

**NAME**

OTHERauxiliary

**SYNOPSIS****Modules****double****real****complex****complex16****Functions**subroutine **dcombssq** (V1, V2)**DCOMBSSQ** adds two scaled sum of squares quantities.logical function **disnan** (DIN)**DISNAN** tests input for NaN.subroutine **dlabad** (SMALL, LARGE)**DLABAD**subroutine **dlacpy** (UPLO, M, N, A, LDA, B, LDB)**DLACPY** copies all or part of one two-dimensional array to another.subroutine **dlae2** (A, B, C, RT1, RT2)**DLAE2** computes the eigenvalues of a 2-by-2 symmetric matrix.subroutine **dlaezb** (IJOB, NITMAX, N, MMAX, MINP, NBMIN, ABSTOL, RELTOL, PIVMIN, D, E, E2, NVAL, AB, C, MOUT, NAB, WORK, IWORK, INFO)**DLAEBZ** computes the number of eigenvalues of a real symmetric tridiagonal matrix which are less than or equal to a given value, and performs other tasks required by the routine sstebz.subroutine **dlaev2** (A, B, C, RT1, RT2, CS1, SN1)**DLAEV2** computes the eigenvalues and eigenvectors of a 2-by-2 symmetric/Hermitian matrix.subroutine **dlagts** (JOB, N, A, B, C, D, IN, Y, TOL, INFO)**DLAGTS** solves the system of equations  $(T-\lambda I)x = y$  or  $(T-\lambda I)Tx = y$ , where  $T$  is a general tridiagonal matrix and  $\lambda$  a scalar, using the LU factorization computed by slagtf.logical function **dlaisnan** (DIN1, DIN2)**DLAISNAN** tests input for NaN by comparing two arguments for inequality.integer function **dlaneg** (N, D, LLD, SIGMA, PIVMIN, R)**DLANEG** computes the Sturm count.double precision function **dlanst** (NORM, N, D, E)**DLANST** returns the value of the 1-norm, or the Frobenius norm, or the infinity norm, or the element of largest absolute value of a real symmetric tridiagonal matrix.double precision function **dlapy2** (X, Y)**DLAPY2** returns  $\sqrt{x^2+y^2}$ .double precision function **dlapy3** (X, Y, Z)**DLAPY3** returns  $\sqrt{x^2+y^2+z^2}$ .subroutine **dlarnv** (IDIST, ISEED, N, X)**DLARNV** returns a vector of random numbers from a uniform or normal distribution.subroutine **dlarra** (N, D, E, E2, SPLTOL, TNRM, NSPLIT, ISPLIT, INFO)**DLARRA** computes the splitting points with the specified threshold.subroutine **dlarrb** (N, D, LLD, IFIRST, ILAST, RTOL1, RTOL2, OFFSET, W, WGAP, WERR, WORK, IWORK, PIVMIN, SPDIA, TWIST, INFO)**DLARRB** provides limited bisection to locate eigenvalues for more accuracy.subroutine **dlarrc** (JOB, N, VL, VU, D, E, PIVMIN, EIGCNT, LCNT, RCNT, INFO)**DLARRC** computes the number of eigenvalues of the symmetric tridiagonal matrix.subroutine **dlarrd** (RANGE, ORDER, N, VL, VU, IL, IU, GERS, RELTOL, D, E, E2, PIVMIN, NSPLIT, ISPLIT, M, W, WERR, WL, WU, IBLOCK, INDEXW, WORK, IWORK, INFO)**DLARRD** computes the eigenvalues of a symmetric tridiagonal matrix to suitable accuracy.subroutine **dlarre** (RANGE, N, VL, VU, IL, IU, D, E, E2, RTOL1, RTOL2, SPLTOL, NSPLIT, ISPLIT, M, W, WERR, WGAP, IBLOCK, INDEXW, GERS, PIVMIN, WORK, IWORK, INFO)**DLARRE** given the tridiagonal matrix  $T$ , sets small off-diagonal elements to zero and for each unreduced block  $T_i$ , finds base representations and eigenvalues.subroutine **dlarrf** (N, D, L, LD, CLSTRT, CLEND, W, WGAP, WERR, SPDIA, CLGAPL, CLGAPR, PIVMIN, SIGMA, DPLUS, LPLUS, WORK, INFO)

**DLARRF** finds a new relatively robust representation such that at least one of the eigenvalues is relatively isolated.

subroutine **dlarrj** (N, D, E2, IFIRST, ILAST, RTOL, OFFSET, W, WERR, WORK, IWORK, PIVMIN, SPDIAM, INFO)

**DLARRJ** performs refinement of the initial estimates of the eigenvalues of the matrix T.

subroutine **dlarrk** (N, IW, GL, GU, D, E2, PIVMIN, RELTOL, W, WERR, INFO)

**DLARRK** computes one eigenvalue of a symmetric tridiagonal matrix T to suitable accuracy.

subroutine **dlarrv** (N, D, E, INFO)

**DLARRR** performs tests to decide whether the symmetric tridiagonal matrix T warrants expensive computations which guarantee high relative accuracy in the eigenvalues.

subroutine **dlartg** (F, G, CS, SN, R)

**DLARTG** generates a plane rotation with real cosine and real sine.

subroutine **dlartgp** (F, G, CS, SN, R)

**DLARTGP** generates a plane rotation so that the diagonal is nonnegative.

subroutine **dlaruv** (ISEED, N, X)

**DLARUV** returns a vector of n random real numbers from a uniform distribution.

subroutine **dlas2** (F, G, H, SSMIN, SSMAX)

**DLAS2** computes singular values of a 2-by-2 triangular matrix.

subroutine **dlascl** (TYPE, KL, KU, CFROM, CTO, M, N, A, LDA, INFO)

**DLASCL** multiplies a general rectangular matrix by a real scalar defined as cto/cfrom.

subroutine **dlasd0** (N, SQRE, D, E, U, LDU, VT, LDVT, SMLSIZ, IWORK, WORK, INFO)

**DLASD0** computes the singular values of a real upper bidiagonal n-by-m matrix B with diagonal d and off-diagonal e. Used by sbdsdc.

subroutine **dlasd1** (NL, NR, SQRE, D, ALPHA, BETA, U, LDU, VT, LDVT, IDXQ, IWORK, WORK, INFO)

**DLASD1** computes the SVD of an upper bidiagonal matrix B of the specified size. Used by sbdsdc.

subroutine **dlasd2** (NL, NR, SQRE, K, D, Z, ALPHA, BETA, U, LDU, VT, LDVT, DSIGMA, U2, LDU2, VT2, LDVT2, IDXP, IDX, IDXC, IDXQ, COLTYP, INFO)

**DLASD2** merges the two sets of singular values together into a single sorted set. Used by sbdsdc.

subroutine **dlasd3** (NL, NR, SQRE, K, D, Q, LDQ, DSIGMA, U, LDU, U2, LDU2, VT, LDVT, VT2, LDVT2, IDXC, CTOT, Z, INFO)

**DLASD3** finds all square roots of the roots of the secular equation, as defined by the values in D and Z, and then updates the singular vectors by matrix multiplication. Used by sbdsdc.

subroutine **dlasd4** (N, I, D, Z, DELTA, RHO, SIGMA, WORK, INFO)

**DLASD4** computes the square root of the i-th updated eigenvalue of a positive symmetric rank-one modification to a positive diagonal matrix. Used by dbdsdc.

subroutine **dlasd5** (I, D, Z, DELTA, RHO, DSIGMA, WORK)

**DLASD5** computes the square root of the i-th eigenvalue of a positive symmetric rank-one modification of a 2-by-2 diagonal matrix. Used by sbdsdc.

subroutine **dlasd6** (ICOMPQ, NL, NR, SQRE, D, VF, VL, ALPHA, BETA, IDXQ, PERM, GIVPTR, GIVCOL, LDGCOL, GIVNUM, LDGNUM, POLES, DIFL, DIFR, Z, K, C, S, WORK, IWORK, INFO)

**DLASD6** computes the SVD of an updated upper bidiagonal matrix obtained by merging two smaller ones by appending a row. Used by sbdsdc.

subroutine **dlasd7** (ICOMPQ, NL, NR, SQRE, K, D, Z, ZW, VF, VFW, VL, VLW, ALPHA, BETA, DSIGMA, IDX, IDXP, IDXQ, PERM, GIVPTR, GIVCOL, LDGCOL, GIVNUM, LDGNUM, C, S, INFO)

**DLASD7** merges the two sets of singular values together into a single sorted set. Then it tries to deflate the size of the problem. Used by sbdsdc.

subroutine **dlasd8** (ICOMPQ, K, D, Z, VF, VL, DIFL, DIFR, LDDIFR, DSIGMA, WORK, INFO)

**DLASD8** finds the square roots of the roots of the secular equation, and stores, for each element in D, the distance to its two nearest poles. Used by sbdsdc.

subroutine **dlasda** (ICOMPQ, SMLSIZ, N, SQRE, D, E, U, LDU, VT, K, DIFL, DIFR, Z, POLES, GIVPTR, GIVCOL, LDGCOL, PERM, GIVNUM, C, S, WORK, IWORK, INFO)

