1 SUMMARY

This routine uses an **implicitly restarted block Lanczos method** or **rational Lanczos method** to compute selected eigenpairs of $\mathbf{Ax}=\lambda\mathbf{x}$ where $\mathbf{A}$ is a large real symmetric matrix or $\mathbf{Ax}=\lambda\mathbf{Mx}$, with $\mathbf{A}$ and $\mathbf{M}$ large real symmetric matrices and either $\mathbf{A}$ or $\mathbf{M}$ positive (semi) definite.

The computed approximate eigenvalues are called Ritz values and the corresponding approximate eigenvectors are Ritz vectors. If we denote by OP the operator that is applied to the vectors in the Lanczos process, EA16 may be used to compute Ritz pairs for one of the following problems:

1. Standard eigenvalue problem : $\mathbf{Ax}=\lambda\mathbf{x}$, $\mathbf{A}$ symmetric.

   This is solved using one of the following modes:

   (a) Regular mode. Here $\mathtt{OP}=\mathbf{A}$.

   (b) Shift-invert mode. Here $\mathtt{OP}=(\mathbf{A}-\sigma\mathbf{I})^{-1}$ with σ not an eigenvalue of $\mathbf{A}$.

   The computed Ritz vectors are orthogonal with respect to the standard innerproduct $\langle\mathbf{x},\mathbf{y}\rangle=\mathbf{x}^T\mathbf{y}$.

2. Generalised eigenvalue problem : $\mathbf{Ax}=\lambda\mathbf{Mx}$, $\mathbf{A}$ symmetric, $\mathbf{M}$ symmetric positive (semi) definite.

   This is solved using one of the following modes:

   (a) Regular inverse mode. Here $\mathtt{OP}=\mathbf{M}^{-1}\mathbf{A}$ with $\mathbf{M}$ nonsingular.

   (b) Shift-invert mode. Here $\mathtt{OP}=(\mathbf{A}-\sigma\mathbf{M})^{-1}\mathbf{M}$ with σ not an eigenvalue of $\mathbf{Ax}=\lambda\mathbf{Mx}$.

   The computed Ritz vectors are orthogonal with respect to the $\mathbf{M}$ innerproduct $\langle\mathbf{x},\mathbf{y}\rangle=\mathbf{x}^T\mathbf{My}$.

3. Buckling problem : $\mathbf{Ax}=\lambda\mathbf{Mx}$, $\mathbf{A}$ symmetric positive (semi) definite, $\mathbf{M}$ symmetric.

   This is solved using the following mode:

   (a) Buckling mode. Here $\mathtt{OP}=(\mathbf{A}-\sigma\mathbf{M})^{-1}\mathbf{A}$ with σ not equal to zero or to an eigenvalue of $\mathbf{Ax}=\lambda\mathbf{Mx}$.

   The computed Ritz vectors are orthogonal with respect to the $\mathbf{A}$ innerproduct $\langle\mathbf{x},\mathbf{y}\rangle=\mathbf{x}^T\mathbf{Ay}$.

In the shift-invert and buckling modes, σ is called the pole.

The method is described in detail by Meerbergen and Scott (2000), *The design of a block rational Lanczos code with partial reorthogonalization and implicit restarting*, Rutherford Technical Report RAL-TR-2000-011.

Note that EA16 is designed for computing a limited number of eigenvalues and eigenvectors. EA16 **cannot** be used for computing all the eigenpairs of $\mathbf{Ax}=\lambda\mathbf{x}$ or $\mathbf{Ax}=\lambda\mathbf{Mx}$. The matrices $\mathbf{A}$ and $\mathbf{M}$ need not be available in an explicit form.

ATTRIBUTES — **Version:** 1.4.1. (18th November 2021) **Types:** Real (single, double). **Precision:** At least 8-byte arithmetic is recommended. **Calls:** _LAMCH, _LASET, _LARNV, _SYEV, _GESVD, _SBEV, _LACPY, _LARTG, _LARFG, _RSCL (LAPACK), _NRM2, I_AMAX, _SCAL, _COPY, _GEMM, _AXPY, _GEMV, _GER, _TBSV (BLAS), KB07, KB08 (HSL). **Language:** Fortran 77. **Original date:** March 2000. **Origin:** K. Meerbergen and J.A. Scott, Rutherford Appleton Laboratory.

2 HOW TO USE THE PACKAGE

Although there are both single and double precision versions of the routine available, the user is **strongly advised** to use the double precision version unless single precision on his or her machine actually means 8-byte arithmetic.

2.1 Overall control and argument lists

There are three entries:

(a) EA16I/ID sets default values for the control parameters. It should normally be called once prior to any calls to EA16A/AD and EA16B/BD.

(b) EA16A/AD computes the amount of workspace that will be required by the Lanczos method. It must be called once before any calls to EA16B/BD.

(c) EA16B/BD computes selected eigenpairs. This routine must be called repeatedly using a reverse communication interface.

EA16 requires the multiplication of sets of vectors by the linear operator OP and by the matrix **M** (generalised problem) or by **A** (buckling problem). Reverse communication is used so that the user does not have to pass the matrices **A** and **M** to EA16 but, each time a multiplication is required, control is returned to the user. Note that multiplication of sets of vectors by OP involves solving a linear system of the form $(\mathbf{A}-\sigma\mathbf{I})\mathbf{U}=\mathbf{W}$ (shift-invert mode for the standard problem) or $(\mathbf{A}-\sigma\mathbf{M})\mathbf{U}=\mathbf{W}$ (shift-invert mode for the generalised problem and buckling mode) or $\mathbf{MU}=\mathbf{W}$ (regular inverse mode for the generalised problem). Two examples to illustrate the calling sequence are given in Section 5.

2.1.1 To set default values for the control parameters

The single precision version

```
CALL EA16I(ICNTL,CNTL)
```

The double precision version

```
CALL EA16ID(ICNTL,CNTL)
```

ICNTL is an INTEGER array of length 20 that need not be set by the user. On return it contains default values (see Section 2.2.1 for details).

CNTL is a REAL (DOUBLE PRECISION in the D version) array of length 15 that need not be set by the user. On return it contains default values (see Section 2.2.1 for details).

2.1.2 To calculate workspace requirements

The single precision version

```
CALL EA16A(N,BLK,NWANT,NV,LIWORK,LWORK,ICNTL,INFO)
```

The double precision version

```
CALL EA16AD(N,BLK,NWANT,NV,LIWORK,LWORK,ICNTL,INFO)
```

N is an INTEGER variable that must be set by the user to *n*, the order of the matrices **A** and **M**. This argument is not altered by the routine. **Restriction**: N > 3.

BLK is an INTEGER variable that must be set by the user to the block size for the Lanczos method. For an unblocked method, the user should set BLK = 1. It may be advantageous to use BLK > 1 when the multiplication of OP with a set of vectors allows the use of Level 3 BLAS kernels. This argument is not altered by the routine. **Restriction**: 1 ≤ BLK ≤ N/4.

NWANT is an INTEGER variable that must be set by the user to the number of required eigenvalues. This argument is not altered by the routine. **Restriction**: 1 ≤ NWANT ≤ N − 3*BLK.

NV is an INTEGER variable that must be set by the user to the maximum number of Lanczos vectors. NV is the number of columns of the matrix V used by EA16B/BD. This argument is not altered by the routine. **Restriction**: if NWANT is a multiple of BLK, then NV ≥ NWANT + 3*BLK, otherwise NV ≥ NWANT − MOD(NWANT,BLK) + 4*BLK. We advise setting NV ≥ 2*NWANT (see Section 3.2).

