EB13

PACKAGE SPECIFICATION HSL

1 SUMMARY

Given a **real unsymmetric** $n \times n$ **matrix** $\mathbf{A} = \{a_{ij}\}$, this routine uses **Arnoldi based methods to calculate the** r **eigenvalues** λ_i**,** $i = 1, 2, ..., r$**, that are of largest absolute value, or are right-most, or are of largest imaginary parts.** The right-most eigenvalues are those with the most positive real part. There is an option to compute the associated eigenvectors $\mathbf{y}_i$, $i = 1, 2, ..., r$, where $\mathbf{A}\mathbf{y}_i = \lambda_i \mathbf{y}_i$. The routine may be used to compute the left-most eigenvalues of **A** by using −**A** in place of **A**.

The Arnoldi methods offered by EB13 are:

(1) The basic (iterative) Arnoldi method.

(2) Arnoldi's method with Chebychev acceleration of the starting vectors.

(3) Arnoldi's method applied to the preconditioned matrix $p_l(\mathbf{A})$, where p_l is a Chebychev polynomial.

Each method is available in blocked and unblocked form.

The methods are described in detail by Scott (1993), *An Arnoldi code for computing selected eigenvalues of sparse real unsymmetric matrices.* Rutherford Report RAL-93-097.

ATTRIBUTES — **Version:** 1.1.3. (29th March 2023) **Types:** Real (single, double). **Calls:** FA14, FD15, KB06, _DOT, _NRM2, _AXPY, _COPY, _SCAL, _GER, _GEMV, and _GEMM. **Language:** Fortran 77. **Original date:** December 1993. **Origin:** J.A. Scott, Rutherford Appleton Laboratory.

2 HOW TO USE THE PACKAGE

2.1 Overall control and argument lists

There are three entries:

(a) EB13I/ID sets default values for the control parameters. It would normally be called once prior to any calls to EB13A/AD and EB13B/BD.

(b) EB13A/AD uses the Arnoldi based method chosen by the user to calculate either the eigenvalues of largest absolute value, or the right-most eigenvalues, or the eigenvalues of largest imaginary parts of **A**.

(c) EB13B/BD uses data provided by EB13A/AD to calculate the eigenvectors corresponding to the converged eigenvalues. A second call to EB13B/BD can be used to compute the (scaled) eigenvector residuals

$$\frac{\|(\mathbf{A}\mathbf{y}_i - \lambda_i \mathbf{y}_i)\|}{\|\mathbf{A}\|}, \; 1 \le i \le r, \tag{1}$$

(or

$$\frac{\|(\mathbf{A}\mathbf{y}_i - \lambda_i \mathbf{y}_i)\|}{\|\mathbf{A}\mathbf{y}_i\|}, \; 1 \le i \le r \tag{2}$$

or

$$\|(\mathbf{A}\mathbf{y}_i - \lambda_i \mathbf{y}_i)\|, \; 1 \le i \le r). \tag{3}$$

Use of EB13B/BD is optional.

The algorithm used by EB13A/AD and EB13B/BD requires the multiplication of sets of vectors by the matrix **A**. Reverse communication is used so that the user does not have to pass the matrix **A** to EB13 but, each time a matrix-matrix multiplication **AW** is required, control is returned to the user. A simple example to illustrate the calling sequence is given in Section 5.

2.1.1 To set default values for the control parameters

The single precision version

```
      CALL EB13I(ICNTL,CNTL)
```

The double precision version

```
      CALL EB13ID(ICNTL,CNTL)
```

`ICNTL` is an `INTEGER` array of length `11` 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 `1` that need not be set by the user. On return it contains a default value (see Section 2.2.1 for details).

2.1.2 To calculate the selected eigenvalues

The single precision version

```
      CALL EB13A(IND,N,NUMEIG,NBLOCK,NSTEPS,ER,EI,LN,X,U,W,IKEEP,
     +           RKEEP,ICNTL,CNTL,INFO,RINFO)
```

The double precision version

```
      CALL EB13AD(IND,N,NUMEIG,NBLOCK,NSTEPS,ER,EI,LN,X,U,W,IKEEP,
     +            RKEEP,ICNTL,CNTL,INFO,RINFO)
```

`IND` is an `INTEGER` variable that must be set by the user. If `IND` $=$ `0` the eigenvalues of largest absolute value are computed; `IND` $=$ `1` the right-most eigenvalues are computed; if `IND` $=$ `2` the eigenvalues of largest imaginary parts are computed. This argument is not altered by the routine.

`N` is an `INTEGER` variable that must be set by the user to n, the order of the matrix **A**. This variable must be preserved by the user between calls to `EB13A/AD` and must be passed unchanged to `EB13B/BD`. This argument is not altered by the routine. **Restriction**: `N` $\geq$ `3`.

`NUMEIG` is an `INTEGER` variable that must be set by the user to r, the number of eigenvalues required. This variable must be preserved by the user between calls to `EB13A/AD` and must be passed unchanged to `EB13B/BD`. This argument is not altered by the routine. **Restriction**: `1` $\leq$ `NUMEIG` $\leq$ `N-2`.

`NBLOCK` is an `INTEGER` variable that must be set by the user to the block size for the Arnoldi method. For an unblocked method, the user should set `NBLOCK` $=$ `1`. Information concerning a suitable value for `NBLOCK` is given in Section 2.5. This variable must be preserved by the user between calls to `EB13A/AD` and must be passed unchanged to `EB13B/BD`. This argument is not altered by the routine. **Restriction**: `NBLOCK` $\geq$ `1`.

