead to breakdown or the computation of spurious eigenvalues.

Meerbergen and Spence (1997) showed that the Lanczos method applied to S using the $\hat{M}$ inner product is equivalent to solving $S_1 u = \theta u$ using the $\tilde{M}$ inner product. The nullspace of M thus plays no role in the Lanczos method. As a result, the components of the Lanczos vectors in the nullspace of M may grow in an uncontrolled way. The Lanczos vectors may be dominated by components in the nullspace of M so that $\|x\|_2$ is very large even if $\|x\|_M = 1$. If M does not have an explicit zero block, the Lanczos vectors may still be seriously affected. It is even possible that the inner product $x^T M x$ becomes negative during the M orthogonalization of the Lanczos vectors. This leads to breakdown of the method.

One way to avoid breakdown is to remove the components in the nullspace of M. Since the nullspace of M is the nullspace of S, this can be achieved by filtering, i.e. premultiplying the Lanczos vectors by S. When zero is a defective eigenvalue, $S\mathcal{V}_k$ may still have components in the nullspace of S so a second filter step $S(S\mathcal{V}_k)$ is required in this case. When S has eigenvalues that are small in modulus but nonzero, more than two filter steps may be helpful. Meerbergen (2000) suggests using the following algorithm to avoid breakdown. Note that an implicit restart with a zero shift has precisely the effect of a filter step. Therefore, we call this implicit filtering or an implicit filter step.

Algorithm 2.7 (Avoiding breakdown in the Lanczos method)

1. *Choose σ and $V_1 \in \mathbf{R}^{n \times b}$ and the number of filter steps γ.*
2. *Perform γ filter steps $V_1 = S^\gamma \cdot V_1$.*
3. *Perform k steps of the Lanczos method. If $\|V_j\|_F > \|V_1\|_F / \sqrt{u}$, goto Step 4. If the M orthogonalization of V_j suffers breakdown, let $j = j \Leftrightarrow 1$ and goto Step 4.*
4. *Perform γ implicit restarts with zero shifts.*
5. *Compute Ritz values and Ritz vectors.*
6. *Check convergence.*
7. *Continue the Lanczos method from iteration $j + 1$.*

By performing an implicit restart on the first $j \Leftrightarrow 1$ Lanczos iterations we hope to prevent breakdown on iteration j. When $\|V_j\|_F$ becomes too large, we also perform γ implicit restarts to reduce this norm. Since an implicit restart for $j < 2$ does not make any sense, a breakdown for $j = 2$ is incurable.

Recall from Theorem 2.1 that $\mathcal{V}_k$ can be cheaply multiplied by S using an implicit restart. In Meerbergen and Spence (1997) it was shown that the error in $S\mathcal{V}_k R_k^{-1} \Leftrightarrow \mathcal{V}_k^+$ is relatively small when the condition number of $\underline{T}_k$ is small. This is the case when T_k has no eigenvalues near zero. This is ensured by regularly performing an implicit restart which filters away eigenvalues near zero.

The zero eigenvalue of S corresponds to the infinite eigenvalue of $Kx = \lambda Mx$, which is usually not wanted. When this eigenvalue is defective, S_1 also has a zero eigenvalue and so may T_k. The undesired zero eigenvalue of S may still be computed. When extremal eigenvalues are wanted, i.e. leftmost or rightmost, this may lead to spurious eigenvalues and a poor pole selection. The poles may converge to infinity in the rational Lanczos method. This was illustrated for the rational Krylov method in De Samblanx, Meerbergen and Bultheel (1997). Premultiplying by S^γ and implicit filtering removes the nullspace of S^γ and the presence of spurious eigenvalues (Meerbergen and Spence 1997). Algorithm 2.7 also helps in this situation.

The Ritz vectors $x = \mathcal{V}_k z$ with $T_k z = \theta z$ may have a component in the nullspace of S because the latter is not controlled by the Lanczos method. As already mentioned, the M orthogonal Lanczos method does not see the nullspace of M and x may have an undesired component in the nullspace of M. This component may be filtered by replacing x by Sx/θ. Computation of Sx can be achieved by a simple trick, called purification (Ericsson and Ruhe 1980). Let (θ, x) be a Ritz pair of S. Then

$$\begin{aligned} Sx/\theta &= S\mathcal{V}_k z/\theta = \mathcal{V}_{k+1} T_k z/\theta \\ &= x + V_k B_k E_k^T z/\theta \ . \end{aligned}$$

If the residual norm is $\rho = \|B_k E_k^T z\|_M$, then $\|y\|_M = (1 + (\rho/\theta)^2)^{-1/2}$, which is close to 1 for a small relative error on θ. In `EA16`, we use purification for the locked Ritz vectors in conjunction with pre-filtered starting vectors and implicit filtering. Note that purification is usually not necessary because implicit filtering removes components of the Lanczos vectors in the nullspace of S.

3 Description of `EA16`

In this section, we briefly look at existing software implementing the Lanczos method. We see that there are a number of packages available that have been developed over the last 20 years, but none of these offers all the features discussed in this paper. This leads us to discuss the design of our new code `EA16`. In particular, we look at the user interface to `EA16` and highlight some of the many options available to the user.

3.1 Existing Lanczos codes

A number of Lanczos codes are currently available, the most well-known of these probably being LANCZOS (Cullum and Willoughby 1985*a*), `EA15` (Parlett and Reid 1981), ARPACK (Lehoucq, Sorensen and Yang 1998), BLZPACK (Marques 1995), and the Boeing code (Boeing 1989). Apart from the Boeing code, each of the other codes is available either in the public domain (through Netlib) or for use under licence from the Harwell Subroutine Library. In this

section, we highlight some of the major features of these codes and their differences, which we summarise in Table 1.

As already discussed, the Lanczos method builds an orthonormal basis for a Krylov space by a cheap process based on the three term recurrence relation. All the codes listed above apart from ARPACK use this relation; ARPACK uses the less efficient Arnoldi process (Arnoldi 1951, Saad 1992, Lehoucq et al. 1998).

It is generally advantageous in terms of both efficiency and reliability to use a block Lanczos method, with a blocksize greater than 1. This is because multiplicities may be missed with a blocksize of 1 and the operations that are performed with the matrices A, M, and K can often be more efficiently implemented when working with a block of vectors. The codes LANCZOS and BLZPACK as well as the Boeing code are block codes.