**DLASDA** computes the singular value decomposition (SVD) of a real upper bidiagonal matrix with diagonal d and off-diagonal e. Used by sbdsdc.

subroutine **dlasdq** (UPLO, SQRE, N, NCVT, NRU, NCC, D, E, VT, LDVT, U, LDU, C, LDC, WORK,



INFO)

**DLASDQ** computes the SVD of a real bidiagonal matrix with diagonal *d* and off-diagonal *e*.

Used by *sbdsc*.

subroutine **dlasdt** (N, LVL, ND, INODE, NDIML, NDIMR, MSUB)

**DLASDT** creates a tree of subproblems for bidiagonal divide and conquer. Used by *sbdsc*.

subroutine **dlaset** (UPLO, M, N, ALPHA, BETA, A, LDA)

**DLASET** initializes the off-diagonal elements and the diagonal elements of a matrix to given values.

subroutine **dlasr** (SIDE, PIVOT, DIRECT, M, N, C, S, A, LDA)

**DLASR** applies a sequence of plane rotations to a general rectangular matrix.

subroutine **dlasq** (N, X, INCX, SCALE, SUMSQ)

**DLASQ** updates a sum of squares represented in scaled form.

subroutine **dlasv2** (F, G, H, SSMIN, SSMAX, SNR, CSR, SNL, CSL)

**DLASV2** computes the singular value decomposition of a 2-by-2 triangular matrix.

integer function **ieeck** (ISPEC, ZERO, ONE)

**IEECK**

integer function **iladlc** (M, N, A, LDA)

**ILADLC** scans a matrix for its last non-zero column.

integer function **iladlr** (M, N, A, LDA)

**ILADLR** scans a matrix for its last non-zero row.

integer function **ilaenv** (ISPEC, NAME, OPTS, N1, N2, N3, N4)

**ILAENV**

integer function **ilaenv2stage** (ISPEC, NAME, OPTS, N1, N2, N3, N4)

**ILAENV2STAGE**

integer function **iparmq** (ISPEC, NAME, OPTS, N, ILO, IHI, LWORK)

**IPARMQ**

logical function **lsamen** (N, CA, CB)

**LSAMEN**

subroutine **scombssq** (V1, V2)

**SCOMBSSQ** adds two scaled sum of squares quantities

logical function **sisnan** (SIN)

**SISNAN** tests input for NaN.

subroutine **slabad** (SMALL, LARGE)

**SLABAD**

subroutine **slacpy** (UPLO, M, N, A, LDA, B, LDB)

**SLACPY** copies all or part of one two-dimensional array to another.

subroutine **slae2** (A, B, C, RT1, RT2)

**SLAE2** computes the eigenvalues of a 2-by-2 symmetric matrix.

subroutine **slaebz** (IJOB, NITMAX, N, MMAX, MINP, NBMIN, ABSTOL, RELTOL, PIVMIN, D, E, E2, NVAL, AB, C, MOUT, NAB, WORK, IWORK, INFO)

**SLAEBZ** computes the number of eigenvalues of a real symmetric tridiagonal matrix which are less than or equal to a given value, and performs other tasks required by the routine *sstebz*.

subroutine **slaev2** (A, B, C, RT1, RT2, CS1, SN1)

**SLAEV2** computes the eigenvalues and eigenvectors of a 2-by-2 symmetric/Hermitian matrix.

subroutine **slag2d** (M, N, SA, LDSA, A, LDA, INFO)

**SLAG2D** converts a single precision matrix to a double precision matrix.

subroutine **slagts** (JOB, N, A, B, C, D, IN, Y, TOL, INFO)

**SLAGTS** solves the system of equations  $(T-\lambda I)x = y$  or  $(T-\lambda I)Tx = y$ , where *T* is a general tridiagonal matrix and  $\lambda$  a scalar, using the LU factorization computed by *slagtf*.

logical function **slaisnan** (SIN1, SIN2)

**SLAISNAN** tests input for NaN by comparing two arguments for inequality.

integer function **slaneg** (N, D, LLD, SIGMA, PIVMIN, R)

**SLANEG** computes the Sturm count.

real function **slanst** (NORM, N, D, E)

**SLANST** returns the value of the 1-norm, or the Frobenius norm, or the infinity norm, or the element of largest absolute value of a real symmetric tridiagonal matrix.

real function **slapy2** (X, Y)

**SLAPY2** returns  $\sqrt{x^2+y^2}$ .

real function **slapy3** (X, Y, Z)



**SLAPY3** returns  $\sqrt{x^2+y^2+z^2}$ .

subroutine **slarnv** (IDIST, ISEED, N, X)  
**SLARNV** returns a vector of random numbers from a uniform or normal distribution.

subroutine **slarra** (N, D, E, E2, SPLTOL, TNRM, NSPLIT, ISPLIT, INFO)  
**SLARRA** computes the splitting points with the specified threshold.

subroutine **slarrb** (N, D, LLD, IFIRST, ILAST, RTOL1, RTOL2, OFFSET, W, WGAP, WERR, WORK, IWORK, PIVMIN, SPDIAM, TWIST, INFO)  
**SLARRB** provides limited bisection to locate eigenvalues for more accuracy.

subroutine **slarrc** (JOB, N, VL, VU, D, E, PIVMIN, EIGCNT, LCNT, RCNT, INFO)  
**SLARRC** computes the number of eigenvalues of the symmetric tridiagonal matrix.

subroutine **slarrd** (RANGE, ORDER, N, VL, VU, IL, IU, GERS, RELTOL, D, E, E2, PIVMIN, NSPLIT, ISPLIT, M, W, WERR, WL, WU, IBLOCK, INDEXW, WORK, IWORK, INFO)  
**SLARRD** computes the eigenvalues of a symmetric tridiagonal matrix to suitable accuracy.

subroutine **slarre** (RANGE, N, VL, VU, IL, IU, D, E, E2, RTOL1, RTOL2, SPLTOL, NSPLIT, ISPLIT, M, W, WERR, WGAP, IBLOCK, INDEXW, GERS, PIVMIN, WORK, IWORK, INFO)  
**SLARRE** given the tridiagonal matrix T, sets small off-diagonal elements to zero and for each unreduced block  $T_i$ , finds base representations and eigenvalues.

subroutine **slarrf** (N, D, L, LD, CLSTRT, CLEND, W, WGAP, WERR, SPDIAM, CLGAPL, CLGAPR, PIVMIN, SIGMA, DPLUS, LPLUS, WORK, INFO)  
**SLARRF** finds a new relatively robust representation such that at least one of the eigenvalues is relatively isolated.

subroutine **slarrj** (N, D, E2, IFIRST, ILAST, RTOL, OFFSET, W, WERR, WORK, IWORK, PIVMIN, SPDIAM, INFO)  
**SLARRJ** performs refinement of the initial estimates of the eigenvalues of the matrix T.

subroutine **slarrk** (N, IW, GL, GU, D, E2, PIVMIN, RELTOL, W, WERR, INFO)  
**SLARRK** computes one eigenvalue of a symmetric tridiagonal matrix T to suitable accuracy.

subroutine **slarrs** (N, D, E, INFO)  
**SLARRS** performs tests to decide whether the symmetric tridiagonal matrix T warrants expensive computations which guarantee high relative accuracy in the eigenvalues.

subroutine **slartg** (F, G, CS, SN, R)  
**SLARTG** generates a plane rotation with real cosine and real sine.

subroutine **slartgp** (F, G, CS, SN, R)  
**SLARTGP** generates a plane rotation so that the diagonal is nonnegative.

subroutine **slaruv** (ISEED, N, X)  
**SLARUV** returns a vector of n random real numbers from a uniform distribution.

subroutine **slas2** (F, G, H, SSMIN, SSMAX)  
**SLAS2** computes singular values of a 2-by-2 triangular matrix.

subroutine **slascl** (TYPE, KL, KU, CFROM, CTO, M, N, A, LDA, INFO)  
**SLASCL** multiplies a general rectangular matrix by a real scalar defined as cto/cfrom.

subroutine **slasd0** (N, SQRE, D, E, U, LDU, VT, LDVT, SMLSIZ, IWORK, WORK, INFO)  
**SLASD0** computes the singular values of a real upper bidiagonal n-by-m matrix B with diagonal d and off-diagonal e. Used by sbdsdc.

subroutine **slasd1** (NL, NR, SQRE, D, ALPHA, BETA, U, LDU, VT, LDVT, IDXQ, IWORK, WORK, INFO)  
**SLASD1** computes the SVD of an upper bidiagonal matrix B of the specified size. Used by sbdsdc.

subroutine **slasd2** (NL, NR, SQRE, K, D, Z, ALPHA, BETA, U, LDU, VT, LDVT, DSIGMA, U2, LDU2, VT2, LDVT2, IDXP, IDX, IDXC, IDXQ, COLTYP, INFO)  
**SLASD2** merges the two sets of singular values together into a single sorted set. Used by sbdsdc.