`NSTEPS` is an `INTEGER` variable that must be set by the user to the number of Arnoldi steps on each iteration. Information concerning a suitable value for `NSTEPS` is given in Section 2.5. This variable must be preserved by the user between calls to `EB13A/AD`. This argument is not altered by the routine. **Restrictions**: `NSTEPS` $\geq$ `2` and min(`N, NUMEIG+2`) $\leq$ `NSTEPS*NBLOCK` $\leq$ `N`.

`ER,EI` are `REAL` (`DOUBLE PRECISION` in the D version) arrays of length `NSTEPS*NBLOCK` that need not be set by the user. On each exit with `IKEEP(1)` $>$ `0`, the first `NUMEIG` entries of `ER` and `EI` hold, in order, approximations to the real and imaginary parts of the sought-after eigenvalues of **A**, respectively. On exit with `IKEEP(1)` $=$ `0`, the first `INFO(4)` entries of `ER` and `EI` hold, in order, the real and imaginary parts of the converged eigenvalues, respectively. Complex conjugate pairs of eigenvalues appear consecutively with the eigenvalue with positive imaginary part appearing first. These arrays must not be altered by the user between calls to `EB13A/AD` and must be passed unchanged to `EB13B/BD`.

`LN` is an `INTEGER` variable that must be set by the user to the first dimension of the arrays `X`, `U`, and `W`. This argument is not altered by the routine. **Restriction:** LN $\geq$ N.

X is a REAL (DOUBLE PRECISION in the D version) two-dimensional array with dimensions (LN, NSTEPS*NBLOCK). If ICNTL(10) = 0 (the default) or 2, X need not be set by the user. If ICNTL(10) = 1 or 3, the first NBLOCK columns of X must contain an initial estimate of the NBLOCK basis vectors which span the subspace corresponding to the NBLOCK eigenvalues of **A** that are of largest absoute value (IND = 0), the right-most eigenvalues (IND = 1), or the eigenvalues of largest imaginary parts (IND = 3). On exit with IKEEP(1) = 0, the first INFO(4) columns of X hold, in order, orthonormalized basis vectors which span the subspace corresponding to the INFO(4) converged eigenvalues. This array must not be altered by the user between calls to EB13A/AD and must be passed unchanged to EB13B/BD.

U,W are REAL (DOUBLE PRECISION in the D version) two-dimensional arrays with dimensions (LN, NSTEPS*NBLOCK). On each exit with IKEEP(1) > 0, the user must multiply columns IKEEP(2) to IKEEP(3) of W by **A**, place the results in the corresponding columns of U, and recall EB13A/AD.

IKEEP is an INTEGER array of length 11 + m, where m = NSTEPS*NBLOCK. Prior to the first call to EB13A/AD, IKEEP(1) must be set by the user to 0. On exit, IKEEP(1) > 0 indicates convergence has not yet been achieved. To continue the computation, the user must multiply columns IKEEP(2), IKEEP(2)+1,..., IKEEP(3) of the array W by **A**, place the results in the corresponding columns of U, and recall EB13A/AD. IKEEP(1) = 0 on exit indicates the calculation is complete. The only other entry of IKEEP of interest to the user is IKEEP(9). If ICNTL(10) = 2 or 3, prior to the first call to EB13A/AD, IKEEP(9) must be set by the user to hold the seed for the random number generator FA14. All other entries of IKEEP are used as workspace. This array must not be altered by the user between calls to EB13A/AD.

RKEEP is a REAL (DOUBLE PRECISION in the D version) array of length 13 + $m*(2*m$ + 10), where m = NSTEPS*NBLOCK. If ICNTL(7) = 0 (the default), RKEEP(1) must be set by the user to the norm of **A** (or an estimate of the norm of **A**) and, in this case, RKEEP(1) is not altered by the routine. If ICNTL(7) = 1 or 2, RKEEP(1) need not be set by the user. The other entries of RKEEP need not be set by the user. This array must not be altered by the user between calls to EB13A/AD. On exit with IKEEP(1) = 0, RKEEP(13), RKEEP(14),..., RKEEP(12 + $m*m$) hold the real Schur form **T** (see Section 4). The first 12 + $m*m$ entries of RKEEP must be passed unchanged to EB13B/BD.

ICNTL is an INTEGER array of length 11 that contains control parameters and must be set by the user. Default values for the components may be set by a call to EB13I/ID. 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 1 that contains a control parameter and must be set by the user. The default value for CNTL(1) may be set by a call to EB13I/ID. Details of the control parameter CNTL(1) are given in Section 2.2.1. This argument is not altered by the routine.

INFO is an INTEGER array of length 6 that need not be set by the user. It is used to hold information about the execution of the subroutine. On exit from EB13A/AD, 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.

RINFO is a REAL (DOUBLE PRECISION in the D version) array of length 5 that need not be set by the user. It is used to hold information about the execution of the subroutine. For details of the information output in RINFO, see Section 2.2.2.

2.1.3 To compute the corresponding eigenvectors.

The single precision version

```
CALL EB13B(N,NBLOCK,NSTEPS,ER,EI,LN,X,U,W,NEV,Y,RES,IKEEP,RKEEP)
```

The double precision version

```
CALL EB13BD(N,NBLOCK,NSTEPS,ER,EI,LN,X,U,W,NEV,Y,RES,IKEEP,RKEEP)
```

N,NBLOCK,NSTEPS,ER,EI,LN,X must be unchanged since the call to EB13A/AD that returned IKEEP(1) = 0. These arguments are not altered by the routine.