LIWORK is an INTEGER variable that need not be set by the user. On exit, LIWORK holds the minimum length of the work array IWORK required by EA16B/BD. LIWORK is of the order of NV.

LWORK is an INTEGER variable that need not be set by the user. On exit, LWORK holds the minimum length of the work array WORK required by EA16B/BD. LWORK is of the order of $3*(BLK+3)*(NV-BLK)+2*(MLANC+BLK)^2$ where MLANC = ICNTL(4)*BLK if $2 \le ICNTL(4) \le NV/BLK$, otherwise MLANC = NV.

ICNTL is an INTEGER array of length 20 that contains control parameters and must be set by the user. Default values may be set by a call to EA16I/ID. Details of the control parameters are given in Section 2.2.1. This argument is not altered by the routine.

INFO is an INTEGER array of length 20 that need not be set by the user. It is used to hold information about the execution of the subroutine. On exit from EA16A/AD, a value for INFO(1) of zero indicates that the subroutine has performed successfully. For nonzero values, see Section 2.3.

2.1.3 To calculate selected eigenvalues

The single precision version

```
      CALL EA16B(N,BLK,NWANT,NV,MODE,WHICH,IDO,IPOS,V,LDV,BV,LDBV,
     +           RANGE,SIGMA,NEINEG,IWORK,LIWORK,WORK,LWORK,ICNTL,CNTL,INFO)
```

The double precision version

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

N, BLK, NWANT, and NV must be unchanged since the call to EA16A/AD and are unchanged on exit.

MODE is an INTEGER variable that must be set by the user to the eigenvalue solver mode. The following modes are available:

- 1: Standard eigenvalue problem $\mathbf{Ax} = \lambda\mathbf{x}$, using regular mode and standard innerproduct. $OP = \mathbf{A}$.
- 2: Standard eigenvalue problem $\mathbf{Ax} = \lambda\mathbf{x}$, using shift-invert mode and standard innerproduct. $OP = (\mathbf{A} - \sigma\mathbf{I})^{-1}$.
- 3: Generalised eigenvalue problem $\mathbf{Ax} = \lambda\mathbf{Mx}$, using regular inverse mode and **M** innerproduct. $OP = \mathbf{M}^{-1}\mathbf{A}$.
- 4: Generalised eigenvalue problem $\mathbf{Ax} = \lambda\mathbf{Mx}$, using shift-invert mode and **M** innerproduct. $OP = (\mathbf{A} - \sigma\mathbf{M})^{-1}\mathbf{M}$. If **M** is semi-definite, the user must set ICNTL(15) > 0.
- 5: Buckling problem $\mathbf{Ax} = \lambda\mathbf{Mx}$, using shift-invert mode and **A** innerproduct. $OP = (\mathbf{A} - \sigma\mathbf{M})^{-1}\mathbf{A}$. If **A** is semi-definite the user must set ICNTL(15) > 0.

This argument is not altered by the routine. **Restriction**: MODE = 1, 2, 3, 4, 5.

WHICH is an INTEGER variable that must be set by the user to specify which eigenvalues of the original problem are required. Possible values are:

- 1: Eigenvalues furthest from the point RANGE(1) (MODE = 1 and 3 only). Ritz values are returned in order of descending distance from RANGE(1). If the eigenvalues of largest modulus are wanted, RANGE(1) should be set to zero.
- -1: Eigenvalues closest to the point RANGE(1). Ritz values are returned in order of ascending distance from RANGE(1). If the eigenvalues of smallest modulus are wanted, RANGE(1) should be set to zero.

-2/2: Right-most/left-most eigenvalues. Ritz values are returned in descending/ascending order.

-3/3: Half the number of wanted eigenvalues are computed from each end of the spectrum (MODE = 1 and 3 only). If NWANT is odd, WHICH = 3 computes one more Ritz value from the upper end than from the lower end and WHICH = -3 computes one more from the lower end than from the upper end. The Ritz values are returned alternating from both ends of the spectrum in decreasing distance from the centre of the spectrum, starting with the extreme value on the upper end when WHICH=3 and on the lower end when WHICH=–3.

-4/4: Eigenvalues to the right/left of the point RANGE(1). Ritz values are returned in ascending/descending order. If NWANT is greater than the number of Ritz values to the right/left of RANGE(1), the code computes less than NWANT eigenvalues and terminates with a warning (INFO(1) = 16).

5: Eigenvalues inside the interval (RANGE(1), RANGE(2)). Ritz values inside the interval are returned in ascending order. If NWANT is greater than the number of Ritz values inside the interval, the code computes less than NWANT eigenvalues and terminates with a warning (INFO(1) = 16).

10: Eigenvalues specified by the user (see IDO = 6). This option is intended only for experienced users. Ritz values are returned in order of descending priority.

This argument is not altered by the routine. **Restrictions**: WHICH = ±1, ±2, ±3, ±4, 5, 10. If WHICH = 1 or ±3, MODE must be equal to 1 or 3.

IDO is an INTEGER variable that is used as the reverse communication flag. Prior to the first call to EA16B/BD, IDO must be set by the user to 0. On exit, if IDO = 100, the computation has terminated. Otherwise, IDO ≠ 100 indicates convergence has not yet been achieved and to continue the computation the user must take the following action before recalling EA16B/BD:

1: The user must compute V(1:N, IPOS(3):IPOS(4)) = OP*V(1:N, IPOS(1):IPOS(2)). Note that BV(1:N, IPOS(3):IPOS(4)) already contains **M***V(1:N, IPOS(1):IPOS(2)) (MODE = 4) and **A***V(1:N, IPOS(1):IPOS(2)) (MODE = 5). The intervals [IPOS(1):IPOS(2)] and [IPOS(3):IPOS(4)] do not overlap and IPOS(2)–IPOS(1)+1 = IPOS(4)–IPOS(3)+1 = BLK. For MODE = 2, 4, or 5, each time the pole σ is changed, a return with IDO = 1 is preceded by a return with IDO = 4. The factorization performed with IDO = 4 can be used to apply OP.

2: MODE = 3, 4, or 5 only. If MODE = 3 or 4, the user must compute
BV(1:N, IPOS(3):IPOS(4)) = **M***V(1:N, IPOS(1):IPOS(2)).
If MODE = 5, the user must compute BV(1:N, IPOS(3):IPOS(4)) = **A***V(1:N, IPOS(1):IPOS(2)).
Note that IPOS(2)–IPOS(1)+1 = IPOS(4)–IPOS(3)+1 is at most BLK.

3: MODE = 2, 4, or 5 with ICNTL(5) < 0 only. The user may select a new pole σ for the Lanczos process. The pole must be placed in SIGMA. The current Ritz values for the original eigenvalue problem are in WORK(IPOS(5):IPOS(6)) and the first NCONV = INFO(7) of these have converged. WORK(IPOS(5):IPOS(5)+NCONV-1) and WORK(IPOS(5)+NCONV:IPOS(6)) are reordered following WHICH. If the user does not want to change the pole, SIGMA should be unchanged.