The major memory cost of the Lanczos method is that required to store the basis vectors. For large-scale practical applications, it is not possible to build large Krylov subspaces. To overcome this, after a prescribed number of steps, BLZPACK and the Boeing code stop the Lanczos process and then build a new Krylov subspace starting with a new basis vector (explicit restarting). ARPACK adopts the more efficient approach of implicit restarting (Sorensen 1992). As discussed in §2.3, implicit restarting reduces the Krylov subspace and then builds it out again, thus avoiding throwing away useful information already computed. Implicit restarting also helps improve the numerical stability of the Lanczos process when M is a semi-definite matrix (Meerbergen and Spence 1997, Lehoucq et al. 1998).

BLZPACK and the Boeing code are the only existing codes designed specifically for solving the generalized problem (2) using the spectral transformation Lanczos method. These two codes each have an automatic pole selection strategy and change the pole to achieve faster convergence. When the pole changes, the Lanczos method is restarted by building a new Krylov space.

`EA15` is designed for computing all the eigenvalues in a specified interval, without regard for multiplicities. The code is efficient because, in common with LANCZOS, it does not use reorthogonalization and store only three basis vectors.

Table 1: Features of other codes

code	cheap orthogonalization	implicit restart	blocking	M inner product	rational Krylov	automatic pole selection
LANCZOS	×		×			
`EA15`	×					
ARPACK		×		×		
BLZPACK	×		×	×		×
Boeing	×		×	×		×
`EA16`	×	×	×	×	×	×

We see from Table 1 that there are a number of features that it is desirable for a Lanczos code to possess but each of the existing codes only offer some of these features. Our aim when designing `EA16` was to develop a state of the art block Lanczos code for solving both the standard and generalized eigenvalue problems, incorporating cheap orthogonalization, implicit restarting, and automatic pole selection. In addition, to avoid throwing away the old subspace when the pole is changed, a key design feature of `EA16` is its use of rational Krylov (see §2.4.3).

3.2 Design and options

The EA16 package is written in standard Fortran 77; a Fortran 90 version is planned. For maximum flexibility and portability, reverse communication is used for operations with the matrices A, K, and M, as well as for other tasks that are dependent on the problem being solved and on whether or not a spectral transformation is being used. The EA16 package is designed for computing a relatively small number of eigenvalues and eigenvectors; EA16 **cannot** be used for computing all the eigenpairs of $Ax = \lambda x$ or $Kx = \lambda Mx$.

Subroutines in the EA16 package are named according to the naming convention of HSL 2000. The single-precision version subroutines all have names that commence with EA16 and have one more letter. The corresponding double-precision versions have the same names with an additional letter D. For clarity, in the remainder of §3 we refer only to the single-precision subroutines. There are three subroutines in the EA16 package that are called directly by the user. These are as follows:

1. EA16I sets default values for the control parameters. It should normally be called once prior to any calls to EA16A and EA16B. The control parameters allow the user to select a number of different options, some of which are only intended for the more experienced user and for difficult problems where the standard approach may not be the best. Full details of the control parameters are given in the user documentation (see Appendix).

2. EA16A computes the amount of workspace that will be required by the code. The user is required to specify the order N of the matrices A, K, and M and the number NWANT of sought-after eigenvalues. In addition, the user must choose the block size BLK for the Lanczos method and specify NV, the maximum number of Lanczos vectors. EA16A performs checks on the user's data and returns to the user the minimum lengths LIWORK and LWORK of the integer and real work arrays required by the Lanczos method. EA16A must be called once before any calls to the main subroutine EA16B. We note that the workspace needed by EA16B is independent of whether the user is solving a standard or generalized eigenvalue problem and of which eigenvalues are required.

3. EA16B computes selected eigenpairs. This routine must be called repeatedly using the reverse communication interface. Using the parameters MODE and WHICH, the user must specify which eigenvalue problem is to be solved and which eigenvalues are required. These parameters are discussed in detail below.

Examples illustrating the calling sequence and the use of some of the control parameters are included in §4 and in the user documentation (see Appendix).

3.2.1 Eigenvalue solver modes

EA16 offers five different eigenvalue solver modes. For convenience, we denote by OP the operator that is applied to the vectors in the Lanczos process. With this notation, the modes that may be selected using the parameter MODE are :

1: The standard eigenvalue problem $Ax = \lambda x$ using the regular mode and the standard inner-product. Here $OP = A$.

2: The standard eigenvalue problem $Ax = \lambda x$ using shift-invert mode and the standard inner-product. Here $OP = (A \Leftrightarrow \sigma I)^{-1}$, with the pole σ not equal to an eigenvalue of A.

3: The generalized eigenvalue problem $Kx = \lambda Mx$ with M positive definite using the regular inverse mode and the M inner-product. Here $OP = M^{-1}K$.

4: The generalized eigenvalue problem $Kx = \lambda Mx$ with M positive (semi) definite. using shift-invert mode and the M inner-product. Here $OP = (K \Leftrightarrow \sigma M)^{-1}M$, with the pole σ not equal to an eigenvalue of $Kx = \lambda Mx$.

5: The buckling problem $Kx = \lambda Mx$ with K positive (semi) definite using the buckling mode and the K inner-product. Here $OP = (K \Leftrightarrow \sigma M)^{-1}K$, with the pole σ not equal to zero or to an eigenvalue of $Kx = \lambda Mx$. Note that the buckling mode should only be used for the generalized problem in the case when M is not positive semi-definite.

For each mode, `EA16` requires the multiplication of sets of vectors by the linear operator OP. Additionally, `EA16` requires the multiplication of vectors by the matrix M (`MODE` $= 2$ and 4 only) and by K (`MODE` $= 5$ only). The reverse communication interface prevents the user from having to pass the matrix A or K and M explicitly to `EA16`; instead, each time a multiplication by OP, M, or K is required, control is returned to the user. The parameter `IDO` is used to indicate the action required by the user (see Section 3.2.3). Note that, for the generalized problem and for the shift-invert mode for the standard problem, multiplication of sets of vectors by OP involves solving a linear system of equations. In particular, the user must solve systems of the form $BU = W$ where $B = (A \Leftrightarrow \sigma I)$ (`MODE` $= 2$), or $B = M$ (`MODE` $= 3$), or $B = (K \Leftrightarrow \sigma M)$ (`MODE` $=$ 4 or 5). The reverse communication interface allows the user to choose any appropriate solver.