U,W are REAL (DOUBLE PRECISION in the D version) two-dimensional arrays with dimensions (LN, NSTEPS*NBLOCK). On exit with IKEEP(1) = 1, if the scaled eigenvector residuals are wanted, the user must multiply columns IKEEP(2) to IKEEP(3) of W by **A**, place the results in the corresponding columns of U, and recall EB13B/BD.

NEV is an INTEGER variable that must be set by the user to the number of eigenvalues successfully computed by EB13A/AD. NEV should be equal to the value of INFO(4) returned by EB13A/AD. This argument is not altered by the routine.

Y is a REAL (DOUBLE PRECISION in the D version) two-dimensional array with dimensions (LN, NEV) that need not be set by the user. On each exit, if the I th computed eigenvalue is real (EI(I) = 0), the I th column of Y contains the corresponding eigenvector; if the I th computed eigenvalue is complex with positive imaginary part (EI(I) > 0), the I th and (I+1) th columns of Y contain, respectively, the real and imaginary parts of the corresponding eigenvector. This array must not be altered by the user between calls to EB13B/BD.

RES is a REAL (DOUBLE PRECISION in the D version) array of length NEV that need not be set by the user. If ICNTL(7) ≠ 2 (respectively, ICNTL(7) = 2) on entry to EB13A/AD, on exit from EB13B/BD with IKEEP(1) = 0, RES(I) contains the scaled eigenvector residual (1) (respectively, (2)) for the Ith computed eigenvector.

IKEEP is an INTEGER array of length at least 3. Prior to the first call to EB13B/BD, IKEEP(1) must be set by the user to 0. On exit from the first call to EB13B/BD, IKEEP(1) = 1 is returned, which indicates that the eigenvectors have been computed. To compute the (scaled) eigenvector residuals, the user must multiply columns IKEEP(2), IKEEP(2)+1,..., IKEEP(3) of the array W by **A**, place the results in the corresponding columns of U, and recall EB13B/BD. IKEEP(1) = 0 on exit indicates the calculation is complete. If ICNTL(7) = 2 on entry to EB13A/AD, then on exit from EB13B/BD with IKEEP(1) = 0, IKEEP(4) holds the number of computed eigenvectors $\mathbf{y}_i$ for which $\|\mathbf{A}\mathbf{y}_i\|_2$ was found to less than u, the machine precision.

RKEEP is a REAL (DOUBLE PRECISION in the D version) array of length at least $12 + m*m$, where $m =$ NSTEPS*NBLOCK. On the initial call to EB13B/BD, the first $12 + m*m$ entries of RKEEP must be unchanged since the call to EB13A/AD that returned IKEEP(1) = 0. The contents of this array are destroyed by the routine.

2.2 Arrays for control and information

2.2.1 Control parameters

The elements of the arrays ICNTL and CNTL control the action of EB13A/AD. Default values may be set by calling EB13I/ID.

ICNTL(1) is the stream number for error messages and has the default value 6. Printing of error messages is suppressed if ICNTL(1) ≤ 0.

ICNTL(2) is the stream number for warnings and diagnostic messages and has the default value 6. Printing of such messages is suppressed if ICNTL(2) ≤ 0.

ICNTL(3) is the maximum degree of the Chebychev iteration polynomial used by EB13A/AD and has the default value 80. ICNTL(3) is not accessed by EB13A/AD if ICNTL(9) = 1.

ICNTL(4) controls the degree of the Chebychev iteration polynomial and has default value 0. If ICNTL(4) = 0, at each iteration the degree of the Chebychev polynomial is determined by the routine EB13A/AD. If ICNTL(4) > 0, the degree of the Chebychev polynomial is taken to be ICNTL(4). ICNTL(4) is not accessed by EB13A/AD if ICNTL(9) = 1.

ICNTL(5) determines the maximum number of matrix-vector multiplications allowed by EB13A/AD. The maximum number allowed is ICNTL(5)*NUMEIG. ICNTL(5) has the default value ICNTL(5) = $2*10^4$. A matrix-matrix multiplication **AW** where **W** is an N × K matrix is taken to be K matrix-vector multiplications.

ICNTL(6) is used to control the printing of error, warning, and diagnostic messages in EB13A/AD. It has default value 2. Possible values are:

1 Only error messages are output.

2 Error and warning messages output.

3 As for 2, plus scalar parameters and the control parameters on the first entry to EB13A/AD, and information arrays INFO and RINFO on the exit which returns IKEEP(1) = 0.

4 As for 3, plus the converged INFO(4) eigenvalues on the exit which returns IKEEP(1) = 0.

5 As for 4, plus the information arrays INFO and RINFO and the residual for the first unconverged eigenvalue on each iteration.

6 As for 5, plus the computed approximations to the NUMEIG sought-after eigenvalues on each iteration.

ICNTL(7) is used to indicate whether the stopping criteria for the i th eigenvalue is to use $||\mathbf{A}||$ or $||(\mathbf{AX})_i||$ (see Section 4). It has default value 0 and, in this case, $||\mathbf{A}||$ will be used and the user must set RKEEP(1) equal to $||\mathbf{A}||$ (or an estimate of $||\mathbf{A}||$) prior to the first call to EB13A/AD. If the user wants to use $||\mathbf{A}||$ in the stopping criteria but $||\mathbf{A}||$ is unavailable, the user should set ICNTL(7) = 1 and the Frobenius norm of **A** will be computed by EB13A/AD. This will involve n matrix-vector multiplications. Note that, if the user is able to multiply vectors by both **A** and its transpose $\mathbf{A}^T$, an estimate of the 1-norm of **A** can be computed using at most 10 matrix-vector multiplications by calling the Harwell Subroutine Library routine MC41. If the user wants to use $||(\mathbf{AX})_i||$ in the stopping criteria, the user should set ICNTL(7) = 2. If the user wants the stopping criteria to be based solely on the norm of the residual (without involving $||\mathbf{A}||$ or $||(\mathbf{AX})_i||$), the user should set ICNTL(7) = 3. If ICNTL(7) = 2 or 3, $||\mathbf{A}||$ is not computed.