4: MODE = 2, 4, 5 only. The user must set NEINEG to the number of negative eigenvalues of $\mathbf{A}-\sigma\mathbf{I}$ (MODE = 2) or $\mathbf{A}-\sigma\mathbf{M}$ (MODE = 4 or 5) where σ = SIGMA (see Section 3.2 for guidance). If the user is not able to provide this information, NEINEG should be given a negative value. In addition, if the user is using a direct linear solver for performing the multiplication of vectors by OP, the user must factorize $\mathbf{A}-\sigma\mathbf{I}$ (MODE = 2) or $\mathbf{A}-\sigma\mathbf{M}$ (MODE = 4 or 5). 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 the pole SIGMA has been changed. The user can communicate a failure of the matrix factorization for the pole SIGMA by setting IDO = –4 and recalling EA16B/BD, with no other changes to the parameters. If EA16B/BD is not able to continue, the computation will terminate with the error flag INFO(1) = –16.

5: WHICH = ±2, 10 and ICNTL(7) ≠ 0 only. The user may change MODE from 1 to 2 or from 3 to 4. Other

changes to MODE raise an error and cause the code to stop. If ICNTL(5) ≥ 0, the pole that is in SIGMA will be used; if ICNTL(5) < 0, the user can choose a pole, which must be placed in SIGMA.

6: WHICH = 10 only. The user must select the wanted and unwanted Ritz values by giving each Ritz value a priority. If a Ritz value is not wanted, the user should give it zero priority; the most important Ritz values should be given the highest priority. Note that more than one Ritz value can be given the same priority. In the simplest case, the user divides the Ritz values into two classes: the wanted Ritz values (priority 1) and the unwanted Ritz values (priority 0). The current Ritz values are in WORK(IPOS(5):IPOS(6)) and are for the original problem, not for OP. The user must supply NRITZ = IPOS(10) – IPOS(9) + 1 priority values in IWORK(IPOS(9): IPOS(10)), with the priority for the current Ritz value which is in
WORK(IPOS(5)+K-1) in position IWORK(IPOS(9)+K-1), K = 1,2,..., NRITZ.

7: ICNTL(9) = 10 only. The user must supply NSHIFT = IPOS(8) – IPOS(7) + 1 implicit shifts (see Section 4) in WORK(IPOS(7):IPOS(8)). The current Ritz values are in WORK(IPOS(5):IPOS(6)) and the first NCONV = INFO(7) of these have converged. The Ritz values may be in arbitrary order. They are the Ritz values corresponding to the operator OP and not to the original eigenvalue problem. This is important since the implicit shifts are applied to the operator OP and not to the original problem.

IPOS is an INTEGER array of length 10 that need not be set by the user. IPOS is used to point to locations in the arrays V, BV, WORK, and IWORK of matrices (and vectors) required by the Lanczos process.

V is a REAL (DOUBLE PRECISION in the D version) array with dimensions (LDV, NV). If ICNTL(3) = 0 (the default), V need not be set by the user. If ICNTL(3) ≠ 0, on the first call to EA16B/BD (IDO = 0), the first BLK columns of V must contain BLK basis vectors (these vectors need not be orthogonal). If some or all of the basis vectors are found by EA16 to be linearly dependent, the dependent vectors are replaced by random vectors. On successful exit with IDO = 100, the first INFO(7) columns of V contain the converged Ritz vectors, ordered according to WHICH.

LDV is an INTEGER variable that must be set by the user to the first dimension of the array V. This argument is not altered by the routine. **Restriction:** LDV ≥ N.

BV is a REAL (DOUBLE PRECISION in the D version) array with leading dimension LDBV. The array is not accessed if MODE = 1 or 2 and, in this case, the second dimension may be 1. If MODE = 3, 4, or 5, the second dimension must be at least BLK.

LDBV is an INTEGER variable that must be set by the user to the first dimension of the array BV. This argument is not altered by the routine. **Restriction:** LDBV ≥ 1 (MODE = 1 or 2), LDBV ≥ N (MODE = 3, 4, or 5).

RANGE is a REAL (DOUBLE PRECISION in the D version) array of length 2. If WHICH = 5, RANGE(1) and RANGE(2) must be set by the user prior to the first call to EA16A/AD to the end points of the interval inside of which eigenvalues are required. If WHICH = ±1 or ±4, RANGE(1) must be set by the user (see WHICH) and RANGE(2) is not accessed. RANGE(1) and RANGE(2) must not be set equal to an eigenvalue since this may endanger safe pole selection. Note that when MODE = 5 and **A** is positive semi definite, zero is an eigenvalue. RANGE is not accessed for other values of WHICH. This argument is not altered by the routine. **Restrictions**: If WHICH = 5, RANGE(2) > RANGE(1). If WHICH = 5, MODE = 5 and ICNTL(15) > 0, RANGE(1) and RANGE(2) must have the same sign.

SIGMA is a REAL (DOUBLE PRECISION in the D version) variable. SIGMA is only used if MODE = 2, 4, or 5. In these cases, if WHICH = 2, -2 or 10, SIGMA must be set by the user prior to the first call to EA16B/BD to the pole σ in the shift-invert operator. SIGMA must not be selected equal or close to an eigenvalue. The following restrictions hold. When MODE = 5, the user must not pick SIGMA equal to zero. SIGMA is unchanged by the code if ICNTL(5) = 0 (the default). If ICNTL(5) > 0, SIGMA may be changed by the code and must not be altered by the user; if ICNTL(5) < 0, the user may choose a new value for SIGMA when IDO = 3 is returned. If the user changes from MODE = 1 to 2 (or from 3 to 4) (ICNTL(7) ≠ 0 only) and ICNTL(5) < 0, the user may select a value for SIGMA when IDO = 3 is returned; otherwise, SIGMA is set by the code and should not be altered by the user.

NEINEG is an INTEGER variable. NEINEG is only used if MODE = 2, 4, or 5. In these cases, when IDO = 4 is returned the user must set NEINEG to the number of negative eigenvalues of $\mathbf{A}-\sigma\mathbf{I}$ (MODE = 2) or $\mathbf{A}-\sigma\mathbf{M}$ (MODE = 4 or 5), where σ=SIGMA. If the user cannot compute this number, NEINEG should be given a negative value. This argument is not altered by the routine.

IWORK is an INTEGER array of length LIWORK. On return from EA16B/BD with IDO = 100, IWORK(1:4) contains the last seeds used by the LAPACK random number generator _LARNV to generate random vectors for the Lanczos method. IWORK is used by the routine as a work array and, if WHICH=10, to hold priority values (see IDO=6).

LIWORK is an INTEGER variable that must be unchanged since the call to EA16A/AD. This argument is not altered by the routine.

WORK is a REAL (DOUBLE PRECISION in the D version) array of length LWORK. This array is used as a work array and for the reverse communication. On successful exit with IDO = 100, the INFO(7) converged Ritz values are in WORK(IPOS(5):IPOS(6)), ordered according to WHICH, and if ICNTL(16)$\neq$0, with the corresponding scaled norms of the residuals in WORK(IPOS(7):IPOS(8)). Note that INFO(7) = IPOS(8)-IPOS(7)+1. The first INFO(8) values in WORK(IPOS(5):IPOS(6)) are the wanted Ritz values. WORK(1:4) contain trust interval information when MODE = 2, 4, or 5. EA16B/BD guarantees that the eigenvalues in the intervals $(-\infty,$WORK(1)], [WORK(2),WORK(3)] and [WORK(4), $+\infty)$ are computed. If WORK(2) > WORK(3), the interval [WORK(2),WORK(3)] is empty.

LWORK is an INTEGER variable that must be unchanged since the call to EA16A/AD. This argument is not altered by the routine.