For many applications, unless the number of Lanczos vectors `NV` is large, the regular modes 1 and 3 converge very slowly. The number of Lanczos vectors is typically limited by the amount of memory available. We suggest that `NV` is chosen to be larger than the maximum of $2 * $ `NWANT` and `NWANT` $+ 10 * $ `BLK`, where `NWANT` and `BLK` are the number of required eigenvalues and `BLK` is the block size.

It is generally satisfactory to use a blocksize of 1 but `EA16` may be more efficient (as well as more reliable when computing multiple eigenvalues) if a larger value is chosen. The Lanczos method with blocksize 1 (Lanczos 1950) can have difficulties finding multiplicities of eigenvalues. Chatelin (1993) even claims that the Lanczos method cannot find multiplicities in exact arithmetic. It has been suggested (see for example Cullum and Willoughby, 1985*a*) that the blocksize should be chosen larger than or equal to the multiplicity of the eigenvalues. This is only true for methods that use an explicit restart strategy. In `EA16`, we use purging and implicit restarting for reducing the dimension of the subspace, which enables us to keep significant spectral information in the Krylov subspace. In addition, the use of locking reduces the multiplicity of the locked eigenvalues. If an eigenpair (λ, x) is locked, the Lanczos method computes a Lanczos basis orthogonal to x. Mathematically, the eigenvalues of $A|_{\{x\}^\perp}$ are computed. If λ has multiplicity two, $A|_{\{x\}^\perp}$ has an eigenvalue λ with multiplicity one. Thus locking allows `EA16` to compute multiplicities even with a blocksize of 1.

Nevertheless, block methods are usually still more reliable, because multiple eigenvalues may converge *at the same iteration*, without the need for locking and additional iterations. Moreover, block methods allow the use of high level BLAS kernels, which are usually implemented very efficiently on cache machines. This reduces the computational time for the Lanczos method. We give examples to illustrate this in §4.

For the shift-invert and buckling modes, when extremal eigenvalues are wanted, the initial value for the pole σ must be set by the user. In all other cases, `EA16` picks a suitable initial pole. By default, σ is fixed. However, an option exists that allows either `EA16` to propose a change of pole or for the user to select a new pole. The user can control the number of restarts between two changes of pole. Varying σ can speed up convergence significantly, particularly if the user was unable to select a good initial pole or if several eigenvalues are required. The disadvantage of changing the pole is that, if a direct solver is being used, the user factorize the

matrix ($A \Leftrightarrow \sigma^+ I$) (MODE = 2) or ($K \Leftrightarrow \sigma^+ M$) (MODE = 4 or 5) for the new pole σ^+. If an iterative solver is used, the user may need to set up a suitable preconditioner.

3.2.2 Specifying the required eigenvalues

The parameter WHICH must be set by the user to specify the eigenvalues of the original problem that are required. If the user wants eigenvalues to the left or right of a point, the user must also set the parameter RANGE(1) to specify the point; if eigenvalues lying within an interval are sought, the user must set RANGE(1) and RANGE(2) to define the interval. Possible values for WHICH are :

1: Eigenvalues furthest from the point RANGE(1). This option is only available for MODE = 1 and 3. To compute the eigenvalues of largest modulus, the user should set RANGE(1) = 0.

-1: Eigenvalues closest to the point RANGE(1). To compute the eigenvalues of smallest modulus, the user should set RANGE(1) = 0.

-2/2: The right-most/left-most eigenvalues.

-3/3: Half the number of wanted eigenvalues are computed from each end of the spectrum. This option is only available for MODE = 1 and 3. If the number of required eigenvalues is odd, WHICH = 3 computes one more eigenvalue from the upper end than from the lower end and WHICH = $\Leftrightarrow$3 computes one more from the lower end than from the upper end.

-4/4: The eigenvalues to the right/left of the point RANGE(1). If the number of required eigenvalues is greater than the number of Ritz values to the left of RANGE(1), the code computes fewer than the number of requested eigenvalues and terminates with a warning.

5: The eigenvalues inside the interval (RANGE(1), RANGE(2)). If the number of required eigenvalues is greater than the number of Ritz values inside the interval, the code computes fewer than the number of requested eigenvalues and terminates with a warning.

10: This option allows the user to select the wanted and unwanted Ritz values. It makes the code very flexible but we advise that it should only be used by experienced users when none of the other WHICH values is appropriate. If this option is chosen, during the computation the user will be asked to give each Ritz value a priority. This is achieved using the reverse communication interface. When asked for priority values, if a Ritz value is not wanted, the user should give it zero priority, while the most important Ritz values should be given the highest priority. The simplest case is the one where the user divides the Ritz values into two classes: the wanted Ritz values (priority 1) and the unwanted Ritz values (priority 0).

It may be difficult for the user to select a suitable value for the initial pole when extremal eigenvalues are requested (WHICH = $\pm$2). In this case the user may optionally choose to start the computation using the regular mode (MODE = 1 or 3) and, at an appropriate place in the computation, to switch to shift-invert mode (MODE = 2 or 4). When switching modes, the user may take advantage of the latest Ritz values to choose an appropriate pole or may allow EA16 to select σ. A change of mode is irreversible.

3.2.3 Continuing the computation

Once the user has selected the eigensolver mode and which eigenvalues are required (setting the pole σ and `RANGE` as necessary), the computation can proceed. The user makes repeated calls to `EA16B`. On the first call, the user must set the reverse communication parameter `IDO` to 0. On exit from `EA16B`, a value of `IDO` equal to 100 indicates the computation has terminated. By checking the error flag `INFO(1)`, the user can determine whether or not the requested eigenvalues have converged. If `IDO` $\neq$ 100, convergence has not yet been achieved and to continue the computation the user must take the action appropriate to the value of `IDO` and then recall `EA16B`. The action required for each value of `IDO` is explained in detail in the user documentation (see Appendix), but in outline the action is as follows :

1: The user must multiply a set of vectors by OP. The number of vectors is equal to `BLK`, the block size chosen by the user for the Lanczos method.