subroutine **slasd3** (NL, NR, SQRE, K, D, Q, LDQ, DSIGMA, U, LDU, U2, LDU2, VT, LDVT, VT2, LDVT2, IDXC, CTOT, Z, INFO)  
**SLASD3** finds all square roots of the roots of the secular equation, as defined by the values in D and Z, and then updates the singular vectors by matrix multiplication. Used by sbdsdc.

subroutine **slasd4** (N, I, D, Z, DELTA, RHO, SIGMA, WORK, INFO)  
**SLASD4** computes the square root of the i-th updated eigenvalue of a positive symmetric rank-one modification to a positive diagonal matrix. Used by sbdsdc.

subroutine **slasd5** (I, D, Z, DELTA, RHO, DSIGMA, WORK)



**SLASD5** computes the square root of the  $i$ -th eigenvalue of a positive symmetric rank-one modification of a 2-by-2 diagonal matrix. Used by sbdsdc.

subroutine **slasd6** (ICOMPQ, NL, NR, SQRE, D, VF, VL, ALPHA, BETA, IDXQ, PERM, GIVPTR, GIVCOL, LDGCOL, GIVNUM, LDGNUM, POLES, DIFL, DIFR, Z, K, C, S, WORK, IWORK, INFO)

**SLASD6** computes the SVD of an updated upper bidiagonal matrix obtained by merging two smaller ones by appending a row. Used by sbdsdc.

subroutine **slasd7** (ICOMPQ, NL, NR, SQRE, K, D, Z, ZW, VF, VFW, VL, VLW, ALPHA, BETA, DSIGMA, IDX, IDXP, IDXQ, PERM, GIVPTR, GIVCOL, LDGCOL, GIVNUM, LDGNUM, C, S, INFO)

**SLASD7** merges the two sets of singular values together into a single sorted set. Then it tries to deflate the size of the problem. Used by sbdsdc.

subroutine **slasd8** (ICOMPQ, K, D, Z, VF, VL, DIFL, DIFR, LDDIFR, DSIGMA, WORK, INFO)

**SLASD8** finds the square roots of the roots of the secular equation, and stores, for each element in D, the distance to its two nearest poles. Used by sbdsdc.

subroutine **slasda** (ICOMPQ, SMLSIZ, N, SQRE, D, E, U, LDU, VT, K, DIFL, DIFR, Z, POLES, GIVPTR, GIVCOL, LDGCOL, PERM, GIVNUM, C, S, WORK, IWORK, INFO)

**SLASDA** computes the singular value decomposition (SVD) of a real upper bidiagonal matrix with diagonal d and off-diagonal e. Used by sbdsdc.

subroutine **slasdq** (UPLO, SQRE, N, NCVT, NRU, NCC, D, E, VT, LDVT, U, LDU, C, LDC, WORK, INFO)

**SLASDQ** computes the SVD of a real bidiagonal matrix with diagonal d and off-diagonal e. Used by sbdsdc.

subroutine **slasdt** (N, LVL, ND, INODE, NDIML, NDIMR, MSUB)

**SLASDT** creates a tree of subproblems for bidiagonal divide and conquer. Used by sbdsdc.

subroutine **slaset** (UPLO, M, N, ALPHA, BETA, A, LDA)

**SLASET** initializes the off-diagonal elements and the diagonal elements of a matrix to given values.

subroutine **slasr** (SIDE, PIVOT, DIRECT, M, N, C, S, A, LDA)

**SLASR** applies a sequence of plane rotations to a general rectangular matrix.

subroutine **slasq** (N, X, INCX, SCALE, SUMSQ)

**SLASQ** updates a sum of squares represented in scaled form.

subroutine **slasv2** (F, G, H, SSMIN, SSMAX, SNR, CSR, SNL, CSL)

**SLASV2** computes the singular value decomposition of a 2-by-2 triangular matrix.

subroutine **xerbla** (SRNAME, INFO)

**XERBLA**

subroutine **xerbla\_array** (SRNAME\_ARRAY, SRNAME\_LEN, INFO)

**XERBLA\_ARRAY**

## Detailed Description

This is the group of Other Auxiliary routines

## Function Documentation

subroutine **dcombssq** (double precision, dimension( 2 ) V1, double precision, dimension( 2 ) V2)

**DCOMBSSQ** adds two scaled sum of squares quantities.

### Purpose:

**DCOMBSSQ** adds two scaled sum of squares quantities,  $V1 := V1 + V2$ .

That is,

$$V1\_scale^{**2} * V1\_sumsq := V1\_scale^{**2} * V1\_sumsq + V2\_scale^{**2} * V2\_sumsq$$

### Parameters

V1

V1 is DOUBLE PRECISION array, dimension (2).

The first scaled sum.

$V1(1) = V1\_scale$ ,  $V1(2) = V1\_sumsq$ .

V2



V2 is DOUBLE PRECISION array, dimension (2).  
 The second scaled sum.  
 $V2(1) = V2\_scale$ ,  $V2(2) = V2\_sumsq$ .

**Author**

Univ. of Tennessee  
 Univ. of California Berkeley  
 Univ. of Colorado Denver  
 NAG Ltd.

**Date**

November 2018

**logical function disnan (double precision, intent(in) DIN)**

**DISNAN** tests input for NaN.

**Purpose:**

DISNAN returns .TRUE. if its argument is NaN, and .FALSE. otherwise. To be replaced by the Fortran 2003 intrinsic in the future.

**Parameters**

*DIN*

DIN is DOUBLE PRECISION  
 Input to test for NaN.

**Author**

Univ. of Tennessee  
 Univ. of California Berkeley  
 Univ. of Colorado Denver  
 NAG Ltd.

**Date**

June 2017

**subroutine dlabad (double precision SMALL, double precision LARGE)**

**DLABAD**

**Purpose:**

DLABAD takes as input the values computed by DLAMCH for underflow and overflow, and returns the square root of each of these values if the log of LARGE is sufficiently large. This subroutine is intended to identify machines with a large exponent range, such as the Crays, and redefine the underflow and overflow limits to be the square roots of the values computed by DLAMCH. This subroutine is needed because DLAMCH does not compensate for poor arithmetic in the upper half of the exponent range, as is found on a Cray.

**Parameters**

*SMALL*

SMALL is DOUBLE PRECISION  
 On entry, the underflow threshold as computed by DLAMCH.  
 On exit, if LOG10(LARGE) is sufficiently large, the square root of SMALL, otherwise unchanged.

*LARGE*

LARGE is DOUBLE PRECISION  
 On entry, the overflow threshold as computed by DLAMCH.  
 On exit, if LOG10(LARGE) is sufficiently large, the square



root of LARGE, otherwise unchanged.

#### Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

#### Date

December 2016

**subroutine dlacpy (character UPLO, integer M, integer N, double precision, dimension( lda, \* ) A, integer LDA, double precision, dimension( ldb, \* ) B, integer LDB)**

**DLACPY** copies all or part of one two-dimensional array to another.

#### Purpose:

DLACPY copies all or part of a two-dimensional matrix A to another matrix B.

#### Parameters

##### UPLO

UPLO is CHARACTER\*1

Specifies the part of the matrix A to be copied to B.

= 'U': Upper triangular part

= 'L': Lower triangular part

Otherwise: All of the matrix A

##### M

M is INTEGER

The number of rows of the matrix A.  $M \geq 0$ .

##### N

N is INTEGER

The number of columns of the matrix A.  $N \geq 0$ .

##### A

A is DOUBLE PRECISION array, dimension (LDA,N)

The m by n matrix A. If UPLO = 'U', only the upper triangle or trapezoid is accessed; if UPLO = 'L', only the lower triangle or trapezoid is accessed.

##### LDA

LDA is INTEGER

The leading dimension of the array A.  $LDA \geq \max(1,M)$ .

##### B

B is DOUBLE PRECISION array, dimension (LDB,N)

On exit,  $B = A$  in the locations specified by UPLO.

##### LDB

LDB is INTEGER

The leading dimension of the array B.  $LDB \geq \max(1,M)$ .

#### Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

#### Date



December 2016

**subroutine dlae2 (double precision A, double precision B, double precision C, double precision RT1, double precision RT2)**

**DLAE2** computes the eigenvalues of a 2-by-2 symmetric matrix.

**Purpose:**

DLAE2 computes the eigenvalues of a 2-by-2 symmetric matrix

[ A B ]

[ B C ].

On return, RT1 is the eigenvalue of larger absolute value, and RT2 is the eigenvalue of smaller absolute value.

**Parameters**

*A*

A is DOUBLE PRECISION

The (1,1) element of the 2-by-2 matrix.

*B*

B is DOUBLE PRECISION

The (1,2) and (2,1) elements of the 2-by-2 matrix.

*C*

C is DOUBLE PRECISION

The (2,2) element of the 2-by-2 matrix.

*RT1*

RT1 is DOUBLE PRECISION

The eigenvalue of larger absolute value.

*RT2*

RT2 is DOUBLE PRECISION

The eigenvalue of smaller absolute value.

**Author**

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

**Date**

December 2016

**Further Details:**

RT1 is accurate to a few ulps barring over/underflow.

RT2 may be inaccurate if there is massive cancellation in the determinant  $A \cdot C - B \cdot B$ ; higher precision or correctly rounded or correctly truncated arithmetic would be needed to compute RT2 accurately in all cases.