ICNTL(8) controls which algorithm is used to compute the Chebychev ellipse (see Section 4 for details). It has default value 1. Possible values are:

1 Algorithm of Ho (see Ho 1990).

2 Algorithm of Braconnier (see Braconnier 1993).

3 Algorithm of Saad (see Saad 1984).

(References are given in Section 4). ICNTL(8) is not accessed by EB13A/AD if ICNTL(9) = 1.

ICNTL(9) controls which variant of Arnoldi's method is used to compute the eigenvalues if IND = 1 or 2 on the first call to EB13A/AD. It has default value 2. Possible values are:

1 Arnoldi (no Chebychev acceleration). If IND = 1 this method should only be used if the right-most eigenvalues are also those of largest absolute value and if IND = 2 it should only be used if the eigenvalues with largest imaginary parts are those of largest absolute value. If IND = 0 on the first call to EB13A/AD, this method is used.

2 Arnoldi with Chebychev acceleration of the starting vectors.

3 Arnoldi's method applied to the preconditioned matrix $p_l(\mathbf{A})$, where p_l is a Chebychev polynomial.

These algorithms are described in detail by Scott (1993).

`ICNTL(10)` controls whether the user has supplied an initial estimate of the basis vectors corresponding to the `NBLOCK` eigenvalues of **A** that are of largest absoute value (`IND` = `0`), the right-most eigenvalues (`IND` = `1`), or the eigenvalues of largest imaginary parts (`IND` = `3`). `ICNTL(10)` also controls whether or not the user wishes to supply the initial seed value for the random number generator `FA14` which is set on the first call, i.e. when `IKEEP(1)` = 0. The possible values are:

- `0` (the default), the user supplies no estimate of the basis vectors and the initial seed value is set by `EA13A/AD`.
- `1` the user supplies the estimate of the basis vectors but no seed value.
- `2` the user supplies the seed value but no basis vectors.
- `3` the user supplies both the basis vectors and the seed value.

`ICNTL(11)` determines the maximum number of Arnoldi iterations allowed by `EB13A/AD`. The maximum number allowed is `ICNTL(11)*NUMEIG`. `ICNTL(11)` has the default value `ICNTL(11)` = `500`.

`CNTL(1)` is the convergence tolerance (see Section 4). It has default value $u*10^3$, where u is the machine precision.

2.2.2 Information arrays

The arrays `INFO` and `RINFO` are used to provide the user with information on the execution of `EB13A/AD`.

`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)` holds the total number of matrix-vector products formed.

`INFO(3)` holds the current degree of the Chebychev iteration polynomial.

`INFO(4)` holds the number of eigenvalues which have converged. In general, `INFO(4)` is less than or equal to `NUMEIG` but it may be equal to `NUMEIG+1` if the `NUMEIG`th computed eigenvalue is the first eigenvalue in a complex conjugate pair.

`INFO(5)` holds the number of Arnoldi iterations which have been performed.

`INFO(6)` holds the highest degree of the Chebychev iteration polynomial used by the algorithm.

`RINFO(1),RINFO(2),RINFO(3)` hold the current ellipse parameters. The ellipse has centre `RINFO(1)`, foci `RINFO(1)+C`, `RINFO(1)-C`, and crosses the real axis at `RINFO(1)+A`, `RINFO(1)-A`, where `C` = $\sqrt{\texttt{RINFO(2)}}$ and `A` = $\sqrt{\texttt{RINFO(3)}}$. `RINFO(1)`, `RINFO(2)`, and `RINFO(3)` are set to zero if `ICNTL(9)` = `1` or `IND` = `0`.

`RINFO(4)` holds the value of the convergence parameter satisfied by the `INFO(4)` computed eigenvalues.

`RINFO(5)` holds the norm of the matrix **A**. If `ICNTL(7)` = `0` (the default), `RINFO(5)` holds the norm supplied by the user, and if `ICNTL(7)` = `1`, `RINFO(5)` holds the Frobenius norm of **A** computed by `EB13A/AD`. If `ICNTL(7)` = `2`, the norm of **A** is not computed and `RINFO(5)` is set to zero. If `ICNTL(7)` = `3`, `RINFO(5)` is set to one.

2.3 Error diagnostics

If `EB13A/AD` returns with a negative value of `INFO(1)`, an error has occurred; if `EB13A/AD` returns with a positive value of `INFO(1)`, a warning has been issued. There are no error or warning returns from `EB13B/BD`. Error messages are output on unit `ICNTL(1)` and warnings on unit `ICNTL(2)`. Possible non-zero values of `INFO(1)` are given below.

–1 Value of `N` out of range. `N` < `3`. Immediate return with input parameters unchanged.

–2 Value of `NBLOCK` out of range. `NBLOCK` < `1`. Immediate return with input parameters unchanged.

–3 Value of `NUMEIG` out of range. Either `NUMEIG` > `N-2` or `NUMEIG` < `1`. Immediate return with input parameters unchanged.