ICNTL is an INTEGER array of length 20 that contains control parameters and must be set by the user. Default values may be set by a call to EA16I/ID. ICNTL must be unchanged (except ICNTL(6)) since the call to EA16A/AD. Details of the control parameters are given in Section 2.2.1. This argument is not altered by the routine.

CNTL is a REAL (DOUBLE PRECISION in the D version) array of length 15 that contains control parameters and must be set by the user. Default values may be set by a call to EA16I/ID. Details of the control parameters are given in Section 2.2.1. This argument is not altered by the routine.

INFO is an INTEGER array of length 20 that need not be set by the user. It is used to hold information about the execution of the subroutine. On exit from EA16B/BD, a value for INFO(1) of zero indicates that the subroutine has performed successfully. For nonzero values, see Section 2.3. For details of the information output in the other components, see Section 2.2.2.

2.2 Arrays for control and information

2.2.1 Control parameters

The arrays ICNTL and CNTL control the action of EA16A/AD and EA16B/BD. Default values may be set by calling EA16I/ID.

ICNTL(1) is the stream number for error, warning, and diagnostic messages and has the default value 6. Printing is suppressed if ICNTL(1)<0.

ICNTL(2) is used to control the level of printing. The default value is 2. Possible values are:

- 0 No messages are printed.
- 1 Only error messages are printed.
- 2 Error and warning messages printed.
- 3 As for 2, plus scalar parameters and the control parameters on the first entry to EA16B/BD, and scalar parameters and the information array INFO on exit from EA16B/BD.
- 4 As for 3, plus converged Ritz values on exit with IDO = 100 and INFO(1)$\geq$0.

5 As for 4, plus information after each implicit restart (see Section 4).

ICNTL(3) controls whether initial basis vectors are user-supplied. If ICNTL(3) = 0 (the default), vectors are not supplied by the user and random initial vectors are generated by the code. If the user wishes to supply vectors, then prior to the first call to EA16B/BD, they must be placed in the first BLK columns of the array V and ICNTL(3) must be set to a nonzero value.

ICNTL(4) is the number of steps of the Lanczos process before the computation of Ritz values and an implicit restart (see Section 4). If ICNTL(4) < 2 or larger than NV/BLK-1, the maximum number of steps possible is performed. The default value is 0.

ICNTL(5) controls whether the pole SIGMA in the shift-invert operator is fixed (MODE = 2, 4, or 5 only). The default is ICNTL(5) = 0. The following options are available:

0 The pole is fixed.

1 (respectively, -1) The pole may be changed by EA16 (respectively, the user) when the elapsed CPU time since the last call with IDO = 4 is sufficiently large (see ICNTL(17)).

2 (respectively, -2) The pole may be changed by EA16 (respectively, the user) every ICNTL(18) restarts.

3 (respectively, -3) The pole may be changed by EA16 (respectively, the user) when either the elapsed CPU time since the last call with IDO = 4 is sufficiently large (see ICNTL(17)) or after ICNTL(18) restarts.

ICNTL(5) is not used if MODE = 1 or 3.

ICNTL(6) controls the maximum number of multiplications of sets of vectors by the operator OP allowed by EA16B/BD (that is, the number of returns from EA16B/BD with IDO = 1). The maximum is given by ICNTL(6) * N. The default value is 2. Note that this parameter ensures finite termination. Values smaller than 1 are treated as the default.

ICNTL(7) is used to allow the user to change MODE (WHICH = ±2 or 10 only). If ICNTL(7) = 0 (the default), no change is allowed. Otherwise, when IDO = 5 is returned, the user may change from MODE = 1 to 2, or from MODE = 3 to 4. Other changes lead to an error (INFO(1) = -1). Once made, a change is irreversible. Note that if WHICH = ±1, ±3, ±4, or 5, ICNTL(7) is not accessed.

ICNTL(8) controls the size of the Krylov subspace after a reduction of the subspace (see Section 4). After an implicit restart, the Krylov subspace is reduced to ICNTL(8) percent of its previous dimension. The default value is 50. ICNTL(8) should lie between 1 and 99; if ICNTL(8) lies outside this range, a warning is issued and the default value is used. Note that if the code is not able to reduce the subspace dimension to ICNTL(8) percent of its previous dimension, EA16B/BD will choose a suitable reduction parameter.

ICNTL(9) controls the implicit restarting (see Section 4). The default is ICNTL(9) = 1. The following options are available:

1 Purging of unwanted Ritz values.

2 Exact shifts.

4 Chebyshev shifts. This choice of shifts is not allowed for all combinations of WHICH and MODE. When this choice is not allowed, a warning is issued and the default is used.

10 User-defined shifts (see IDO = 7).

ICNTL(10:13) contains the initial seeds for the LAPACK random number generator _LARNV. The default values are (0,0,0,1). ICNTL(10:13) must lie be between 0 and 4095 and ICNTL(13) must be odd. If these conditions are broken, the code continues with seeds that satisfy these bounds and a warning is issued (INFO(1) = 4).

ICNTL(14) is not currently used.

ICNTL(15) is only accessed for the generalised and buckling problems and has default value 0. Values other than the default should only be used if **M** is positive semi-definite (MODE = 3 or 4) or **A** is positive semi-definite (MODE = 5). The generalised eigenvalue problem has an infinite eigenvalue when **M** is positive semi-definite. Usually, the user is not interested in the computation of the infinite eigenvalue. This can be avoided by setting ICNTL(15) equal to the index of the infinite eigenvalue: this is usually 1 but is equal to 2 if the eigenvalue is defective. If the index is not known, ICNTL(15) should be set to 2. The buckling problem has a zero eigenvalue when **A** is positive semi-definite. If its computation is not required, the user should set ICNTL(15) to the index of the zero eigenvalue (again 1 or 2). If **A** has very small non-zero eigenvalues, higher values of ICNTL(15) may be used. See Section 4 for further explanation. Negative values are treated as zero.

ICNTL(16) has default value 0. If ICNTL(16) $\neq$ 0, the scaled residual norms

$$\|\mathtt{OP}*\mathbf{x}-\theta\mathbf{x}\|/|\theta|$$

of the converged Ritz values are computed and stored in WORK(IPOS(7):IPOS(8)) just before the final return to the user (IDO = 100). Computation of the residual norms may require significant additional work and it may be more efficient for the user to compute the residuals himself. If a negative innerproduct is encountered during the computation of a residual, the corresponding residual is set to zero (an error is not returned so that the computation of the remaining residuals can be completed).

ICNTL(17) is only accessed if |ICNTL(5)| = 1 or 3. In these cases, let T1 be the CPU time when IDO = 4 is returned to the user and let T2 be the CPU time when the user recalls EA16B/BD with IDO = 4. If T3 > T2 is the current CPU time, a new pole is selected once the time T3 – T2 used by EA16 exceeds |ICNTL(17)| percent of the factorization time T2 – T1. The default value is 200.

ICNTL(18) is only accessed if |ICNTL(5)| = 2 or 3. In these cases, |ICNTL(18)| is the number of restarts between a change of pole. The default value is 1.

ICNTL(19) and ICNTL(20) are not currently used.

CNTL(1) controls the orthogonalization of the Lanczos vectors. On return with IDO = 100, $\max\{\langle \mathtt{V}(:,\mathtt{I}),\mathtt{V}(:,\mathtt{J})\rangle : 1\le \mathtt{I},\mathtt{J}\le \mathtt{INFO(7)}, \mathtt{I}\neq\mathtt{J}\}$ is of the order of CNTL(1). CNTL(1) has default value $\sqrt{u}$, where u is the machine precision.