Overflow is possible only if RT1 is within a factor of 5 of overflow.

Underflow is harmless if the input data is 0 or exceeds  
underflow\_threshold / macheps.

**subroutine dlaebz (integer IJOB, integer NITMAX, integer N, integer MMAX, integer MINP, integer NBMIN, double precision ABSTOL, double precision RELTOL, double precision PIVMIN, double precision, dimension( \* ) D, double precision, dimension( \* ) E, double precision, dimension( \* ) E2, integer, dimension( \* ) NVAL, double precision, dimension( mmax, \* ) AB,**





**double precision, dimension( \* ) C, integer MOUT, integer, dimension( mmax, \* ) NAB, double precision, dimension( \* ) WORK, integer, dimension( \* ) IWORK, integer INFO)**

**DLAEBZ** computes the number of eigenvalues of a real symmetric tridiagonal matrix which are less than or equal to a given value, and performs other tasks required by the routine sstebz.

#### **Purpose:**

DLAEBZ contains the iteration loops which compute and use the function  $N(w)$ , which is the count of eigenvalues of a symmetric tridiagonal matrix  $T$  less than or equal to its argument  $w$ . It performs a choice of two types of loops:

IJOB=1, followed by

IJOB=2: It takes as input a list of intervals and returns a list of sufficiently small intervals whose union contains the same eigenvalues as the union of the original intervals.

The input intervals are  $(AB(j,1), AB(j,2)]$ ,  $j=1, \dots, MINP$ .

The output interval  $(AB(j,1), AB(j,2)]$  will contain eigenvalues  $NAB(j,1)+1, \dots, NAB(j,2)$ , where  $1 \leq j \leq MOUT$ .

IJOB=3: It performs a binary search in each input interval  $(AB(j,1), AB(j,2)]$  for a point  $w(j)$  such that  $N(w(j)) = NVAL(j)$ , and uses  $C(j)$  as the starting point of the search. If such a  $w(j)$  is found, then on output  $AB(j,1) = AB(j,2) = w$ . If no such  $w(j)$  is found, then on output  $(AB(j,1), AB(j,2)]$  will be a small interval containing the point where  $N(w)$  jumps through  $NVAL(j)$ , unless that point lies outside the initial interval.

Note that the intervals are in all cases half-open intervals, i.e., of the form  $(a, b]$ , which includes  $b$  but not  $a$ .

To avoid underflow, the matrix should be scaled so that its largest element is no greater than  $\text{overflow}^{**}(1/2) * \text{underflow}^{**}(1/4)$  in absolute value. To assure the most accurate computation of small eigenvalues, the matrix should be scaled to be not much smaller than that, either.

See W. Kahan "Accurate Eigenvalues of a Symmetric Tridiagonal Matrix", Report CS41, Computer Science Dept., Stanford University, July 21, 1966

Note: the arguments are, in general, \*not\* checked for unreasonable values.

#### **Parameters**

##### *IJOB*

IJOB is INTEGER

Specifies what is to be done:

= 1: Compute NAB for the initial intervals.

= 2: Perform bisection iteration to find eigenvalues of  $T$ .

= 3: Perform bisection iteration to invert  $N(w)$ , i.e., to find a point which has a specified number of eigenvalues of  $T$  to its left.

Other values will cause DLAEBZ to return with  $INFO=-1$ .

##### *NITMAX*

NITMAX is INTEGER

The maximum number of "levels" of bisection to be



performed, i.e., an interval of width  $W$  will not be made smaller than  $2^{(-NITMAX)} * W$ . If not all intervals have converged after NITMAX iterations, then INFO is set to the number of non-converged intervals.

*N*

*N* is INTEGER

The dimension  $n$  of the tridiagonal matrix  $T$ . It must be at least 1.

*MMAX*

*MMAX* is INTEGER

The maximum number of intervals. If more than *MMAX* intervals are generated, then DLAEBZ will quit with  $INFO=MMAX+1$ .

*MINP*

*MINP* is INTEGER

The initial number of intervals. It may not be greater than *MMAX*.

*NBMIN*

*NBMIN* is INTEGER

The smallest number of intervals that should be processed using a vector loop. If zero, then only the scalar loop will be used.

*ABSTOL*

*ABSTOL* is DOUBLE PRECISION

The minimum (absolute) width of an interval. When an interval is narrower than *ABSTOL*, or than *RELTOL* times the larger (in magnitude) endpoint, then it is considered to be sufficiently small, i.e., converged. This must be at least zero.

*RELTOL*

*RELTOL* is DOUBLE PRECISION

The minimum relative width of an interval. When an interval is narrower than *ABSTOL*, or than *RELTOL* times the larger (in magnitude) endpoint, then it is considered to be sufficiently small, i.e., converged. Note: this should always be at least  $\text{radix} * \text{machine epsilon}$ .

*PIVMIN*

*PIVMIN* is DOUBLE PRECISION

The minimum absolute value of a "pivot" in the Sturm sequence loop.

This must be at least  $\max |e(j)|^2 * \text{safe\_min}$  and at least *safe\_min*, where *safe\_min* is at least the smallest number that can divide one without overflow.

*D*

*D* is DOUBLE PRECISION array, dimension (*N*)

The diagonal elements of the tridiagonal matrix  $T$ .

*E*

*E* is DOUBLE PRECISION array, dimension (*N*)

The offdiagonal elements of the tridiagonal matrix  $T$  in positions 1 through  $N-1$ .  $E(N)$  is arbitrary.

*E2*

*E2* is DOUBLE PRECISION array, dimension (*N*)



The squares of the offdiagonal elements of the tridiagonal matrix  $T$ .  $E2(N)$  is ignored.

#### *NVAL*

*NVAL* is INTEGER array, dimension (MINP)

If *IJOB*=1 or 2, not referenced.

If *IJOB*=3, the desired values of  $N(w)$ . The elements of *NVAL* will be reordered to correspond with the intervals in *AB*.

Thus, *NVAL*(*j*) on output will not, in general be the same as *NVAL*(*j*) on input, but it will correspond with the interval (*AB*(*j*,1),*AB*(*j*,2)] on output.

#### *AB*

*AB* is DOUBLE PRECISION array, dimension (MMAX,2)

The endpoints of the intervals. *AB*(*j*,1) is *a*(*j*), the left endpoint of the *j*-th interval, and *AB*(*j*,2) is *b*(*j*), the right endpoint of the *j*-th interval. The input intervals will, in general, be modified, split, and reordered by the calculation.

#### *C*

*C* is DOUBLE PRECISION array, dimension (MMAX)

If *IJOB*=1, ignored.

If *IJOB*=2, workspace.

If *IJOB*=3, then on input *C*(*j*) should be initialized to the first search point in the binary search.

#### *MOUT*

*MOUT* is INTEGER

If *IJOB*=1, the number of eigenvalues in the intervals.

If *IJOB*=2 or 3, the number of intervals output.

If *IJOB*=3, *MOUT* will equal MINP.

#### *NAB*

*NAB* is INTEGER array, dimension (MMAX,2)

If *IJOB*=1, then on output *NAB*(*i*,*j*) will be set to  $N(AB(i,j))$ .

If *IJOB*=2, then on input, *NAB*(*i*,*j*) should be set. It must satisfy the condition:

$N(AB(i,1)) \leq NAB(i,1) \leq NAB(i,2) \leq N(AB(i,2))$ ,

which means that in interval *i* only eigenvalues

$NAB(i,1)+1, \dots, NAB(i,2)$  will be considered. Usually,

$NAB(i,j)=N(AB(i,j))$ , from a previous call to DLAEBZ with *IJOB*=1.

On output, *NAB*(*i*,*j*) will contain

$\max(na(k), \min(nb(k), N(AB(i,j))))$ , where *k* is the index of the input interval that the output interval

(*AB*(*j*,1),*AB*(*j*,2)] came from, and *na*(*k*) and *nb*(*k*) are the input values of *NAB*(*k*,1) and *NAB*(*k*,2).

If *IJOB*=3, then on output, *NAB*(*i*,*j*) contains  $N(AB(i,j))$ , unless  $N(w) > NVAL(i)$  for all search points *w*, in which case *NAB*(*i*,1) will not be modified, i.e., the output value will be the same as the input value (modulo reorderings -- see *NVAL* and *AB*), or unless  $N(w) < NVAL(i)$  for all search points *w*, in which case *NAB*(*i*,2) will not be modified. Normally, *NAB* should be set to some distinctive value(s) before DLAEBZ is called.

#### *WORK*

*WORK* is DOUBLE PRECISION array, dimension (MMAX)

Workspace.



**WORK**

WORK is INTEGER array, dimension (MMAX)  
Workspace.

**INFO**

INFO is INTEGER  
= 0: All intervals converged.  
= 1--MMAX: The last INFO intervals did not converge.  
= MMAX+1: More than MMAX intervals were generated.

**Author**

Univ. of Tennessee  
Univ. of California Berkeley  
Univ. of Colorado Denver  
NAG Ltd.