–4 Value of `NSTEPS` out of range. Either `NSTEPS` < `2`, or `NSTEPS*NBLOCK` > `N`, or

NSTEPS*NBLOCK < min(N, NUMEIG+2). Immediate return with input parameters unchanged.

–5 Value of LN out of range. LN < N. Immediate return with input parameters unchanged.

–6 Norm of **A** is less than $u*10^2$, where u is the machine precision. Immediate return.

–7 Algorithm required more matrix-vector multiplications than specified by the control parameter ICNTL(5). The user may increase ICNTL(5) and restart the computation; full details are given in Section 2.4.

–8 The algorithm was unable to find all the NUMEIG wanted eigenvalues with the user-supplied parameters because, at some stage, there were no points on which to construct the Chebychev ellipse (see Section 4). This could happen if, for example, the matrix **A** has multiple eigenvalues. The code will return with the number of successfully computed eigenvalues in INFO(4) (see Section 2.2.2). To compute all NUMEIG requested eigenvalues the user is advised to try increasing NSTEPS and restarting the computation (see Section 2.4 for details). This error cannot be returned if ICNTL(9) = 1.

–9 Algorithm required more iterations than specified by the control parameter ICNTL(11). The user may increase ICNTL(11) and restart the computation; full details are given in Section 2.4.

+1 The user-supplied convergence tolerance CNTL(1) lies outside the interval (u,1.0), where u is the machine precision. The default convergence tolerance $u*10^3$ is used.

+2 The convergence tolerance CNTL(1) (or the default accuracy if CNTL(1) was supplied out of range) was not achieved. The tolerance achieved by the INFO(4) accepted eigenvalues is returned in RINFO(4). INFO(1) = +2 will overwrite INFO(1) = +1.

+3 The Arnoldi iteration stopped in fewer than NSTEPS. Some spurious zero (or almost zero) eigenvalues may be returned. The user may try choosing a smaller value for NSTEPS. INFO(1) = +3 will overwrite INFO(1) = +1 or +2.

2.4 Restarting the computation

In certain circumstances, the user may wish to restart the computation after an error has been flagged. If INFO(1) = –7 on exit from EB13A/AD, the algorithm has failed to converge in the number of matrix-vector multiplications specified by control parameter ICNTL(5). To restart the computation from the point at which INFO(1) = –7 was returned, the user should increase ICNTL(5), set INFO(1) = 0, multiply columns IKEEP(2) to IKEEP(3) of W by **A**, place the results in the corresponding columns of U, and recall EB13A/AD without making any other changes to the input parameters. However, it may be more efficient to start the computation again with either a different choice for the control parameter ICNTL(9) (ICNTL(9) controls which variant of Arnoldi's method is used) or an increased value for NSTEPS and/or a different choice for NBLOCK.

If INFO(1) = –8 on exit from EB13A/AD, the user may restart the computation taking advantage of the basis vectors already found by increasing NSTEPS, setting INFO(1) = 0, and recalling EB13A/AD without making any other changes to the input parameters (but note that the dimension of some of the arrays is dependent on NSTEPS).

If INFO(1) = –9 on exit from EB13A/AD, the algorithm has failed to converge in the number of Arnoldi iterations specified by control parameter ICNTL(11). Again, the computation can be restarted from the point at which this error was returned by increasing ICNTL(11), setting INFO(1) = 0, and recalling EB13A/AD without making any other changes to the input parameters.

2.5 Choosing values for NBLOCK and NSTEPS

The best choice for NBLOCK and NSTEPS is problem dependent but when choosing these parameters we advise the user to consider the following points.

(1) NBLOCK should be chosen to be at least as large as the largest cluster of wanted eigenvalues.

(2) The storage required by EB13 is proportional to $(\text{NBLOCK}*\text{NSTEPS})^2$ and, at each iteration, the cost of the Arnoldi steps is proportional to $(\text{NBLOCK}*\text{NSTEPS})^2*\text{N}$, and the cost of computing the eigenvalues of the

Hessenberg matrix (see Section 4) is proportional to $(\texttt{NBLOCK*NSTEPS})^3$.

In general, we would recommend the user to choose `NBLOCK` and `NSTEPS` so that `NSTEPS*NBLOCK` lies in the range `3*NUMEIG` to `10*NUMEIG`. For further advice regarding the choice of `NBLOCK` and `NSTEPS`, as well as numerical results which illustrate the effects of changing the values of these parameters, the user is referred to Scott (1993).

2.6 Shift-and-invert strategies and the generalized eigenproblem

Subroutine `EB13A/AD` can be used with $\mathbf{A}$ replaced by the shifted and inverted matrix $(\mathbf{A}-p\,\mathbf{I})^{-1}$. The user should set `ICNTL(9)` $=1$ and on each return to the user with `IKEEP(1)` >0, the matrix-matrix multiplication $\mathbf{U}=(\mathbf{A}-p\,\mathbf{I})^{-1}\mathbf{W}$ is required. This is equivalent to solving the system

$$(\mathbf{A}-p\,\mathbf{I})\,\mathbf{U}=\mathbf{W}. \tag{4}$$

If $\mathbf{A}$ is large the solution of the system (4) may itself be quite time-consuming but note that, if a direct method of solution is used, the decomposition of $\mathbf{A}-p\,\mathbf{I}$ into triangular factors needs only to be done once for each choice of the shift p, the only operations in the inner loop being the relatively cheap forward and backward substitutions.