CNTL(2) and CNTL(3) are convergence tolerances. A computed Ritz pair $(\mathbf{x},\lambda)$ is accepted as an approximate eigenpair if

$$\|\mathtt{OP}*\mathbf{x}-\theta\mathbf{x}\| \le u*\|\mathbf{T}\|_2 + |\mathtt{CNTL(2)}| + |\mathtt{CNTL(3)}|*|\theta|,$$

where **T** is the Lanczos matrix (see Section 4), u is the machine precision and $\theta=\lambda$ for MODE = 1 or 3, $\theta=(\lambda-\sigma)^{-1}$ for MODE = 2 or 4, and $\theta=(\lambda-\sigma)^{-1}\lambda$ for MODE = 5. The default values for CNTL(2) and CNTL(3) are zero and $\sqrt{u}$, respectively. The norm ||.|| is induced from the innerproduct <.,.> and $\|.\|_2$ denotes the two-norm.

CNTL(4) controls the growth of rounding errors and the distance of the new pole to the Ritz values when the pole is changed. CNTL(4) has default value 500.0. After the change of pole, the rounding errors are bounded by

$$\|\mathbf{T}\|_2 u\,|\mathtt{CNTL(4)}|,$$

where **T** is the Lanczos matrix (see Section 4) and u is the machine precision. When CNTL(4) is close to 1, the choice of the new pole is more restricted, but the growth of rounding errors due to the change of pole is reduced. If |CNTL(4)| is larger, the choice of the new pole is less restricted, but the growth of rounding errors due to the change of pole may grow. In addition, the solution of linear systems with $\mathbf{A}-\sigma\mathbf{I}$ or $\mathbf{A}-\sigma\mathbf{M}$ may be ill-conditioned. CNTL(4) is not accessed if MODE = 1 or 3.

CNTL(5) controls the growth in the two-norm of the Lanczos vectors when MODE = 4 or 5 and ICNTL(15) > 0. EA16 aims to keep $\|\mathtt{V}(:,\mathtt{J})\|_2 \le |\mathtt{CNTL(5)}|*\|\mathtt{V}(:,1)\|_2$ for J = 1,..., NV. A large growth in the two-norms can lead to inaccurate eigenvalues and eigenvectors or to negative innerproducts (see INFO(1) = –11). CNTL(5) has

default value $u^{\frac{1}{4}}$, where u is the machine precision.

CNTL(6) to CNTL(10) are not currently used.

2.2.2 Information arrays

The array INFO is used to provide the user with information on the execution of EA16A/AD and EA16B/BD.

INFO(1) has the value zero if a call was successful, has a positive value if a warning has been issued, and a negative value in the event of an error (see Section 2.3).

INFO(2) returns additional information in the event of an error (see Section 2.3.)

INFO(3) holds the number of operations performed with OP. This is the number of calls to EA16B/BD with IDO = 1.

INFO(4) holds the number of operations performed with **M** (MODE = 3 or 4) or **A** (MODE = 5). This is the number of calls to EA16B/BD with IDO = 2.

INFO(5) holds the number of restarts of the implicitly restarted Lanczos method.

INFO(6) is the number of matrix factorizations. This is the number of calls to EA16B/BD with IDO = 4.

INFO(7) is the number of converged Ritz values and vectors.

INFO(8) is the minimum of NWANT and the number of converged Ritz values satisfying WHICH (this may be less than INFO(7)).

INFO(9) is the number of converged Ritz values satisfying WHICH (this may be less than INFO(7)).

INFO(10:20) are set but do not currently contain information of interest to the user.

2.3 Error and warning diagnostics

If EA16A/AD or EA16B/BD returns with a negative value of INFO(1), an error has occurred; if EA16B/BD returns with a positive value of INFO(1), a warning has been issued. Messages are output on unit ICNTL(1). Possible negative values of INFO(1) are:

–1 Value of MODE out of range (first call to EA16B/BD). Immediate return with input parameters unchanged. This error is also returned if MODE has been given an incorrect value by the user on a call with IDO = 5.

–2 Value of N out of range (EA16A/AD or first call to EA16B/BD). Immediate return with input parameters unchanged.

–3 Value of NV is too small (EA16A/AD or first call to EA16B/BD). INFO(2) holds the minimum value for NV. Immediate return with input parameters unchanged.

–4 Value of NWANT out of range (EA16A/AD or first call of EA16B/BD). Immediate return with input parameters unchanged.

–5 Value of BLK out of range (EA16A/AD or first call to EA16B/BD). Immediate return with input parameters unchanged.

–6 The combination of WHICH and MODE is not allowed.

–7 Value of WHICH out of range (first call to EA16B/BD). Immediate return with input parameters unchanged.

–8 Value of LDV out of range (first call to EA16B/BD). Immediate return with input parameters unchanged.

–9 Value of LDBV out of range (first call to EA16B/BD). Immediate return with input parameters unchanged.

–10 WHICH = 5 and RANGE(2) ≤ RANGE(1) or WHICH = 5, MODE = 5, ICNTL(15) > 0 and RANGE(1) < 0 and RANGE(2) > 0 (first call to EA16B/BD). Immediate return with input parameters unchanged.

–11 The **A** (MODE = 5) or **M** (MODE = 3 or 4) innerproduct is negative.

–12 Value of LIWORK is too small (first call to EA16B/BD). Immediate return with input parameters unchanged. Use EA16A/AD to compute its minimum value.

–13 Value of LWORK is too small (first call to EA16B/BD). Immediate return with input parameters unchanged. Use EA16A/AD to compute its minimum value.

–14 The number of operations specified by ICNTL(6) has been exceeded. The user may increase ICNTL(6) and recall EA16B/BD (with no other changes to the input parameters).

–15 Initial SIGMA is too close to zero for MODE = 5.

–16 IDO = –4 was set by the user, but the code was not able to suggest a better value of SIGMA.

Positive values of INFO(1) associated with a warning are:

1 NV is larger than N. A value equal to N is used (call to EA16A/AD and first call to EA16B/BD).

2 ICNTL(8) out of range (call to EA16A/AD and first call to EA16B/BD). The default value is used.

4 The seeds in ICNTL(10:13) lie out of range (first call to EA16B/BD). The code chooses appropriate seeds.

8 CNTL(4) < 1.0. The default 500.0 is used (first call to EA16B/BD).

16 The number of computed Ritz values is less than the number requested (EA16B/BD). This may be because NWANT is either greater than the number of eigenvalues to the right/left of RANGE(1) (WHICH = –4/4) or is greater than the number of eigenvalues inside the interval (RANGE(1), RANGE(2)) (WHICH = 5).

32 Either the requested choice of shifts (ICNTL(9)) cannot be used with the chosen MODE or WHICH, or ICNTL(9) is out of range. The default choice of shifts is used.

64 The pole has not been altered.

Positive values of INFO(1) are summed so that the user can identify all warnings issued by EA16B/BD, e.g. INFO(1) = 3 indicates both warnings 1 and 2 are raised.

3 GENERAL INFORMATION

3.1 Summary of information.

Use of common: Common blocks are not used.

Other routines called directly: The HSL routines KB07A/AD, KB08A/AD; the LAPACK routines, SLAMCH/DLAMCH, SLASET/DLASET, SLARNV/DLARNV, SSYEV/DSYEV, SGESVD/DGESVD, SSBEV/DSBEV, SLACPY/DLACPY, SLARTG/DLARTG, SLARFG/DLARFG, SRSCL/DRSCL; the BLAS routines SNRM2/DNRM2, ISAMAX/IDAMAX, SSCAL/DSCAL, COPY, GEMM, SAXPY/DAXPY, SGEMV/DGEMV, SGER/DGER, STBSV/DTBSV. Procedures internal to EA16 are EA16E/ED to EA16Z/ZD, EA17A/AD to EA17Z/ZD, and EA18A/AD to EA18Z/ZD.