2: The user must multiply a set of vectors by M (`MODE` = 3 or 4) or K (`MODE` = 5). The number of vectors will be at most `BLK`.

3: The user may select a new pole σ for the Lanczos process.

4: The user must set the parameter `NEINEG` to the number of negative eigenvalues of the matrix B where $B = A \Leftrightarrow \sigma I$ (`MODE` = 2) or $B = K \Leftrightarrow \sigma M$ (`MODE` = 4 or 5). In addition, if the user is using a direct linear solver for performing the multiplication of vectors by OP, the user must factorize B. If an iterative solver is being used, the user may set up a suitable preconditioner. `IDO` = 4 is returned for the initial matrix factorization and when σ has been changed. The user can communicate a failure of the matrix factorization for the pole by setting `IDO` = -4 and recalling `EA16B`, with no other changes to the parameters.

5: The user may change `MODE` from 1 to 2 or from 3 to 4, and optionally choose the pole σ for the shift-invert mode. Other changes to `MODE` raise an error and cause the computation to terminate.

6: The user must select the wanted and unwanted Ritz values by giving each Ritz value a priority (`WHICH` =10 only).

7: The user must supply shifts for implicit restarting.

We remark that if `WHICH` is not equal to 10 and the default settings are used for all the control parameters, the only values of `IDO` $\neq$ 100 that can be returned to the user are 1, 2 and 4. In particular, for the standard eigenvalue problem with the regular mode, only `IDO` = 1 is returned and the user needs only to supply a routine to multiply sets of vectors by A. Similarly, for the generalized eigenvalue problem with the regular mode, only `IDO` = 1 and 2 can be returned. With the default settings, the mode and pole σ are fixed and `EA16` selects the shifts for implicit restarting (see §2.3).

When a spectral transformation is used, the user needs to set the parameter `NEINEG`, which may be computed from the (sparse) matrix factorization. As we explained in §2.4.2, this information allows the code to decide on the appropriate expansion of the trust interval. It also helps the code in selecting a robust value for the pole σ. The code returns the final trust interval for the computed Ritz values. If the user is unable to provide `NEINEG`, `EA16` will not compute a trust interval and, unless supplied by the user, the code picks new poles based solely on the computed Ritz values.

3.2.4 The stopping criteria

In EA16, a computed Ritz pair (x, λ) is accepted as an approximate eigenpair if

$$\|OP * x \Leftrightarrow \theta x\| \leq u * \|\underline{T}_k\| + |\texttt{CNTL(2)}| + |\texttt{CNTL(3)}| * |\theta|, \tag{15}$$

where $\underline{T}_k$ is the Lanczos matrix, u is the machine precision and $\theta = \lambda$ (MODE = 1 or 3), $\theta = (\lambda \Leftrightarrow \sigma)^{-1}$ (MODE = 2 or 4), and $\theta = (\lambda \Leftrightarrow \sigma)^{-1}\lambda$ (MODE = 5). Here CNTL(2) and CNTL(3) are control parameters that may be set by the user. Their default values are zero and $\sqrt{u}$, respectively. The first term in the right-hand side of (15) is the minimum residual norm that makes sense in finite precision arithmetic. The second and third terms are a combination of an absolute and relative backward error tolerance; the user can decide on the importance of and balance between each of these terms.

4 Applications of symmetric eigenvalue problems and numerical examples

The Lanczos method can be used for computing eigenvalues and eigenvectors for any type of symmetric (or Hermitian) eigenvalue problem. In this section, we give a number of applications and numerical examples for which the method is appropriate. The computations were performed by EA16 on a DEC Compaq AlphaServer DS20 using vendor-supplied BLAS.

4.1 Computation of singular values

The computation of the truncated singular value decomposition (SVD) (Golub and Van Loan 1996)) arises in many applications. The SVD of an $l \times n$ matrix A is defined as follows:

$$A = U\Sigma V^T,$$

where $U \in \mathbf{R}^{l\times l}$ and $V \in \mathbf{R}^{n\times n}$ are unitary matrices and $\Sigma \in \mathbf{R}^{l\times n}$ is a diagonal matrix. The diagonal entries of Σ are called the *singular values* and are positive. The columns of U and V are called *singular vectors*. One application is the computation of the best rank-p approximation of A with $p < \min(n, l)$. The matrix A is approximated by $U_p\Sigma_p V_p^T$, where $U_p \in \mathbf{R}^{l\times p}$ and $V_p \in \mathbf{R}^{n\times p}$ are both unitary, and Σ_p is a $p \times p$ diagonal matrix with the largest singular values on its diagonal. The error in the approximation is

$$\|A \Leftrightarrow U_p\Sigma_p V_p^T\| = \sigma_{p+1},$$

where σ_{p+1} is the $(p+1)$st largest singular value. There are two commonly used ways of computing the singular value decomposition of large matrices.

1. It is easy to see that $A^TAV = V\Sigma^2$, so the columns of V_p and the singular values can be computed by computing the eigenpairs of the symmetric matrix A^TA. Once V_p has been computed, provided the p largest singular values are nonzero, U_p can be formed from $U_p = AV_p\Sigma_p^{-1}$. Similarly, we have $AA^TU = U\Sigma^2$. Thus the singular value problem can be solved by solving a symmetric eigenvalue problem of dimension $\min(n, l)$. Since the singular values are squared, this approach is not attractive for computing small singular values, but it improves the separation of large singular values and thus their convergence in the Lanczos method.

2. The singular values and singular vectors can be computed from the eigendecomposition

$$\begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} \begin{pmatrix} V & \Leftrightarrow V \\ U & U \end{pmatrix} = \begin{pmatrix} V & \Leftrightarrow V \\ U & U \end{pmatrix} \begin{bmatrix} \Sigma & 0 \\ 0 & \Leftrightarrow \Sigma \end{bmatrix} .$$

   Since the singular values are not squared, this approach can be better for finding the smaller singular values and corresponding singular vectors. The disadvantage is that the dimension of the eigenvalue problem is $l + n$.