**Date**

December 2016

**Further Details:**

This routine is intended to be called only by other LAPACK routines, thus the interface is less user-friendly. It is intended for two purposes:

- (a) finding eigenvalues. In this case, DLAEBZ should have one or more initial intervals set up in AB, and DLAEBZ should be called with IJOB=1. This sets up NAB, and also counts the eigenvalues. Intervals with no eigenvalues would usually be thrown out at this point. Also, if not all the eigenvalues in an interval  $i$  are desired, NAB( $i,1$ ) can be increased or NAB( $i,2$ ) decreased. For example, set NAB( $i,1$ )=NAB( $i,2$ )-1 to get the largest eigenvalue. DLAEBZ is then called with IJOB=2 and MMAX no smaller than the value of MOUT returned by the call with IJOB=1. After this (IJOB=2) call, eigenvalues NAB( $i,1$ )+1 through NAB( $i,2$ ) are approximately AB( $i,1$ ) (or AB( $i,2$ )) to the tolerance specified by ABSTOL and RELTOL.
- (b) finding an interval  $(a',b']$  containing eigenvalues  $w(f),\dots,w(l)$ . In this case, start with a Gershgorin interval  $(a,b)$ . Set up AB to contain 2 search intervals, both initially  $(a,b)$ . One NVAL element should contain  $f-1$  and the other should contain 1, while C should contain  $a$  and  $b$ , resp. NAB( $i,1$ ) should be -1 and NAB( $i,2$ ) should be  $N+1$ , to flag an error if the desired interval does not lie in  $(a,b)$ . DLAEBZ is then called with IJOB=3. On exit, if  $w(f-1) < w(f)$ , then one of the intervals --  $j$  -- will have AB( $j,1$ )=AB( $j,2$ ) and NAB( $j,1$ )=NAB( $j,2$ )= $f-1$ , while if, to the specified tolerance,  $w(f-k)=\dots=w(f+r)$ ,  $k > 0$  and  $r \geq 0$ , then the interval will have N(AB( $j,1$ ))=NAB( $j,1$ )= $f-k$  and N(AB( $j,2$ ))=NAB( $j,2$ )= $f+r$ . The cases  $w(l) < w(l+1)$  and  $w(l-r)=\dots=w(l+k)$  are handled similarly.

**subroutine dlaev2 (double precision A, double precision B, double precision C, double precision RT1, double precision RT2, double precision CS1, double precision SN1)**

**DLAEV2** computes the eigenvalues and eigenvectors of a 2-by-2 symmetric/Hermitian matrix.

**Purpose:**

DLAEV2 computes the eigendecomposition of a 2-by-2 symmetric matrix



$$\begin{bmatrix} A & B \\ B & C \end{bmatrix}.$$

On return, RT1 is the eigenvalue of larger absolute value, RT2 is the eigenvalue of smaller absolute value, and (CS1,SN1) is the unit right eigenvector for RT1, giving the decomposition

$$\begin{bmatrix} CS1 & SN1 \\ -SN1 & CS1 \end{bmatrix} \begin{bmatrix} A & B \\ B & C \end{bmatrix} \begin{bmatrix} CS1 & -SN1 \\ SN1 & CS1 \end{bmatrix} = \begin{bmatrix} RT1 & 0 \\ 0 & RT2 \end{bmatrix}.$$

### Parameters

*A*

*A* is DOUBLE PRECISION  
The (1,1) element of the 2-by-2 matrix.

*B*

*B* is DOUBLE PRECISION  
The (1,2) element and the conjugate of the (2,1) element of the 2-by-2 matrix.

*C*

*C* is DOUBLE PRECISION  
The (2,2) element of the 2-by-2 matrix.

*RT1*

*RT1* is DOUBLE PRECISION  
The eigenvalue of larger absolute value.

*RT2*

*RT2* is DOUBLE PRECISION  
The eigenvalue of smaller absolute value.

*CS1*

*CS1* is DOUBLE PRECISION

*SN1*

*SN1* is DOUBLE PRECISION  
The vector (*CS1*, *SN1*) is a unit right eigenvector for *RT1*.

### Author

Univ. of Tennessee  
Univ. of California Berkeley  
Univ. of Colorado Denver  
NAG Ltd.

### Date

December 2016

### Further Details:

*RT1* is accurate to a few ulps barring over/underflow.

*RT2* may be inaccurate if there is massive cancellation in the determinant  $A \cdot C - B \cdot B$ ; higher precision or correctly rounded or correctly truncated arithmetic would be needed to compute *RT2* accurately in all cases.

*CS1* and *SN1* are accurate to a few ulps barring over/underflow.

Overflow is possible only if *RT1* is within a factor of 5 of overflow.  
Underflow is harmless if the input data is 0 or exceeds



underflow\_threshold / macheps.

**subroutine dlagts (integer JOB, integer N, double precision, dimension( \* ) A, double precision, dimension( \* ) B, double precision, dimension( \* ) C, double precision, dimension( \* ) D, integer, dimension( \* ) IN, double precision, dimension( \* ) Y, double precision TOL, integer INFO)**  
**DLAGTS** solves the system of equations  $(T - \lambda I)x = y$  or  $(T - \lambda I)Tx = y$ , where  $T$  is a general tridiagonal matrix and  $\lambda$  a scalar, using the LU factorization computed by **slagtf**.

#### **Purpose:**

DLAGTS may be used to solve one of the systems of equations

$$(T - \lambda I)x = y \quad \text{or} \quad (T - \lambda I)Tx = y,$$

where  $T$  is an  $n$  by  $n$  tridiagonal matrix, for  $x$ , following the factorization of  $(T - \lambda I)$  as

$$(T - \lambda I) = P * L * U,$$

by routine **DLAGTF**. The choice of equation to be solved is controlled by the argument **JOB**, and in each case there is an option to perturb zero or very small diagonal elements of  $U$ , this option being intended for use in applications such as inverse iteration.

#### **Parameters**

##### *JOB*

**JOB** is **INTEGER**

Specifies the job to be performed by **DLAGTS** as follows:

- = 1: The equations  $(T - \lambda I)x = y$  are to be solved, but diagonal elements of  $U$  are not to be perturbed.
- = -1: The equations  $(T - \lambda I)x = y$  are to be solved and, if overflow would otherwise occur, the diagonal elements of  $U$  are to be perturbed. See argument **TOL** below.
- = 2: The equations  $(T - \lambda I)Tx = y$  are to be solved, but diagonal elements of  $U$  are not to be perturbed.
- = -2: The equations  $(T - \lambda I)Tx = y$  are to be solved and, if overflow would otherwise occur, the diagonal elements of  $U$  are to be perturbed. See argument **TOL** below.

##### *N*

**N** is **INTEGER**

The order of the matrix  $T$ .

##### *A*

**A** is **DOUBLE PRECISION** array, dimension (**N**)

On entry, **A** must contain the diagonal elements of  $U$  as returned from **DLAGTF**.

##### *B*

**B** is **DOUBLE PRECISION** array, dimension (**N**-1)

On entry, **B** must contain the first super-diagonal elements of  $U$  as returned from **DLAGTF**.

##### *C*

**C** is **DOUBLE PRECISION** array, dimension (**N**-1)

On entry, **C** must contain the sub-diagonal elements of  $L$  as returned from **DLAGTF**.

##### *D*



D is DOUBLE PRECISION array, dimension (N-2)  
On entry, D must contain the second super-diagonal elements of U as returned from DLAGTF.

*IN*

IN is INTEGER array, dimension (N)  
On entry, IN must contain details of the matrix P as returned from DLAGTF.

*Y*

Y is DOUBLE PRECISION array, dimension (N)  
On entry, the right hand side vector y.  
On exit, Y is overwritten by the solution vector x.

*TOL*

TOL is DOUBLE PRECISION  
On entry, with  $JOB < 0$ , TOL should be the minimum perturbation to be made to very small diagonal elements of U. TOL should normally be chosen as about  $\epsilon \cdot \text{norm}(U)$ , where  $\epsilon$  is the relative machine precision, but if TOL is supplied as non-positive, then it is reset to  $\epsilon \cdot \max(\text{abs}(u(i,j)))$ .  
If  $JOB > 0$  then TOL is not referenced.

On exit, TOL is changed as described above, only if TOL is non-positive on entry. Otherwise TOL is unchanged.

*INFO*

INFO is INTEGER  
= 0: successful exit  
< 0: if  $INFO = -i$ , the i-th argument had an illegal value  
> 0: overflow would occur when computing the  $INFO(th)$  element of the solution vector x. This can only occur when JOB is supplied as positive and either means that a diagonal element of U is very small, or that the elements of the right-hand side vector y are very large.

#### Author

Univ. of Tennessee  
Univ. of California Berkeley  
Univ. of Colorado Denver  
NAG Ltd.

#### Date

December 2016

**logical function dlaisnan (double precision, intent(in) DIN1, double precision, intent(in) DIN2)**

**DLAISNAN** tests input for NaN by comparing two arguments for inequality.

#### Purpose:

This routine is not for general use. It exists solely to avoid over-optimization in DISNAN.