Workspace: Workspace is provided by the arrays IWORK(LIWORK) and WORK(LWORK).

Input/output: The output stream for the error and warning messages and diagnostic printing is ICNTL(1) (see Section 2.2.1).

Restrictions:
N > 3.
If NWANT is a multiple of BLK, then NV ≥ NWANT+3*BLK, otherwise NV ≥ NWANT–MOD(NWANT,BLK)+4*BLK.
1 ≤ NWANT ≤ N-3*BLK.
1 ≤ BLK ≤ N/3.
MODE = 1, 2, 3, 4, 5.
WHICH = ±1, ±2, ±3, ±4, 5, 10.
LDV ≥ N.
LDBV ≥ 1 (MODE = 1 or 2), LDBV ≥ N (MODE = 3, 4, or 5).

If WHICH = 5, RANGE(2) > RANGE(1).

3.2 Guidelines for the user

In this section, we offer some guidelines to the user for setting some of the input parameters.

+ It is generally satisfactory to use a blocksize BLK of 1. If the operations performed with IDO = 1 or 2 are much more efficient for higher values of BLK (for example, if the use is able to employ BLAS 3 kernels), it is advisable to choose BLK larger than one. In the literature, choosing BLK equal to the multiplicity of the wanted eigenvalues is often proposed. If BLK = 1, multiplicities are sometimes missed. For reasons of efficiency, we do not recommend using large values of BLK, i.e. larger than 10.
+ For many applications, unless NV is large, the regular modes MODE = 1 and 3 converge very slowly. For WHICH = -1, ±4, and 5, the use of shift-invert is highly recommended.
+ NV is typically limited by the amount of memory available. We suggest NV be chosen larger than the maximum of 2∗NWANT and NWANT + 10∗BLK.
+ If WHICH = ±2 and the user wishes to use shift-invert mode but does not know how to choose an appropriate initial SIGMA, we recommend starting the computation using the regular mode and ICNTL(7) ≠ 0, then switching to shift-invert mode when IDO = 5 is returned.
+ When a large number of eigenvalues are wanted, NV must be large. The Ritz values are computed every NV/BLK-1 Lanczos iterations. To compute the Ritz values more frequently, the user should use ICNTL(4). For example, if NWANT = 100, BLK = 1, and NV = 200, by default the code computes the Ritz values, checks for convergence, performs an implicit restart, and changes the pole every NV – BLK = 199 iterations. The user can ask the code to do this, for example, every 100 iterations by setting ICNTL(4) = 100.
+ The buckling mode should only be used for the generalised problem when **M** is not positive (semi) definite. When **A** is semi-definite in the buckling mode and ICNTL(15) > 0, the eigenvalues with small modulus will fail to converge or converge very slowly.
+ If **M** is positive (semi) definite there are a number of sparse direct solvers, including the HSL routines MA27 and MA57, that compute a factorization of the form $\mathbf{A} - \sigma\mathbf{M} = \mathbf{L}\mathbf{D}\mathbf{L}^T$ (**L** unit lower triangular and **D** diagonal, possibly with 2×2 blocks on the diagonal) and return the number of negative eigenvalues to the user. The user can use this to set NEINEG.
+ WHICH = 10 is designed for experienced users. We recommend the user tries combinations of existing options first. For example, when the user wants to compute the eigenvalues outside an interval, he or she can first compute the eigenvalues to the left with WHICH = 4 and then to the right of the interval with WHICH = -4.
+ Subspace iteration may be simulated by setting ICNTL(9) = 10 and selecting WORK(IPOS(7):IPOS(8)) = 0 when IDO = 7 is returned. With these settings, implicit restarting with zero shifts is used. Note that this is only useful for computing the dominant eigenvalues of OP.

4 METHOD

The algorithm used by EA16 is the block Lanczos method described in detail by Meerbergen and Scott (2000). Implicit restarting is employed to limit the Krylov subspace dimension. We also allow for a change of pole when convergence is slow. The outline of the method is as follows:

(1) Set BLK starting vectors of size N in $\mathbf{V}_1$. The default is for these to be chosen randomly by the code.

(2) Compute the Lanczos basis $\mathbf{V}_{l+1}$ of dimension l ∗ BLK+BLK through l Lanczos steps. The construction of the basis leads to the symmetric band matrix $\mathbf{T}_l$ of order l∗BLK and bandwidth 2*BLK+1.

(3) **Do**

  (a) Expand the Lanczos basis $\mathbf{V}_{l+1}$ to the basis $\mathbf{V}_{m+1}$ of dimension m ∗ BLK+BLK through (m–l) Lanczos

steps. The construction of the basis leads to a symmetric band matrix $\mathbf{T}_m$ of order $m*$BLK and bandwidth 2*BLK+1.

(b) Compute the eigenvalues θ and eigenvectors $\mathbf{z}$ of $\mathbf{T}_m$. The eigenvalues are Ritz values of OP and $\mathbf{x}=\mathbf{V}_m\mathbf{z}$ is the corresponding Ritz vector. Compute the corresponding residual norm $\|\mathtt{OP}*\mathbf{x}-\theta\mathbf{x}\|$, where $\theta=\lambda$ for MODE = 1 or 3, $\theta=(\lambda-\sigma)^{-1}$ for MODE = 2 or 4, and $\theta=(\lambda-\sigma)^{-1}\lambda$ for MODE = 5. The norm ||.|| is induced from the innerproduct <.,.>. Note that we are able to avoid computing the residual norm explicitly.

(c) Check whether the desired Ritz values and vectors satisfy the convergence criterion

$$\|\mathtt{OP}*\mathbf{x}-\theta\mathbf{x}\| \le u*\|\mathbf{T}\|_2 + |\mathtt{CNTL(2)}| + |\mathtt{CNTL(3)}|*|\theta|,$$

where u is the machine precision.

Stop when all desired Ritz values satisfy this criterion.

(d) **Stop** when the maximum number of operations with OP (ICNTL(6)) is reached.

(e) Reduce the dimension of the Lanczos basis by implicit restarting. Transform $\mathbf{T}_m$ into $\mathbf{T}_l$ and $\mathbf{V}_m$ into $\mathbf{V}_l$ with $l=$ICNTL(8)$*m/$100.

(f) Optionally update the pole σ.

In Step (e), the following choices for the reduction of the subspace dimension are provided. When ICNTL(9) = 1, purging is used, i.e. the subspace dimension is reduced by keeping ICNTL(8) percent of the Ritz vectors. The other Ritz vectors are discarded. Reducing the dimension of the subspace always reduces the speed of convergence, but by keeping good Ritz vectors in the subspace, the convergence should not be significantly affected. When performing implicit restarting, a polynomial in OP is applied to the subspace, i.e.

Range(NEW V(:, 1:l*BLK)) = ψ(OP) * Range(OLD V(:, 1:l*BLK)),

where ψ is a polynomial of degree m–l. Its zeros are called shifts. The new subspace is equal to the old subspace minus the subspace that is filtered away by ψ(OP). The part of the subspace that does not contribute significantly to the convergence can be removed by a judicious choice of shifts. The removal of this subspace reduces the dimension of the Krylov space from m*BLK to l*BLK. If exact shifts are used, the shifts are chosen to be unwanted Ritz values. The removed subspace then contains the subspace spanned by the corresponding Ritz vectors. If Chebyshev shifts are chosen, ψ is a Chebyshev polynomial. This choice is suited for filtering eigenvalues in an interval, since Chebyshev polynomials have a minimisation property on an interval.