As an example, suppose we have taken output samples from a physical system that satisfies the relation

$$y_s(t) = \sum_{j=1}^{k} \zeta_{js} \sin(jt)$$

in the time-interval $t \in [0, 2\pi]$ for some measurement points $s = 1, \ldots, n$. As is often the case with measurements, the errors can be significant. The goal is to use the SVD for noise reduction. Let $B = [\sin(t) \cdots \sin(kt_i)]$, with $t_i = (i \Leftrightarrow 1)2\pi/(l \Leftrightarrow 1)$ $(i = 1, \ldots, l)$, be a discretization of the basis functions $\sin(jt)$, $j = 1, \ldots, k$. In this test, we use $l = 10,000$ and $k = 4$. We generate a matrix A with $n = 100$ measurements as follows : $A = BZ + E$, where Z is the $k \times n$ matrix whose entries are the coefficients ζ_{js} chosen randomly between $\Leftrightarrow 1$ and 1, and E has randomly generated entries between $\Leftrightarrow 0.1$ and 0.1 The term BZ represents the exact behaviour of the physical system and E is the measurement error. We compute the approximate SVD $A \simeq U_p \Sigma_p V_p^T$ with $p = 4$ and use this to approximate the measurements. An approximation of rank $p = 4$ should be sufficient because the exact measurements BZ have rank 4.

We now illustrate, using the following pseudocode, how `EA16` can be used to compute the p dominant singular values and corresponding singular vectors of A. We assume A is stored in the array `A(1:L,1:N)` and use the first approach discussed above for computing the singular values, i.e. the dominant eigenpairs of the $n \times n$ dense matrix $A^T A$ are computed.

```
C ........  Define variables and arrays
C         L         : row dimension of the problem
C         N         : column dimension of the problem
C         BLK       : blocksize
C         NWANT     : number of wanted eigenvalues
C         NV        : number of Lanczos vectors
C         MODE      : standard eigenvalue problem
C         WHICH     : eigenvalues furthest from RANGE(1)
C         RANGE(1)  : compute dominant eigenvalues

      L         = 10000
      N         = 100
      BLK       = 1
      NWANT     = 4
      NV        = 25
      MODE      = 1
      WHICH     = 1
      RANGE(1)  = 0.D0

C     Set the default values of the control parameters for EA16.
      CALL EA16ID(ICNTL,CNTL)

C     Compute the storage required.
      CALL EA16AD(N,BLK,NWANT,NV,LIWORK,LWORK,ICNTL,INFO)
```

```
C     Initialise reverse communication parameter IDO
      IDO = 0

 1    CONTINUE
         CALL EA16BD(N, BLK, NWANT, NV, MODE, WHICH, IDO, IPOS,
     &               V, LDV, BV, LDBV, RANGE, SIGMA, NEINEG,
     &               IWORK, LIWORK, WORK, LWORK, ICNTL, CNTL, INFO)

C        Reverse communication action
         IF (IDO.EQ.100) THEN
C           Finished
            GO TO 2
         ELSE IF (IDO.EQ.1) THEN

C           Compute V(:,IPOS(3):IPOS(4)) =A^t * A * V(:,IPOS(1):IPOS(2))

            NVECS = IPOS(4)-IPOS(3)+1
            CALL DGEMM('N', 'N', L, NVECS, N, ONE, A, L,
     &                 V(1,IPOS(1)),LDV, ZERO, AV,L)
            CALL DGEMM('T', 'N', N, NVECS, L, ONE, A, L,
     &                 AV,L, ZERO, V(1,IPOS(3)),LDV)

            GO TO 1
         END IF
 2    CONTINUE
```

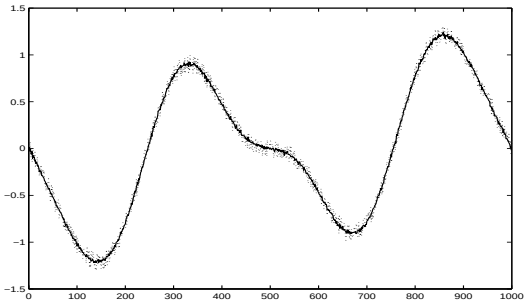
Table 2 presents timings for various blocksizes BLK and different numbers of Lanczos vectors NV. There is a wide variation in performance, with larger block sizes generally giving smaller computation times. This is because the BLAS 3 routine GEMM is very efficient in this case. For this example, the timings also indicate smaller computation times for shorter Lanczos recurrences. Convergence is rapid because the wanted eigenvalues of A^TA are $2.4\,10^4$, $1.8\,10^4$, $1.4\,10^4$, and $9.6\,10^3$ and these are very well separated from the remaining eigenvalues, which are clustered around 5.0. On the same machine, we used the LAPACK routine DGESVD, which is a QR eigensolver for dense matrices. This routine required 0.42 seconds to compute all the singular values and vectors, which is about five to ten times slower than EA16 for the selected singular values. This demonstrates that, although primarily intended for sparse problems, EA16 can be used effectively for computing a small number of eigenpairs of dense matrices.

Table 2: Timings in milliseconds ($10^{-3}s$) for the computation of the four dominant singular values and singular vectors.

	NV =30	NV = 25	NV = 20
BLK =1	76	64	50
BLK =2	42	34	27
BLK =3	34	27	19
BLK =4	27	22	18
BLK =5	24	21	26

The measurements can be reconstructed as $A^{(r)} = U_p\Sigma_pV_p^T$. Figure 3 shows the difference between the first column of A and the first column of $A^{(r)}$. The reconstructed measurement is much smoother.

Figure 3: Comparison of the measurement and its reconstruction after SVD. The dots show the first column of A and the solid line is the first column of $A^{(r)}$.