DLAISNAN checks for NaNs by comparing its two arguments for inequality. NaN is the only floating-point value where  $\text{NaN} \neq \text{NaN}$  returns .TRUE. To check for NaNs, pass the same variable as both arguments.

A compiler must assume that the two arguments are



not the same variable, and the test will not be optimized away.  
 Interprocedural or whole-program optimization may delete this  
 test. The ISNAN functions will be replaced by the correct  
 Fortran 03 intrinsic once the intrinsic is widely available.

### Parameters

*DIN1*

DIN1 is DOUBLE PRECISION

*DIN2*

DIN2 is DOUBLE PRECISION

Two numbers to compare for inequality.

### Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

### Date

June 2017

**integer function dlaneg (integer N, double precision, dimension( \* ) D, double precision, dimension( \* )**

**LLD, double precision SIGMA, double precision PIVMIN, integer R)**

**DLANEG** computes the Sturm count.

### Purpose:

DLANEG computes the Sturm count, the number of negative pivots  
 encountered while factoring tridiagonal  $T - \sigma I = L D L^T$ .  
 This implementation works directly on the factors without forming  
 the tridiagonal matrix  $T$ . The Sturm count is also the number of  
 eigenvalues of  $T$  less than  $\sigma$ .

This routine is called from DLARRB.

The current routine does not use the PIVMIN parameter but rather  
 requires IEEE-754 propagation of Infinities and NaNs. This  
 routine also has no input range restrictions but does require  
 default exception handling such that  $x/0$  produces Inf when  $x$  is  
 non-zero, and Inf/Inf produces NaN. For more information, see:

Marques, Riedy, and Voemel, "Benefits of IEEE-754 Features in  
 Modern Symmetric Tridiagonal Eigensolvers," SIAM Journal on  
 Scientific Computing, v28, n5, 2006. DOI 10.1137/050641624  
 (Tech report version in LAWN 172 with the same title.)

### Parameters

*N*

N is INTEGER

The order of the matrix.

*D*

D is DOUBLE PRECISION array, dimension (N)

The N diagonal elements of the diagonal matrix D.

*LLD*

LLD is DOUBLE PRECISION array, dimension (N-1)

The (N-1) elements  $L(i)*L(i)*D(i)$ .

*SIGMA*





SIGMA is DOUBLE PRECISION  
Shift amount in T -  $\sigma I = L D L^T$ .

#### PIVMIN

PIVMIN is DOUBLE PRECISION  
The minimum pivot in the Sturm sequence. May be used when zero pivots are encountered on non-IEEE-754 architectures.

#### R

R is INTEGER  
The twist index for the twisted factorization that is used for the negcount.

#### Author

Univ. of Tennessee  
Univ. of California Berkeley  
Univ. of Colorado Denver  
NAG Ltd.

#### Date

December 2016

#### Contributors:

Osni Marques, LBNL/NERSC, USA  
Christof Voemel, University of California, Berkeley, USA  
Jason Riedy, University of California, Berkeley, USA

**double precision function dlanst (character NORM, integer N, double precision, dimension( \* ) D, double precision, dimension( \* ) E)**

**DLANST** returns the value of the 1-norm, or the Frobenius norm, or the infinity norm, or the element of largest absolute value of a real symmetric tridiagonal matrix.

#### Purpose:

DLANST returns the value of the one norm, or the Frobenius norm, or the infinity norm, or the element of largest absolute value of a real symmetric tridiagonal matrix A.

#### Returns

DLANST

DLANST = ( max(abs(A(i,j))), NORM = 'M' or 'm'

(  
( norm1(A), NORM = '1', 'O' or 'o'  
(  
( normI(A), NORM = 'I' or 'i'  
(  
( normF(A), NORM = 'F', 'f', 'E' or 'e'

where norm1 denotes the one norm of a matrix (maximum column sum), normI denotes the infinity norm of a matrix (maximum row sum) and normF denotes the Frobenius norm of a matrix (square root of sum of squares). Note that max(abs(A(i,j))) is not a consistent matrix norm.

#### Parameters

*NORM*

NORM is CHARACTER\*1  
Specifies the value to be returned in DLANST as described above.

*N*



OTHERauxiliary(3)

LAPACK

OTHERauxiliary(3)

$N$  is INTEGER

The order of the matrix  $A$ .  $N \geq 0$ . When  $N = 0$ , DLANST is set to zero.

$D$

$D$  is DOUBLE PRECISION array, dimension ( $N$ )  
The diagonal elements of  $A$ .

$E$

$E$  is DOUBLE PRECISION array, dimension ( $N-1$ )  
The ( $n-1$ ) sub-diagonal or super-diagonal elements of  $A$ .

#### Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

#### Date

December 2016

**double precision function dlapy2 (double precision  $X$ , double precision  $Y$ )**

DLAPY2 returns  $\sqrt{x^2+y^2}$ .

#### Purpose:

DLAPY2 returns  $\sqrt{x^2+y^2}$ , taking care not to cause unnecessary overflow.

#### Parameters

$X$

$X$  is DOUBLE PRECISION

$Y$

$Y$  is DOUBLE PRECISION

$X$  and  $Y$  specify the values  $x$  and  $y$ .

#### Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

#### Date

June 2017

**double precision function dlapy3 (double precision  $X$ , double precision  $Y$ , double precision  $Z$ )**

DLAPY3 returns  $\sqrt{x^2+y^2+z^2}$ .

#### Purpose:

DLAPY3 returns  $\sqrt{x^2+y^2+z^2}$ , taking care not to cause unnecessary overflow.

#### Parameters

$X$

$X$  is DOUBLE PRECISION

$Y$

$Y$  is DOUBLE PRECISION

$Z$



Z is DOUBLE PRECISION

X, Y and Z specify the values x, y and z.

#### Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

#### Date

December 2016

**subroutine dlarnv (integer IDIST, integer, dimension( 4 ) ISEED, integer N, double precision, dimension( \* ) X)**

**DLARNV** returns a vector of random numbers from a uniform or normal distribution.

#### Purpose:

DLARNV returns a vector of n random real numbers from a uniform or normal distribution.

#### Parameters

##### *IDIST*

IDIST is INTEGER

Specifies the distribution of the random numbers:

= 1: uniform (0,1)

= 2: uniform (-1,1)

= 3: normal (0,1)

##### *ISEED*

ISEED is INTEGER array, dimension (4)

On entry, the seed of the random number generator; the array elements must be between 0 and 4095, and ISEED(4) must be odd.

On exit, the seed is updated.

##### *N*

N is INTEGER

The number of random numbers to be generated.

##### *X*

X is DOUBLE PRECISION array, dimension (N)

The generated random numbers.

#### Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

#### Date

December 2016

#### Further Details:

This routine calls the auxiliary routine DLARUV to generate random real numbers from a uniform (0,1) distribution, in batches of up to 128 using vectorisable code. The Box-Muller method is used to transform numbers from a uniform to a normal distribution.



**subroutine dlarra (integer N, double precision, dimension( \* ) D, double precision, dimension( \* ) E, double precision, dimension( \* ) E2, double precision SPLTOL, double precision TNRM, integer NSPLIT, integer, dimension( \* ) ISPLIT, integer INFO)**  
**DLARRA** computes the splitting points with the specified threshold.

**Purpose:**

Compute the splitting points with threshold SPLTOL.  
 DLARRA sets any "small" off-diagonal elements to zero.

**Parameters**

*N*

*N* is INTEGER  
 The order of the matrix.  $N > 0$ .

*D*

*D* is DOUBLE PRECISION array, dimension (N)  
 On entry, the N diagonal elements of the tridiagonal matrix T.

*E*

*E* is DOUBLE PRECISION array, dimension (N)  
 On entry, the first (N-1) entries contain the subdiagonal elements of the tridiagonal matrix T; *E*(N) need not be set.  
 On exit, the entries *E*( ISPLIT( I ) ),  $1 \leq I \leq \text{NSPLIT}$ , are set to zero, the other entries of *E* are untouched.

*E2*

*E2* is DOUBLE PRECISION array, dimension (N)  
 On entry, the first (N-1) entries contain the SQUARES of the subdiagonal elements of the tridiagonal matrix T;  
*E2*(N) need not be set.  
 On exit, the entries *E2*( ISPLIT( I ) ),  $1 \leq I \leq \text{NSPLIT}$ , have been set to zero

*SPLTOL*

*SPLTOL* is DOUBLE PRECISION  
 The threshold for splitting. Two criteria can be used:  
*SPLTOL*<0 : criterion based on absolute off-diagonal value  
*SPLTOL*>0 : criterion that preserves relative accuracy

*TNRM*

*TNRM* is DOUBLE PRECISION  
 The norm of the matrix.

*NSPLIT*

*NSPLIT* is INTEGER  
 The number of blocks T splits into.  $1 \leq \text{NSPLIT} \leq N$ .

*ISPLIT*

*ISPLIT* is INTEGER array, dimension (N)  
 The splitting points, at which T breaks up into blocks.  
 The first block consists of rows/columns 1 to *ISPLIT*(1),  
 the second of rows/columns *ISPLIT*(1)+1 through *ISPLIT*(2),  
 etc., and the *NSPLIT*-th consists of rows/columns  
*ISPLIT*(*NSPLIT*-1)+1 through *ISPLIT*(*NSPLIT*)=N.