For the generalised problem with positive semi-definite **M**, to avoid the computation of the infinite eigenvalue we advise setting ICNTL(15) = 1 or 2. Similarly, setting ICNTL(15) = 1 or 2 avoids computing the zero eigenvalue of the buckling problem when **A** is positive semi-definite. In both cases, the computation of the zero eigenvalue of OP is prevented.

When ICNTL(15) > 0, the code removes from the initial vectors components in the direction of the nullspace of OP to the power ICNTL(15). A similar operation is performed on all basis vectors before the computation of the Ritz values. This operation keeps the Krylov space out of the nullspace of OP. A final filtering operation is performed on the converged Ritz vectors, since they might have components in the direction of the nullspace of OP.

Reference K. Meerbergen and J. Scott (2000), *The design of a block rational Lanczos code with partial reorthogonalization and implicit restarting,* Rutherford Technical Report RAL-TR-2000-011. Available from www.numerical.rl.ac.uk/reports/reports.html

5 EXAMPLE OF USE

5.1 Standard eigenvalue problem

The first example illustrates the computation of the dominant eigenvalues of the tridiagonal matrix **A** with 2 on the main diagonal and −1 on the off-diagonals. This matrix is positive definite. We want to compute the six dominant eigenvalues and corresponding eigenvectors. The only operation that the user must provide is a matrix-vector product with **A**.

```
      INTEGER N, BLK, NWANT, NV, MODE, WHICH, LDV, LDBV
      PARAMETER (N=20, BLK=1, NWANT=6, NV=16, MODE=1, WHICH=1)
      PARAMETER (LDV=N, LDBV=1)
      INTEGER LIW, LW
      PARAMETER (LIW=500, LW=1100)
      INTEGER IWORK(LIW)
      DOUBLE PRECISION WORK(LW), V(LDV,NV)

      INTEGER IDO, LIWORK, LWORK, NEINEG, IPOS(10), INFO(20)
      DOUBLE PRECISION SIGMA, RANGE(2), BV(1,1)

      INTEGER ICNTL(20)
      DOUBLE PRECISION CNTL(15)

C     Set the control parameters.
      CALL EA16ID(ICNTL,CNTL)
C     Increase the print level
      ICNTL(2) = 4
C     Allow up to five multiplications of sets of vectors
      ICNTL(6) = 5
C     Ask for residuals
      ICNTL(16) = 1
C     Print diagnostic output to file
      ICNTL(1) = 10
      OPEN(UNIT=ICNTL(1),FILE='ea16ds.capture')

C     Compute the amount of storage required for this routine.
      CALL EA16AD(N,BLK,NWANT,NV,LIWORK,LWORK,ICNTL,INFO)

C     Check the size of the workspace.
      IF (LIWORK.GT.LIW) THEN
         WRITE (6,'(A,I4)') 'Increase LIW to at least ',LIWORK
         STOP
      ELSE IF (LWORK.GT.LW) THEN
         WRITE (6,'(A,I4)') 'Increase LW to at least ',LWORK
         STOP
      END IF

C     Compute NWANT dominant eigenvalues
C     (i.e. furthest from RANGE(1)=0.D0)
      RANGE(1) = 0.D0

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 * V(:,IPOS(1):IPOS(2))
            CALL MATVEC(N, IPOS(4)-IPOS(3)+1, V(1,IPOS(1)),LDV,
     &                  V(1,IPOS(3)), LDV)
            GO TO 1
         END IF
 2    CONTINUE

C     Check for errors
      IF (INFO(1).LT.0) THEN
         WRITE (6,*) 'There was an error : INFO(1) = ', INFO(1)
         STOP
      ELSE
         WRITE(6,*) 'The computed approximate eigenvalues ',
     &        '(Ritz values) are: '
         WRITE(6,FMT='(F11.4)') WORK(IPOS(5):IPOS(6))
      END IF

      STOP
      END
C*******************************************************

      SUBROUTINE MATVEC(N, M, X, LDX, Y, LDY)

C Compute the matrix vector product with A

      INTEGER N, M, LDX, LDY
      DOUBLE PRECISION X(LDX,M), Y(LDY,M)

      INTEGER K, I

      DO 10 K=1,M
         Y(1,K) = 2*X(1,K) - X(2,K)
         DO 100 I=2,N-1
            Y(I,K) = 2*X(I,K) - X(I+1,K) - X(I-1,K)
 100     CONTINUE
         Y(N,K) = 2*X(N,K) - X(N-1,K)
 10   CONTINUE
      RETURN
      END
```

The output produced is:

```
 The computed approximate eigenvalues (Ritz values) are:
     3.9777
     3.9111
     3.8019
     3.6525
     3.4661
     3.2470
     3.0000
```

The file ea16ds.capture is produced, which should have output similar to:

```
-----------------------------------------------------
Arguments of EA16BD on input :
  MODE     :          1
  IDO      :          0
  LDV      :         20
  LDBV     :          1
```

```
  N        :         20
  NV       :         16
  NWANT    :          6
  BLK      :          1
  WHICH    :          1
  RANGE(1) :  0.000000E+00
  LIWORK   :        141
  LWORK    :        760
  ICNTL    :          6          4          0          0          0
                      5          0         50          1          0
                      0          0          1        100          0
                      1        200          1
  CNTL     :  1.054E-08  0.000E+00  1.054E-08  5.000E+02  1.026E-04
-------------------------------------------------------
-------------------------------------------------------
Arguments of EA16BD on output :
IPOS   :          7          7         16         16         85
                 91         70         76          0          0
INFO   :          0          0         39          0          3
                  0          7          6          6
-------------------------------------------------------
+------- Ritz values / residual norms on output -------+

      1:    3.97766E+00 /    2.23957E-12
      2:    3.91115E+00 /    2.02861E-11
      3:    3.80194E+00 /    4.34799E-11
      4:    3.65248E+00 /    2.14723E-10
      5:    3.46610E+00 /    7.90670E-10
      6:    3.24698E+00 /    3.51993E-12
      7:    3.00000E+00 /    1.50700E-10
+-------------------------------------------------------+
```

5.2 Generalised eigenvalue problem

The second example illustrates a change of pole and a change of mode. The purpose is the computation of the 20 rightmost eigenvalues of $\mathbf{A}x = \lambda\mathbf{M}x$, where $\mathbf{M}$ is diagonal with 1 and 2 on the main diagonal and $\mathbf{A}$ is also diagonal so that the ratio of the corresponding diagonal elements of $\mathbf{A}$ and $\mathbf{M}$ form the set { –1, –2, ..., –100 }. The trust interval confirms that all the wanted eigenvalues are computed.