4.2 Structural and acoustic analysis

The computation of a number of eigenvalues to the right of a point α or in an interval $[\alpha, \beta]$ frequently occurs in the modal analysis of structures (Grimes et al. 1986) and acoustic cavities (Pierce 1981). The eigenvalue problem has the form $Kx = \lambda Mx$ with K positive (semi) definite and M positive definite. We used `EA16` in shift-invert mode with variable pole to compute 50 eigenvalues lying to the right of 0 for the problem `BCSST35` from the Harwell Boeing Collection. This problem is of dimension 30237. The linear systems were solved with the sparse direct solver `MA57` (HSL 2000). The following pseudocode illustrates how the various parameters should be chosen for `EA16` and how the reverse communication interface proceeds for the shift-invert mode.

```
      NV       = 100
      MODE     = 4
      NWANT    = 50
      BLK      = 1

C     Set the default values of the control parameters for EA16.
      CALL EA16ID(ICNTL,CNTL)
C     We use 20 Lanczos steps for each restart.
      ICNTL(4) = 20
C     We want to be able to change the pole at each restart.
      ICNTL(5) = 1

C     Compute the storage required.
      CALL EA16AD(N,BLK,NWANT,NV,LIWORK,LWORK,ICNTL,INFO)

C     Compute NWANT eigenvalues to the right of 0.
      WHICH    = -4
      RANGE(1) = 0.D0

C     Initialise the reverse communication parameter IDO
      IDO = 0

 1    CONTINUE
         CALL EA16BD(N, BLK, NWANT, NV, MODE, WHICH, IDO, IPOS,
     &               V, LDV, BV, LDBV, RANGE, SIGMA, NEINEG,
     &               IWORK, LIWORK, LWORK, WORK, ICNTL, CNTL, INFO)

C        Reverse communication action
```

```
         IF (IDO.EQ.100) THEN
C           Finished
            GO TO 2
         ELSE IF (IDO.EQ.1) THEN

            ... Compute V(:,IPOS(3):IPOS(4)) = (K - SIGMA*M)^-1 * BV
C
         ELSE IF (IDO.EQ.2) THEN

             ... Compute BV(:,IPOS(3):IPOS(4)) = M * V(:,IPOS(1):IPOS(2))
C
         ELSE IF (IDO.EQ.4) THEN

            ... Assemble the sparse matrix to be factorized
            ... Factorize the matrix

C           Flag a failure of the factorization.
            IF (...failure...) IDO = -4

         END IF
         GO TO 1

 2    CONTINUE
```

We ran EA16 using different blocksizes BLK. The results are reported in Table 3. We see that the number of linear solves is minimal for blocksize 1 but that larger blocksizes are more efficient. This is because MA57 exploits BLAS 3 kernels.

Table 3: EA16 timings using different blocksizes.

BLK	1	2	3	4	5
number of factorizations	10	6	4	4	6
number of linear solves	100	120	120	124	135
total time (s)	37	28	22	21	26

We also ran EA16 using a fixed pole $\sigma = 0$ and default settings for all the control parameters. Table 4 presents timings for NV = 70, NWANT = 50 and different values of BLK. Again, the number of linear solves is minimal for blocksize 1 but computational times are reduced by using a large blocksize.

Table 4: EA16 timings for default control parameters.

BLK	1	2	3	4	5
linear solves	101	114	132	156	230
time (s)	28	20	19	20	27

4.3 Computing the rightmost eigenvalues of a real symmetric matrix

As already discussed, when the rightmost (or leftmost) eigenvalues are required, the user may not know how to choose an appropriate pole σ. The user documentation given in the Appendix.

includes an example of the use of EA16 for computing the right-most eigenvalues of a matrix A by starting the computation using the regular mode and then switching to shift-invert mode, allowing the code to select a suitable pole σ.

4.4 The solution of problems with a singular mass matrix

In §2.7, we discussed how EA16 attempts to avoid breakdown in the case of a singular mass matrix. We now illustrate how EA16 performs in practice on such problems by considering the 200×200 matrices $K = L^T \tilde{K} L$ and $M = L^T \tilde{M} L$ where

$$\tilde{K} = \begin{bmatrix} \mathrm{diag}(1, \ldots, 150) & I_{150,50} \\ I_{50,150} & 0_{50} \end{bmatrix} \qquad \text{and} \qquad \tilde{M} = \begin{bmatrix} I_{150} & 0 \\ 0 & E \end{bmatrix} .$$

E a 50×50 diagonal matrix with uniformly distributed entries between $\Leftrightarrow 10^{-10}$ and 10^{10}, and L a lower triangular matrix with ones on the main diagonal and with the off-diagonal entries uniformly distributed between $\Leftrightarrow 0.1$ and 0.1. The condition number of M is of the order of $2.5\,10^{18}$. Note that if E was the zero matrix, the index of the zero eigenvalue of the spectral transformation S would be $\gamma = 2$. The leftmost eigenvalue is required. We ran EA16 using the shift-invert mode with a zero pole, a blocksize of 1, and 31 Lanczos vectors. The control parameter ICNTL(15) was set to γ (see Algorithm 2.7). If $\gamma > 0$, the code automatically performs an implicit restart with zero shift when the inner product becomes negative or the ratio of $\|V_j\|_F / \|V_1\|_F$ is larger than $u^{-1/2}$, as suggested in Algorithm 2.7.

Here we show the iteration number at which breakdown occurs for different values of γ.

γ	0	1	2	3	4	5	6
j	19	18	29	28	28	36	40

The iteration number j at which breakdown occurs increases as γ increases, since the initial vector has smaller components in the eigenvectors corresponding to the small eigenvalues of S. For all values of $\gamma > 0$ listed in the table above, EA16 successfully completed the computations and returned the wanted eigenvalue with the desired accuracy.