`EB13A/AD` may also be used for the solution of generalized eigenvalue problems of the form

$$\mathbf{Ax}=\lambda\mathbf{Bx}, \tag{5}$$

where $\mathbf{A}$ and $\mathbf{B}$ are real, sparse, unsymmetric matrices. If $\mathbf{B}$ is nonsingular, (5) can be treated as a standard unsymmetric eigenvalue problem by working with the matrix $\mathbf{B}^{-1}\mathbf{A}$. Again, when control is returned to the user, a linear system of equations must be solved. For this problem, the LU factorisation of $\mathbf{B}$ need only be formed once at the start of the computation. If $\mathbf{B}$ is singular, a shift γ may be introduced so that $(\mathbf{A}-\gamma\mathbf{B})$ is nonsingular and then

$$(\mathbf{A}-\gamma\mathbf{Bx})=(\lambda-\gamma)\mathbf{Bx}$$

may be treated as a standard eigenvalue problem by working with the matrix

$$(\mathbf{A}-\gamma\mathbf{B})^{-1}\mathbf{B}.$$

For further details see Scott (1993) and the references therein.

2.7 Underflows

The nature of the calculations performed in this subroutine means that underflows are likely to occur. It is quite safe to set numbers that underflow to zero, and action by the user may be required to ensure that this is done efficiently by the computing system in use.

3 GENERAL INFORMATION

Use of common: None.

Other routines called directly: `EB13A/AD` calls the following internal subroutines: `EB13C/CD`, `EB13D/DD`, `EB13E/ED`, `EB13F/FD`, `EB13G/GD`, `EB13H/HD`, `EB13J/JD`, `EB13K/KD`, `EB13L/LD`, `EB13M/MD`, `EB13N/ND`, `EB13O/OD`, `EB14A/AD`, `EB14B/BD`, `EB14C/CD`, and `EB14D/DD`. `EB13A/AD` also calls routines `FA14A/AD`, `FD15A/AD`, and `KB06A/AD` from the Harwell Subroutine Library, and the BLAS kernels `SNRM2/DNRM2`, `SCOPY/DCOPY`, `SAXPY/DAXPY`, `SSCAL/DSCAL`, `SGER/DGER`, `SGEMV/DGEMV`, and `SGEMM/DGEMM`.
`EB13B/BD` calls the internal subroutines `EB13P/PD`, `EB13Q/QD`, and `EB13R/RD`, and the BLAS kernels `SDOT/DDOT`, `SNRM2/DNRM2`, `SSCAL/DSCAL`, and `SGEMM/DGEMM`.

Input/output: Error messages are printed on unit `ICNTL(1)` and warnings on unit `ICNTL(2)`; see Section 2.3.

Restrictions:

N ≥ 3,
1 ≤ NUMEIG ≤ N-2,
NBLOCK ≥ 1.

NSTEPS ≥ 2 and min(N, NUMEIG+2) $\leq$ NSTEPS*NBLOCK $\leq$ N.

4 METHOD

EB13A/AD

The algorithm used by EB13A/AD is a variant of Arnoldi's method selected by the user (using the control parameter ICNTL(9)). In each case, the code first checks the input data. If a fatal error is found, INFO(1) is given a negative value and control is returned to the user. If no errors are encountered and the user has not supplied an estimate of the norm of **A** but has set ICNTL(7) = 1, the Frobenius norm $\|\mathbf{A}\|_F = (\sum_i \sum_j a_{ij}^2)^{\frac{1}{2}}$ is computed. To do this, control is returned to the user n times for the necessary matrix-vector products to be formed. The results are accumulated by EB13A/AD and the norm of **A** is held in RINFO(5). EB13A/AD then proceeds as follows:

1. *Initialization:* If ICNTL(10) $\neq$ 1 or 3, generate NBLOCK random vectors $\mathbf{x}_1, ..., \mathbf{x}_{\text{NBLOCK}}$ using the Harwell Subroutine Library routine FA14A/AD. Set $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{\text{NBLOCK}}]$. Orthonormalise the columns of **X** using the modified Gram-Schmidt algorithm and let $\mathbf{X}_1$ be the resulting set of orthonormal columns. Set $l = 1$, $p_l(\lambda) = 1$.
2. *Chebychev acceleration:* ICNTL(9) = 2 only (Arnoldi with Chebychev acceleration of starting vectors). Compute $\mathbf{X} \Leftarrow p_l(\mathbf{A})\mathbf{X}_1$. Orthonormalise the columns of **X** and let $\mathbf{X}_1$ be the resulting set of orthonormal columns.
3. *Arnoldi steps (with preconditioning) :*
```
If ICNTL(9) = 1 or 2, set $\phi(\mathbf{A}) = \mathbf{A}$.
If ICNTL(9) = 3, set $\phi(\mathbf{A}) = p_l(\mathbf{A})$ (Chebychev preconditioned Arnoldi).
For $j = 1, 2, ...,$ NSTEPS do
    (i) $\mathbf{W}_j = \phi(\mathbf{A})\mathbf{X}_j$
    (ii) $\mathbf{H}_{ij} = \mathbf{X}_i^T \mathbf{W}_j$, $i = 1, 2, ..., j$.
    (iii) If $j <$ NSTEPS, $\mathbf{S}_j = \mathbf{W}_j - \sum_{i=1}^{j} \mathbf{X}_i \mathbf{H}_{ij}$
    (iv) If $j <$ NSTEPS, $\mathbf{Q}_j \mathbf{R}_j = \mathbf{S}_j$ (QR factorization of $\mathbf{S}_j$), $\mathbf{H}_{j+1,j} = \mathbf{R}_j$, $\mathbf{X}_{j+1} = \mathbf{Q}_j$.
end do
```
   Set $\mathbf{X} = [\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_{\text{NSTEPS}}]$. (EB13A/AD incorporates iterative refinement into this Gram-Schmidt step).
4. *Hessenberg form*
   (a) If ICNTL(9) = 1 or 2 and NBLOCK > 1, reduce the block Hessenberg matrix $\mathbf{H}_B = \{\mathbf{H}_{ij}\}$ to upper Hessenberg form $\mathbf{H} = \mathbf{V}^T \mathbf{H}_B \mathbf{V}$.
   (b) If ICNTL(9) = 1 or 2 and NBLOCK = 1, set **V** to be the identity matrix.
   (c) If ICNTL(9) = 3, compute $\mathbf{B} = \mathbf{X}^T \mathbf{A} \mathbf{X}$ and reduce **B** to upper Hessenberg form $\mathbf{H} = \mathbf{V}^T \mathbf{B} \mathbf{V}$.
5. *Eigenvalue computation:* Reduce the upper Hessenberg matrix **H** to real Schur form $\mathbf{T} = \mathbf{Z}^T \mathbf{H} \mathbf{Z}$, where **T** is a block triangular matrix and each diagonal block $\mathbf{T}_{ii}$ is either of order 1 or is a 2×2 matrix having complex conjugate eigenvalues, with the eigenvalues ordered along the diagonal blocks. Set $\mathbf{X} \Leftarrow \mathbf{XVZ}$.
6. *Convergence test:* If the first NUMEIG columns of **X** satisfy the convergence criteria
$$\|(\mathbf{AX} - \mathbf{XT})_i\|_2 \leq \text{CNTL(1)} * \|\mathbf{A}\|, \quad 1 \leq i \leq \text{NUMEIG}, \tag{6}$$
   (or, if ICNTL(7) = 2,
$$\|(\mathbf{AX} - \mathbf{XT})_i\|_2 \leq \text{CNTL(1)} * \|(\mathbf{AX})_i\|, \quad 1 \leq i \leq \text{NUMEIG}, \tag{7}$$
   or, if ICNTL(7) = 3,
$$\|(\mathbf{AX} - \mathbf{XT})_i\|_2 \leq \text{CNTL(1)}, \quad 1 \leq i \leq \text{NUMEIG}), \tag{8}$$
   then **stop** else

If `ICNTL(9)` = `2` or `3`, find the ellipse parameters for the Chebychev ellipse containing the unwanted eigenvalues using the algorithm determined by `ICNTL(8)`, compute the degree l of the Chebychev polynomial for the next iteration, and define the Chebychev polynomial $p_l(\lambda)$. Choose the restart vector.
If `ICNTL(9)` = `2` **go to 2**.
If `ICNTL(9)` = `1` or `3` **go to 3**.

`EB13A/AD` incorporates criteria that terminate the computation if convergence is unacceptably slow. In addition, implicit deflation techniques are employed. For full details, the user is referred to Scott (1993).

The stopping criteria (6) is based on the backward error. However, it requires $\|\mathbf{A}\|$ and can lead to misleading results concerning the accuracy of the computed eigenvalues if $\|\mathbf{A}\|$ is large and the eigenbasis is not ill-conditioned. Because of these potential difficulties, `EB13` offers the user the option of using (7) or (8).

EB13B/BD

Once `EB13A/AD` has terminated successfully (`IKEEP(1)` = `0` has been returned), `EB13B/BD` may be called to compute the eigenvectors corresponding to the converged eigenvalues. `EB13B/BD` computes the eigenvectors $\mathbf{V}$ of the real Schur form $\mathbf{T}=\mathbf{X}^T\mathbf{AX}$ using back substitution and then takes $\mathbf{y}_i=(\mathbf{AV})_i$ to be the approximate eigenvector of $\mathbf{A}$ corresponding to λ_i. If the (scaled) eigenvector residuals given by (1) (or (2) if `ICNTL(7)` = `2` or (3) if `ICNTL(7)` = `3`) are required, the user must compute $\mathbf{AY}$, where $\mathbf{Y}$ has columns $\mathbf{y}_1,...,\mathbf{y}_{\texttt{INFO(4)}}$ and recall `EB13B/BD`.

References

Braconnier, (1993). The Arnoldi-Tchebycheff algorithm for solving large nonsymmetric eigenproblems. Tech. Report TR/PA/93/25, CERFACS, Toulouse.

Ho, D. (1990). Tchebychev acceleration technique for large scale nonsymmetric matrices. *Numerische Math.* **56**, 721-734.

Scott, J. A. (1993). An Arnoldi code for computing selected eigenvalues of sparse real unsymmetric matrices. Rutherford Appleton Laboratory Report RAL-93-097.

Saad, Y. (1984). Chebychev acceleration techniques for solving nonsymmetric eigenvalue problems. *Math. Comp.* **42**, 567-588.