This example also illustrates the use of a sparse matrix package for solving the linear equations involved. It uses the HSL package MA57. This is not automatically supplied with EA16.

```
      INTEGER N, BLK, NWANT, NV, WHICH, LDV, LDBV
      PARAMETER (N=100, BLK=1, NWANT=15, NV=30, WHICH=-2)
      PARAMETER (LDV=N, LDBV=N)
      INTEGER LIW, LW
      PARAMETER (LIW=500, LW=2500)
      INTEGER IWORK(LIW)
      DOUBLE PRECISION WORK(LW), V(LDV,NV), BV(N,BLK)

      INTEGER LA
      PARAMETER (LA=2*N)
      INTEGER IRN(LA), ICN(LA)
      DOUBLE PRECISION A(LA), M(LA)
      INTEGER NZ

      INTEGER IDO, MODE, LIWORK, LWORK, NEINEG, IPOS(10), INFO(20)
      DOUBLE PRECISION SIGMA, RANGE(2)
      INTEGER I
```

```
      INTEGER ICNTL(20)
      DOUBLE PRECISION CNTL(15)

C Variables for MA57
      INTEGER ICNT57(20), INFO57(40)
      DOUBLE PRECISION CNTL57(5), RINF57(20)
      INTEGER LF, LKEEP, LRF, LIF, LW57
      PARAMETER (LF=6*N, LKEEP=5*N+LA+MAX(N,LA)+42)
      PARAMETER (LRF=1400, LIF=1300, LW57=N)
      INTEGER IW57(5*N), IF(LIF), KEEP(LKEEP)
      DOUBLE PRECISION F(LA), RF(LRF), W57(LW57)

C Store the matrices M and A in sparse format
      NZ = 0
      DO 10 I=1,N-1
         NZ = NZ + 1
         A(NZ) = 3.D0
         M(NZ) = 2.D0
         ICN(NZ) = I
         IRN(NZ) = I

         NZ = NZ + 1
         A(NZ) = -1.D0
         M(NZ) = -1.D0
         ICN(NZ) = I
         IRN(NZ) = I+1

 10   CONTINUE

      NZ = NZ + 1
      A(NZ) = 3.D0
      M(NZ) = 2.D0
      ICN(NZ) = N
      IRN(NZ) = N

C Set the default values of the control parameters for MA57.
      CALL MA57ID(CNTL57,ICNT57)

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

C By selecting ICNTL(5)=2, ICNTL(18)=1, the pole is changed at
C   every restart.
      ICNTL(5) = 2
      ICNTL(18) = 1

C Allow a change of MODE
      ICNTL(7) = 1

C Compute the amount of storage required for this routine.
      CALL EA16AD(N,BLK,NWANT,NV,LIWORK,LWORK,ICNTL,INFO)

C Check the size of the workspace.
      IF (LIWORK.GT.LIW) THEN
         WRITE (6,'(A,I4)') 'Increase LIW to at least ',LIWORK
         STOP
      ELSE IF (LWORK.GT.LW) THEN
         WRITE (6,'(A,I4)') 'Increase LW to at least ',LWORK
         STOP
      END IF
```

```
C We start with MODE=3  (standard mode).
      MODE = 3

C Factorise M using MA57.
C Store the factorization in F.
      CALL MA57AD(N, NZ, IRN, ICN, LKEEP, KEEP, IW57, ICNT57,
     &            INFO57, RINF57)
      IF (INFO57(1).LT.0) STOP

      CALL MA57BD(N, NZ, M, RF, LRF, IF, LIF, LKEEP, KEEP, IW57,
     &            ICNT57, CNTL57, INFO57, RINF57)
      IF (INFO57(1).LT.0) STOP

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

            IF (MODE.EQ.3) THEN

C Compute  V(:,IPOS(3):IPOS(4)) = M^-1 * A * V(:,IPOS(1):IPOS(2))
C We first compute  A * V(:,IPOS(1):IPOS(2)) and then
C solve a linear system with M, using the factorization in F.

               CALL MATVEC(A, N, NZ, ICN, IRN, V(1,IPOS(1)),
     &                     V(1,IPOS(3)))

C Compute  V(:,IPOS(3)) = inv(F) * V(:,IPOS(1)) using MA57
               CALL MA57CD(1, N, RF,LRF, IF, LIF, BLK, V(1,IPOS(3)),
     &                     LDV, W57, LW57, IW57, ICNT57, INFO57)
            ELSE IF (MODE.EQ.4) THEN

C Compute V(:,IPOS(3):IPOS(4)) = (A - SIGMA M)^-1 * M *
C                                V(:,IPOS(1):IPOS(2))
C with BV = M * V(:,IPOS(1):IPOS(2)

C Compute  V(:,IPOS(3)) = inv(F) * BV using MA57
               CALL DLACPY('All', N, BLK, BV,LDBV, V(1,IPOS(3)),LDV)
               CALL MA57CD(1, N, RF,LRF, IF, LIF, BLK, V(1,IPOS(3)),
     &                     LDV, W57, LW57, IW57, ICNT57, INFO57)
            END IF

         ELSE IF (IDO.EQ.2) THEN

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

            CALL MATVEC(M, N, NZ, ICN, IRN, V(1,IPOS(1)),
     &                  BV(1,IPOS(3)))

         ELSE IF (IDO.EQ.4) THEN
```

```
C Form F = A - SIGMA * M
            DO 20 I=1,NZ
               F(I) = A(I) - SIGMA * M(I)
 20         CONTINUE

C Factorize F using MA57
            CALL MA57BD(N, NZ, F, RF, LRF, IF, LIF, LKEEP, KEEP, IW57,
     &                  ICNT57, CNTL57, INFO57, RINF57)

C If failure, flag SIGMA as unusable
            IF (INFO57(1).LT.0) IDO = -4

C Set the number of negative eigenvalues
            NEINEG = INFO57(24)

         ELSE IF (IDO.EQ.5) THEN

C Change the mode from 3 into 4.
            MODE = 4

         END IF

         GO TO 1

 2    CONTINUE

C Check for errors
      IF (INFO(1).LT.0) THEN
         WRITE (6,*) 'There was an error : INFO(1) = ', INFO(1)
         STOP
      END IF

C Print some information
      WRITE (6,'(A,I5)') 'Number of iterations      : ', INFO(3)
      WRITE (6,'(A,I5)') 'Number of products with M : ', INFO(4)
      WRITE (6,'(A,I5)') 'Number of factorizations  : ', INFO(6)

C Print the computed eigenvalues
      WRITE (6, '(A/8(F9.2))') 'Eigenvalues : ',
     &      (WORK(I),I=IPOS(5),IPOS(6))
      WRITE (6,'(A,F5.2,A)') 'All eigenvalues larger than ',WORK(4),
     &               ' are computed.'

      STOP
      END

C*******************************************************
      SUBROUTINE MATVEC(MAT, N, NZ, ICN, IRN, X, Y)

      INTEGER N, NZ
      INTEGER ICN(NZ), IRN(NZ)
      DOUBLE PRECISION MAT(NZ), X(N), Y(N)

      INTEGER I
      DOUBLE PRECISION ZERO
      PARAMETER (ZERO=0.D0)

      DO 10 I=1,N
         Y(I) = ZERO
 10   CONTINUE
```

```
      DO 20 I=1,NZ
         Y(IRN(I)) = Y(IRN(I)) + MAT(I) * X(ICN(I))
         IF (IRN(I).NE.ICN(I))
     &      Y(ICN(I)) = Y(ICN(I)) + MAT(I) * X(IRN(I))
 20   CONTINUE
      RETURN
      END
```

The output produced is:

```
 Number of iterations      :    59
 Number of products with M :   253
 Number of factorizations  :     3
 Eigenvalues :
   1034.66   259.48   115.93    65.68    42.43    29.79    22.18    17.23
     13.84    11.42     9.63     8.26     7.20     6.36     5.68     5.12
 All eigenvalues larger than  5.40 are computed.
```