*INFO*

*INFO* is INTEGER  
 = 0: successful exit



**Author**

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

**Date**

June 2017

**Contributors:**

Beresford Parlett, University of California, Berkeley, USA

Jim Demmel, University of California, Berkeley, USA

Inderjit Dhillon, University of Texas, Austin, USA

Osni Marques, LBNL/NERSC, USA

Christof Voemel, University of California, Berkeley, USA

**subroutine dlarrb (integer N, double precision, dimension( \* ) D, double precision, dimension( \* ) LLD, integer IFIRST, integer ILAST, double precision RTOL1, double precision RTOL2, integer OFFSET, double precision, dimension( \* ) W, double precision, dimension( \* ) WGAP, double precision, dimension( \* ) WERR, double precision, dimension( \* ) WORK, integer, dimension( \* ) IWORK, double precision PIVMIN, double precision SPDIAM, integer TWIST, integer INFO)**  
**DLARRB** provides limited bisection to locate eigenvalues for more accuracy.

**Purpose:**

Given the relatively robust representation(RRR)  $L D L^T$ , **DLARRB** does "limited" bisection to refine the eigenvalues of  $L D L^T$ ,  $W( \text{IFIRST-OFFSET} )$  through  $W( \text{ILAST-OFFSET} )$ , to more accuracy. Initial guesses for these eigenvalues are input in  $W$ , the corresponding estimate of the error in these guesses and their gaps are input in  $WERR$  and  $WGAP$ , respectively. During bisection, intervals [left, right] are maintained by storing their mid-points and semi-widths in the arrays  $W$  and  $WERR$  respectively.

**Parameters***N**N* is INTEGER

The order of the matrix.

*D**D* is DOUBLE PRECISION array, dimension (N)The N diagonal elements of the diagonal matrix *D*.*LLD**LLD* is DOUBLE PRECISION array, dimension (N-1)The (N-1) elements  $L(i)*L(i)*D(i)$ .*IFIRST**IFIRST* is INTEGER

The index of the first eigenvalue to be computed.

*ILAST**ILAST* is INTEGER

The index of the last eigenvalue to be computed.

*RTOL1**RTOL1* is DOUBLE PRECISION*RTOL2**RTOL2* is DOUBLE PRECISION

Tolerance for the convergence of the bisection intervals.



An interval [LEFT,RIGHT] has converged if  
 $\text{RIGHT} - \text{LEFT} < \text{MAX}(\text{RTOL1} * \text{GAP}, \text{RTOL2} * \text{MAX}(|\text{LEFT}|, |\text{RIGHT}|))$   
 where GAP is the (estimated) distance to the nearest  
 eigenvalue.

**OFFSET**

OFFSET is INTEGER

Offset for the arrays W, WGAP and WERR, i.e., the IFIRST-OFFSET  
 through ILAST-OFFSET elements of these arrays are to be used.

**W**

W is DOUBLE PRECISION array, dimension (N)

On input, W( IFIRST-OFFSET ) through W( ILAST-OFFSET ) are  
 estimates of the eigenvalues of  $L D L^T$  indexed IFIRST through  
 ILAST.

On output, these estimates are refined.

**WGAP**

WGAP is DOUBLE PRECISION array, dimension (N-1)

On input, the (estimated) gaps between consecutive  
 eigenvalues of  $L D L^T$ , i.e.,  $\text{WGAP}(I - \text{OFFSET})$  is the gap between  
 eigenvalues I and I+1. Note that if IFIRST = ILAST  
 then  $\text{WGAP}(\text{IFIRST} - \text{OFFSET})$  must be set to ZERO.

On output, these gaps are refined.

**WERR**

WERR is DOUBLE PRECISION array, dimension (N)

On input, WERR( IFIRST-OFFSET ) through WERR( ILAST-OFFSET ) are  
 the errors in the estimates of the corresponding elements in W.

On output, these errors are refined.

**WORK**

WORK is DOUBLE PRECISION array, dimension (2\*N)

Workspace.

**IWORK**

IWORK is INTEGER array, dimension (2\*N)

Workspace.

**PIVMIN**

PIVMIN is DOUBLE PRECISION

The minimum pivot in the Sturm sequence.

**SPDIAM**

SPDIAM is DOUBLE PRECISION

The spectral diameter of the matrix.

**TWIST**

TWIST is INTEGER

The twist index for the twisted factorization that is used  
 for the negcount.

TWIST = N: Compute negcount from  $L D L^T - \text{LAMBDA } I = L + D + L^T$

TWIST = 1: Compute negcount from  $L D L^T - \text{LAMBDA } I = U - D - U^T$

TWIST = R: Compute negcount from  $L D L^T - \text{LAMBDA } I = N(r) D(r) N(r)$

**INFO**

INFO is INTEGER

Error flag.

**Author**

Univ. of Tennessee



Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

#### Date

June 2017

#### Contributors:

Beresford Parlett, University of California, Berkeley, USA

Jim Demmel, University of California, Berkeley, USA

Inderjit Dhillon, University of Texas, Austin, USA

Osni Marques, LBNL/NERSC, USA

Christof Voemel, University of California, Berkeley, USA

**subroutine dlarrc (character JOBT, integer N, double precision VL, double precision VU, double precision, dimension( \*) D, double precision, dimension( \*) E, double precision PIVMIN, integer EIGCNT, integer LCNT, integer RCNT, integer INFO)**

**DLARRC** computes the number of eigenvalues of the symmetric tridiagonal matrix.

#### Purpose:

Find the number of eigenvalues of the symmetric tridiagonal matrix  $T$  that are in the interval  $(VL, VU]$  if  $JOBT = 'T'$ , and of  $L D L^T$  if  $JOBT = 'L'$ .

#### Parameters

##### *JOBT*

$JOBT$  is CHARACTER\*1

= 'T': Compute Sturm count for matrix  $T$ .

= 'L': Compute Sturm count for matrix  $L D L^T$ .

##### *N*

$N$  is INTEGER

The order of the matrix.  $N > 0$ .

##### *VL*

$VL$  is DOUBLE PRECISION

The lower bound for the eigenvalues.

##### *VU*

$VU$  is DOUBLE PRECISION

The upper bound for the eigenvalues.

##### *D*

$D$  is DOUBLE PRECISION array, dimension ( $N$ )

$JOBT = 'T'$ : The  $N$  diagonal elements of the tridiagonal matrix  $T$ .

$JOBT = 'L'$ : The  $N$  diagonal elements of the diagonal matrix  $D$ .

##### *E*

$E$  is DOUBLE PRECISION array, dimension ( $N$ )

$JOBT = 'T'$ : The  $N-1$  offdiagonal elements of the matrix  $T$ .

$JOBT = 'L'$ : The  $N-1$  offdiagonal elements of the matrix  $L$ .

##### *PIVMIN*

$PIVMIN$  is DOUBLE PRECISION

The minimum pivot in the Sturm sequence for  $T$ .

##### *EIGCNT*

$EIGCNT$  is INTEGER

The number of eigenvalues of the symmetric tridiagonal matrix  $T$  that are in the interval  $(VL, VU]$



*LCNT*

LCNT is INTEGER

*RCNT*

RCNT is INTEGER

The left and right negcounts of the interval.

*INFO*

INFO is INTEGER

**Author**

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

**Date**

June 2016

**Contributors:**

Beresford Parlett, University of California, Berkeley, USA

Jim Demmel, University of California, Berkeley, USA

Inderjit Dhillon, University of Texas, Austin, USA

Osni Marques, LBNL/NERSC, USA

Christof Voemel, University of California, Berkeley, USA

**subroutine dlarrrd (character RANGE, character ORDER, integer N, double precision VL, double precision VU, integer IL, integer IU, double precision, dimension( \* ) GERS, double precision RELTOL, double precision, dimension( \* ) D, double precision, dimension( \* ) E, double precision, dimension( \* ) E2, double precision PIVMIN, integer NSPLIT, integer, dimension( \* ) ISPLIT, integer M, double precision, dimension( \* ) W, double precision, dimension( \* ) WERR, double precision WL, double precision WU, integer, dimension( \* ) IBLOCK, integer, dimension( \* ) INDEXW, double precision, dimension( \* ) WORK, integer, dimension( \* ) IWORK, integer INFO)**

**DLARRD** computes the eigenvalues of a symmetric tridiagonal matrix to suitable accuracy.

**Purpose:**

DLARRD computes the eigenvalues of a symmetric tridiagonal matrix T to suitable accuracy. This is an auxiliary code to be called from DSTEMR.

The user may ask for all eigenvalues, all eigenvalues in the half-open interval (VL, VU], or the IL-th through IU-th eigenvalues.

To avoid overflow, the matrix must be scaled so that its largest element is no greater than  $\text{overflow}^{**}(1/2) * \text{underflow}^{**}(1/4)$  in absolute value, and for greatest accuracy, it should not be much smaller than that.

See W. Kahan "Accurate Eigenvalues of a Symmetric Tridiagonal Matrix", Report CS41, Computer Science Dept., Stanford University, July 21, 1966.