5 EXAMPLE OF USE

The following program illustrates the use of `EB13A/AD` and `EB13B/BD`. We wish to calculate the right-most eigenvalue and corresponding eigenvector of the following matrix

$$\mathbf{A}=\begin{pmatrix}1&0&0&0&2\\3&2&0&0&0\\0&0&0&1&4\\0&0&3&1&0\\0&0&0&0&5\end{pmatrix}.$$

```
C  Code to illustrate the use of EB13AD and EB13BD
C
C     .. Parameters ..
      INTEGER LN,N,NBLOCK,NSTEPS,NUMEIG,NZ
      PARAMETER (LN=5,N=5,NBLOCK=1,NSTEPS=3,NUMEIG=1,NZ=10)
C     ..
C     .. Local Scalars ..
      DOUBLE PRECISION ANORM
      INTEGER I,IND,NEV
C     ..
```

```
C     .. Local Arrays ..
      DOUBLE PRECISION A(NZ),EI(NSTEPS*NBLOCK),ER(NSTEPS*NBLOCK),
     +                 KEEP(13+ (10+2*NSTEPS*NBLOCK)*NSTEPS*NBLOCK),
     +                 RES(N),RINFO(5),U(LN,NSTEPS*NBLOCK),
     +                 W(LN,NSTEPS*NBLOCK),X(LN,NSTEPS*NBLOCK),
     +                 Y(LN,NUMEIG+1),CNTL(1)
      INTEGER ICNTL(11),IKEEP(11+NSTEPS*NBLOCK),INFO(6),IRN(NZ),JCN(NZ)
C     ..
C     .. External Subroutines ..
      EXTERNAL EB13AD,EB13BD,EB13ID,MXMUL
C     ..
C     .. Data statements ..
      DATA IRN/1,2,2,3,4,3,4,1,3,5/
      DATA JCN/1,1,2,3,3,4,4,5,5,5/
      DATA A/6.0D0,3.0D0,7.0D0,5.0D0,3.0D0,
     +       1.0D0,6.0D0,2.0D0,4.0D0,10.0D0/
      DATA ANORM/10.0D0/
C     ..
C Call initialisation routine
      CALL EB13ID(ICNTL,CNTL)
C Reset ICNTL(6) for printing
      ICNTL(6) = 6

C Prepare to call EB13A/AD
      IKEEP(1) = 0
      KEEP(1) = ANORM
      IND = 1

   10 CALL EB13AD(IND,N,NUMEIG,NBLOCK,NSTEPS,ER,EI,LN,X,U,W,IKEEP,
     +            KEEP,ICNTL,CNTL,INFO,RINFO)
      IF (INFO(1).LT.0) THEN
         WRITE (6,FMT=*) ' EB13A/AD failed'
         GO TO 30
      ELSE IF (IKEEP(1).EQ.0) THEN
         WRITE (6,FMT=*) '  Algorithm terminated successfully.'
         GO TO 20
      ELSE
C Form matrix-vector products and recall EB13A/AD
         CALL MXMUL(N,IKEEP(2),IKEEP(3),NZ,IRN,JCN,A,U,W,LN)
         GO TO 10
      END IF

C Compute corresponding eigenvector and residual

   20 NEV = INFO(4)
      CALL EB13BD(N,NBLOCK,NSTEPS,ER,EI,LN,X,U,W,NEV,Y,RES,IKEEP,KEEP)
      CALL MXMUL(N,IKEEP(2),IKEEP(3),NZ,IRN,JCN,A,U,W,LN)
      CALL EB13BD(N,NBLOCK,NSTEPS,ER,EI,LN,X,U,W,NEV,Y,RES,IKEEP,KEEP)
      WRITE (6,FMT=9000) (Y(I,1),I=1,N)
      IF (RES(1).LT.1E-15) THEN
         WRITE (6,FMT=*) '  Residual is near machine precision.'
      ELSE
         WRITE (6,FMT=*) '  Residual greater than machine precision.'
      END IF

   30 STOP
 9000 FORMAT (/'  Computed eigenvector is:',/5G12.4)
      END

      SUBROUTINE MXMUL(N,NLOW,NUP,NZ,IRN,JCN,A,U,W,LN)
```

```
C     .. Scalar Arguments ..
      INTEGER LN,N,NLOW,NUP,NZ
C     ..
C     .. Array Arguments ..
      DOUBLE PRECISION A(NZ),U(LN,NUP),W(LN,NUP)
      INTEGER IRN(NZ),JCN(NZ)
C     ..
C     .. Local Scalars ..
      INTEGER I,ICOL,J,L
C     ..
      DO 30 ICOL = NLOW,NUP
         DO 10 I = 1,N
            U(I,ICOL) = 0.0D0
   10    CONTINUE
         DO 20 L = 1,NZ
            I = IRN(L)
            J = JCN(L)
            U(I,ICOL) = U(I,ICOL) + A(L)*W(J,ICOL)
   20    CONTINUE
   30 CONTINUE
      RETURN
      END
```

This produces the following output:

```
Entering EB13A/AD with
IND = 1 N =     5  NUMEIG =    1  NBLOCK =    1  NSTEPS =    3  LN =      5

ICNTL (1:11) =  6  6  80   0  20000   6   0   1   2   0  500
CNTL =   2.2204D-13

On iteration   1:
INFO (1:6) =       0      5      0      0      1      0
RINFO(1:5) =   0.0000D+00  0.0000D+00  0.0000D+00  2.2204D-13  1.0000D+01
Residual of first unconverged eigenvalue is   1.7269D-01
Computed eigenvalues:
 Real part  Imaginary part   Real part  Imaginary part
 6.7968D+00  7.6617D-01      6.7968D+00 -7.6617D-01

On successful exit:
INFO (1:6) =       0     48     40      1      2     40
RINFO(1:5) =   4.0928D+00  0.0000D+00  1.0000D-08  2.2204D-13  1.0000D+01
Computed eigenvalues:
 Real part  Imaginary part   Real part  Imaginary part
 1.0000D+01  0.0000D+00
  Algorithm terminated successfully.

 Computed eigenvector is:
 0.2944      0.2944      0.5542      0.4157      0.5888
  Residual is near machine precision.
```