4.5 Buckling problems

For buckling problems, K is positive semi definite and M is indefinite and we use the buckling transformation $S_B = (K \Leftrightarrow \sigma M)^{-1} K$. Since K is singular, S_B has a zero eigenvalue that we do not want to compute. Moreover, because K is singular, breakdown of the Lanczos method is possible (see §2.7). To avoid this, when using EA16 we set the control parameter ICNTL(15) to $\gamma = 1$ (see Algorithm 2.7 with S replaced by S_B). The following pseudocode illustrates the use of EA16 for computing the first 50 eigenvalues lying to the right of a point ξ.

```
      NV    = 100
      MODE  = 5
      NWANT = 50
      BLK   = 1

C     Set the default values of the control parameters for EA16.
      CALL EA16ID(ICNTL,CNTL)
C     We use 50 Lanczos steps for each restart.
      ICNTL(4) = 50
C     We allow a change of pole at each restart.
      ICNTL(5) = 1
```

```
C     We set GAMMA=1
      ICNTL(15) = 1

C     Compute the storage required.
      CALL EA16AD(N,BLK,NWANT,NV,LIWORK,LWORK,ICNTL,INFO)

C     Compute NWANT eigenvalues to the right of XI.
      WHICH    = -4
      RANGE(1) =  XI

C     Initialise reverse communication parameter IDO
      IDO = 0

 1    CONTINUE
         CALL EA16BD(N, BLK, NWANT, NV, MODE, WHICH, IDO, IPOS,
     &               V, LDV, BV, LDBV, RANGE, SIGMA, NEINEG,
     &               IWORK, LIWORK, LWORK, WORK, ICNTL, CNTL,
     &               INFO)

C        Reverse communication action
         IF (IDO.EQ.100) THEN
C           Finished
            GO TO 2
         ELSE IF (IDO.EQ.1) THEN

C            ... Compute V(:,IPOS(3):IPOS(4)) = (K - SIGMA*M)^-1 * BV

         ELSE IF (IDO.EQ.2) THEN

C            ... Compute BV(:,IPOS(3):IPOS(4)) = M * V(:,IPOS(1):IPOS(2))

         ELSE IF (IDO.EQ.4) THEN

C            ... Assemble the sparse matrix to be factorized
C            ... Factorize the matrix

C           Flag a failure of the factorization.
            IF (...failure...) IDO = -4

         END IF

         GO TO 1
 2    CONTINUE
```

We report results for problem BCSST27 from the Harwell Boeing Collection (Duff et al. 1992). This problem is of dimension 1224. We took ξ to be the problem scale (Grimes et al. 1994)

$$\xi = \frac{1}{l \sum |M_{ii}|/|K_{ii}|} ,$$

where K_{ii} and M_{ii} are the diagonal entries of K and M, respectively, and l is the number of entries included in the summation, which is taken over all i with $K_{ii} \neq 0$ and $|M_{ii}|/|K_{ii}| < 10^4$. The computed eigenvalues and selected poles are shown in Figure 4. The order in which the poles are chosen is given by the numbers 1 to 6. Note that the second and third poles are chosen too far away from the wanted eigenvalues. The 50 eigenvalues to the right of ξ lying in the trust

interval $[\xi, 11.5]$ were computed. 5 factorizations and a total of linear solves 270 were needed. The total computation time required was $3.2s$. If we allow a factorization every two restarts, the total number of factorizations reduces to 4 with 251 linear solves and a total computation time of $2.9s$.

Figure 4: Computed Ritz values (dots) and selected poles (ticks) for problem `BCSST27`.

5 Concluding remarks

We have discussed the design and development of a new state of the art code `EA16` for the computation of selected eigenpairs of large-scale real symmetric eigenvalue problems. The code is very general, offering the user a large number of options for use in computing eigenpairs of either the standard or the generalized eigenvalue problem. In particular, `EA16` is able to solve generalized eigenvalue problems with a singular mass matrix. The code is based on the block Lanczos method and uses partial reorthogonalization plus implicit restarting, combined with purging and locking of converged Ritz pairs. A spectral transformation may be used to accelerate convergence. A change of pole is allowed. The user may decide on a suitable change of pole or may choose to allow the code to select appropriate poles as the computation proceeds.

`EA16` uses a reverse communication interface. Because of the large number of options available, this means that the code can appear cumbersome to use and the user documentation (see Appendix) is necessarily quite lengthy. To make the code more accessible, we plan to develop a number of specialised routines that call `EA16` but which will themselves have much simpler interfaces, with a minimum number of parameters that must be set by the user. For example, we will develop a routine for the standard eigenvalue problem using the regular mode and another for the shift-invert mode.

We also plan to perform further numerical experiments on a wide range of practical problems and to compare the performance of `EA16` with other publically available Lanczos codes.

References

Arnoldi, W. (1951), 'The principle of minimized iterations in the solution of the matrix eigenvalue problem', *Quart. Appl. Math.* **9**, 17–29.

Baglama, J., Calvetti, D. and Reichel, L. (1998*a*), 'Fast Leja points', *Electronic Trans. Numer. Anal.* **7**, 124–140.

Baglama, J., Calvetti, D., Reichel, L. and Ruttan, A. (1998*b*), 'Computation of a few small eigenvalues of a large matrix with application to liquid crystal modeling', *J. Comput. Phys.* **146**, 203–226.

Boeing (1989), 'The Boeing Extended Mathematical Subprogram Library'. Boeing Computer Services, P.O. Box 24346, Seattle, WA 98124-0346.

Calvetti, D., Reichel, L. and Sorensen, D. (1994), 'An implicitly restarted Lanczos method for large symmetric eigenvalue problems', *Electronic Trans. Numer. Anal.* **2**, 1–21.

Chatelin, F. (1993), *Eigenvalues of matrices*, John Wiley and Sons.

Crouzeix, M., Philippe, B. and Sadkane, M. (1994), 'The Davidson method', *SIAM J. Sci. Comput.* **15**, 62–76.

Cullum, J. and Donath, W. (1974), A block Lanczos algorithms for computing the q algebraically largest eigenvalues and a corresponding eigenspace of large sparse real symmetric matrices, *in* 'Proc. 1974 IEEE Conference on Decision and Control', IEEE Computer Society, pp. 409–426.

Cullum, J. and Willoughby, R. (1985*a*), *Lanczos Algorithms for large symmetric eigenvalue computations. Vol. 1 Theory*, Birkhäuser, Boston.

Cullum, J. and Willoughby, R. (1985*b*), *Lanczos Algorithms for large symmetric eigenvalue computations. Vol. 2 Users Guide*, Birkhäuser, Boston.

Daniel, J., Gragg, W., Kaufman, L. and Stewart, G. (1976), 'Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization', *Math. Comp.* **30**, 772–795.

De Samblanx, G., Meerbergen, K. and Bultheel, A. (1997), 'The implicit application of a rational filter in the RKS method', *BIT* **37**, 925–947.

Dongarra, J., DuCroz, J., Duff, I. and Hammarling, S. (1990), 'A set of Level 3 Basic Linear Algebra Subprograms', *ACM Trans. Math. Softw.* **16**(1), 1–17.

Duff, I. S., Grimes, R. G. and Lewis, J. G. (1992), Users' guide for the Harwell-Boeing sparse matrix collection, Technical Report TR/PA/92/86, CERFACS, Toulouse, France.

Ericsson, T. (1986), A generalised eigenvalue problem and the Lanczos algorithm, *in* J. Cullum and R. Willoughby, eds, 'Large Scale Eigenvalue Problems', Elsevier Science Publishers BV, pp. 95–119.

Ericsson, T. and Ruhe, A. (1980), 'The spectral transformation Lanczos method for the numerical solution of large sparse generalized symmetric eigenvalue problems', *Math. Comp.* **35**, 1251–1268.

Golub, G. and Underwood, R. (1977), The block Lanczos method for computing eigenvalues, *in* J. Rice, ed., 'Mathematical Software III', Academic Press, New York.

Golub, G. and Van Loan, C. (1996), *Matrix computations*, 3rd edn, The Johns Hopkins University Press.

Grimes, R., Lewis, J. and Simon, H. (1986), Eigenvalue problems and algorithms in structural engineering, *in* J. Cullum and R. Willoughby, eds, 'Large Scale Eigenvalue Problems', Elsevier Science Publishers B.V., pp. 81–93.

Grimes, R., Lewis, J. and Simon, H. (1994), 'A shifted block Lanczos algorithm for solving sparse symmetric generalized eigenproblems', *SIAM J. Matrix Anal. Applic.* **15**, 228–272.

HSL (2000), 'A collection of Fortran codes for large scale scientific computation'. Available at http://www.numerical.rl.ac.uk/hsl.

Jia, Z. (1998), 'Polynomial characterizations of the approximate eigenvectors by the refined Arnoldi method and the implicitly restarted refined arnoldi algorithm', *Linear Alg. Appl.* **287**, 191–214.

Lanczos, C. (1950), 'An iteration method for the solution of the eigenvalue problem of linear differential and integral operators', *J. Res. Nat. Bur. Stand.* **45**, 255–282.

Lehoucq, R. B., Sorensen, D. C. and Yang, C. (1998), *ARPACK USERS GUIDE: Solution of Large Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, SIAM, Philadelphia, PA.

Malkus, D. (1981), 'Eigenproblems associated with the discrete LBB condition for incompressible finite elements', *Inter. J. for Engng. Sci.* **19**, 1299–1310.

Marques, O. (1995), BLZPACK : Description and User's Guide, Technical Report TR/PA/95/30, CERFACS, Toulouse, France.

Meerbergen, K. (1999), The rational Lanczos method for Hermitian eigenvalue problems, Technical Report RAL-TR-1999-025, Rutherford Appleton Laboratory. Available through http://www.numerical.rl.ac.uk/reports/reports.html.

Meerbergen, K. (2000), The Lanczos method with semi-inner product, Technical Report RAL-TR-2000-010, Rutherford Appleton Laboratory. Available through http://www.numerical.rl.ac.uk/reports/reports.html.

Meerbergen, K. and Roose, D. (1997), 'The restarted Arnoldi method applied to iterative linear system solvers for the computation of rightmost eigenvalues', *SIAM J. Matrix Anal. Applic.* **18**, 1–20.

Meerbergen, K. and Spence, A. (1997), 'Implicitly restarted Arnoldi and purification for the shift-invert transformation', *Math. Comp.* **66**, 667–689.

Morgan, R. (1996), 'On restarting the Arnoldi method for large nonsymmetric eigenvalue problems', *Math. Comp.* **65**, 1213–1230.

Morgan, R. and Meerbergen, K. (2000), §11.2. Inexact methods, *in* Z. Bai, J. Demmel, J. Dongarra, A. Ruhe and H. van der Vorst, eds, 'Templates for the solution of algebraic eigenvalue problems: a practical guide', SIAM, Philadelphia, USA. In press.

Morgan, R. and Scott, D. (1993), 'Preconditioning the Lanczos algorithm for sparse symmetric eigenvalue problems', *SIAM J. Sci. Comput.* **14**, 585–593.

Nour-Omid, B., Parlett, B., Ericsson, T. and Jensen, P. (1987), 'How to implement the spectral transformation', *Math. Comp.* **48**, 663–673.

Parlett, B. (1980), *The Symmetric Eigenvalue Problem*, Prentice Hall Series in Computational Mathematics, Prentice-Hall.

Parlett, B. and Reid, J. (1981), 'Tracking the progress of the Lanczos algorithm for large symmetric eigenproblems', *IMA J. Numer. Anal.* **1**, 135–155.

Pierce, A. (1981), *Acoustics - An Introduction to its Physical Principles and Applications*, MacGraw Hill, New York.

Ruhe, A. (1984), 'Rational Krylov sequence methods for eigenvalue computation', *Linear Alg. Appl.* **58**, 391–405.

Ruhe, A. (1998), 'Rational Krylov: a practical algorithm for large sparse nonsymmetric matrix pencils', *SIAM J. Sci. Comput.* **19**(5), 1535–1551.

Saad, Y. (1992), *Numerical Methods for Large Eigenvalue Problems*, Algorithms and Architectures for Advanced Scientific Computing, Manchester University Press, Manchester, UK.

Scott, D. (1979), Block Lanczos software for symmetric eigenvalue problems, Technical Report ORNL/CSD 48, Oak Ridge National Laboratory.

Scott, D. (1981), 'Solving sparse symmetric generalised eigenvalue problems without factorisation', *SIAM J. Numer. Anal.* **18**, 102–110.

Sleijpen, G. and van der Vorst, H. (1996), 'A Jacobi-Davidson iteration method for linear eigenvalue problems', *SIAM J. Matrix Anal. Applic.* **17**, 401–425.

Sorensen, D. (1992), 'Implicit application of polynomial filters in a k-step Arnoldi method', *SIAM J. Matrix Anal. Applic.* **13**, 357–385.

Appendix: Specification document for `EA16`

In this Appendix, we include a copy of the specification document for `EA16`. The code will be included in the next release of the Harwell Subroutine Library (HSL 2000) and is available for use now under licence. Anyone interested in using the code may contact the second author (j.scott@rl.ac.uk) author for details of terms and conditions (or see http://www.numerical.rl.ac.uk/hsl).

Note: the specification document is not available as part of this World Wide Web version of the report. Please contact j.scott@rl.ac.uk if you could like a copy of the Appendix.