**Parameters***RANGE*

RANGE is CHARACTER\*1

= 'A': ("All") all eigenvalues will be found.

= 'V': ("Value") all eigenvalues in the half-open interval (VL, VU] will be found.

= 'I': ("Index") the IL-th through IU-th eigenvalues (of the





entire matrix) will be found.

#### ORDER

ORDER is CHARACTER\*1

= 'B': ("By Block") the eigenvalues will be grouped by split-off block (see IBLOCK, ISPLIT) and ordered from smallest to largest within the block.

= 'E': ("Entire matrix") the eigenvalues for the entire matrix will be ordered from smallest to largest.

#### N

N is INTEGER

The order of the tridiagonal matrix T.  $N \geq 0$ .

#### VL

VL is DOUBLE PRECISION

If RANGE='V', the lower bound of the interval to be searched for eigenvalues. Eigenvalues less than or equal to VL, or greater than VU, will not be returned.  $VL < VU$ . Not referenced if RANGE = 'A' or 'I'.

#### VU

VU is DOUBLE PRECISION

If RANGE='V', the upper bound of the interval to be searched for eigenvalues. Eigenvalues less than or equal to VL, or greater than VU, will not be returned.  $VL < VU$ . Not referenced if RANGE = 'A' or 'I'.

#### IL

IL is INTEGER

If RANGE='I', the index of the smallest eigenvalue to be returned.  
 $1 \leq IL \leq IU \leq N$ , if  $N > 0$ ;  $IL = 1$  and  $IU = 0$  if  $N = 0$ .  
 Not referenced if RANGE = 'A' or 'V'.

#### IU

IU is INTEGER

If RANGE='I', the index of the largest eigenvalue to be returned.  
 $1 \leq IL \leq IU \leq N$ , if  $N > 0$ ;  $IL = 1$  and  $IU = 0$  if  $N = 0$ .  
 Not referenced if RANGE = 'A' or 'V'.

#### GERS

GERS is DOUBLE PRECISION array, dimension (2\*N)

The N Gerschgorin intervals (the i-th Gerschgorin interval is (GERS(2\*i-1), GERS(2\*i))).

#### RELTOL

RELTOL is DOUBLE PRECISION

The minimum relative width of an interval. When an interval is narrower than RELTOL times the larger (in magnitude) endpoint, then it is considered to be sufficiently small, i.e., converged. Note: this should always be at least radix\*machine epsilon.

#### D

D is DOUBLE PRECISION array, dimension (N)



The  $n$  diagonal elements of the tridiagonal matrix  $T$ .

*E*

$E$  is DOUBLE PRECISION array, dimension  $(N-1)$

The  $(n-1)$  off-diagonal elements of the tridiagonal matrix  $T$ .

*E2*

$E2$  is DOUBLE PRECISION array, dimension  $(N-1)$

The  $(n-1)$  squared off-diagonal elements of the tridiagonal matrix  $T$ .

*PIVMIN*

PIVMIN is DOUBLE PRECISION

The minimum pivot allowed in the Sturm sequence for  $T$ .

*NSPLIT*

NSPLIT is INTEGER

The number of diagonal blocks in the matrix  $T$ .

$1 \leq \text{NSPLIT} \leq N$ .

*ISPLIT*

ISPLIT is INTEGER array, dimension  $(N)$

The splitting points, at which  $T$  breaks up into submatrices.

The first submatrix consists of rows/columns 1 to ISPLIT(1),

the second of rows/columns ISPLIT(1)+1 through ISPLIT(2),

etc., and the NSPLIT-th consists of rows/columns

ISPLIT(NSPLIT-1)+1 through ISPLIT(NSPLIT)= $N$ .

(Only the first NSPLIT elements will actually be used, but since the user cannot know a priori what value NSPLIT will have,  $N$  words must be reserved for ISPLIT.)

*M*

$M$  is INTEGER

The actual number of eigenvalues found.  $0 \leq M \leq N$ .

(See also the description of INFO=2,3.)

*W*

$W$  is DOUBLE PRECISION array, dimension  $(N)$

On exit, the first  $M$  elements of  $W$  will contain the

eigenvalue approximations. DLARRD computes an interval

$I_j = (a_j, b_j]$  that includes eigenvalue  $j$ . The eigenvalue

approximation is given as the interval midpoint

$W(j) = (a_j + b_j)/2$ . The corresponding error is bounded by

$WERR(j) = \text{abs}(a_j - b_j)/2$

*WERR*

WERR is DOUBLE PRECISION array, dimension  $(N)$

The error bound on the corresponding eigenvalue approximation in  $W$ .

*WL*

WL is DOUBLE PRECISION

*WU*

WU is DOUBLE PRECISION

The interval  $(WL, WU]$  contains all the wanted eigenvalues.

If RANGE='V', then  $WL=VL$  and  $WU=VU$ .

If RANGE='A', then  $WL$  and  $WU$  are the global Gerschgorin bounds on the spectrum.

If RANGE='I', then  $WL$  and  $WU$  are computed by DLAEBZ from the index range specified.



**IBLOCK**

IBLOCK is INTEGER array, dimension (N)  
 At each row/column  $j$  where  $E(j)$  is zero or small, the matrix  $T$  is considered to split into a block diagonal matrix. On exit, if  $INFO = 0$ , IBLOCK( $i$ ) specifies to which block (from 1 to the number of blocks) the eigenvalue  $W(i)$  belongs. (DLARRD may use the remaining  $N-M$  elements as workspace.)

**INDEXW**

INDEXW is INTEGER array, dimension (N)  
 The indices of the eigenvalues within each block (submatrix); for example, INDEXW( $i$ ) =  $j$  and IBLOCK( $i$ ) =  $k$  imply that the  $i$ -th eigenvalue  $W(i)$  is the  $j$ -th eigenvalue in block  $k$ .

**WORK**

WORK is DOUBLE PRECISION array, dimension (4\*N)

**IWORK**

IWORK is INTEGER array, dimension (3\*N)

**INFO**

INFO is INTEGER  
 = 0: successful exit  
 < 0: if  $INFO = -i$ , the  $i$ -th argument had an illegal value  
 > 0: some or all of the eigenvalues failed to converge or were not computed:  
   = 1 or 3: Bisection failed to converge for some eigenvalues; these eigenvalues are flagged by a negative block number. The effect is that the eigenvalues may not be as accurate as the absolute and relative tolerances. This is generally caused by unexpectedly inaccurate arithmetic.  
   = 2 or 3: RANGE='I' only: Not all of the eigenvalues IL:IU were found.  
     Effect:  $M < IU + 1 - IL$   
     Cause: non-monotonic arithmetic, causing the Sturm sequence to be non-monotonic.  
     Cure: recalculate, using RANGE='A', and pick out eigenvalues IL:IU. In some cases, increasing the PARAMETER "FUDGE" may make things work.  
   = 4: RANGE='I', and the Gershgorin interval initially used was too small. No eigenvalues were computed.  
     Probable cause: your machine has sloppy floating-point arithmetic.  
     Cure: Increase the PARAMETER "FUDGE", recompile, and try again.

**Internal Parameters:**

FUDGE DOUBLE PRECISION, default = 2

A "fudge factor" to widen the Gershgorin intervals. Ideally, a value of 1 should work, but on machines with sloppy arithmetic, this needs to be larger. The default for publicly released versions should be large enough to handle the worst machine around. Note that this has no effect



on accuracy of the solution.

#### Contributors:

W. Kahan, University of California, Berkeley, USA  
 Beresford Parlett, University of California, Berkeley, USA  
 Jim Demmel, University of California, Berkeley, USA  
 Inderjit Dhillon, University of Texas, Austin, USA  
 Osni Marques, LBNL/NERSC, USA  
 Christof Voemel, University of California, Berkeley, USA

#### Author

Univ. of Tennessee  
 Univ. of California Berkeley  
 Univ. of Colorado Denver  
 NAG Ltd.

#### Date

June 2016

**subroutine dlarre** (character **RANGE**, integer **N**, double precision **VL**, double precision **VU**, integer **IL**, integer **IU**, double precision, dimension( \*) **D**, double precision, dimension( \*) **E**, double precision, dimension( \*) **E2**, double precision **RTOL1**, double precision **RTOL2**, double precision **SPLTOL**, integer **NSPLIT**, integer, dimension( \*) **ISPLIT**, integer **M**, double precision, dimension( \*) **W**, double precision, dimension( \*) **WERR**, double precision, dimension( \*) **WGAP**, integer, dimension( \*) **IBLOCK**, integer, dimension( \*) **INDEXW**, double precision, dimension( \*) **GERS**, double precision **PIVMIN**, double precision, dimension( \*) **WORK**, integer, dimension( \*) **IWORK**, integer **INFO**)

**DLARRE** given the tridiagonal matrix **T**, sets small off-diagonal elements to zero and for each unreduced block **T<sub>i</sub>**, finds base representations and eigenvalues.

#### Purpose:

To find the desired eigenvalues of a given real symmetric tridiagonal matrix **T**, **DLARRE** sets any "small" off-diagonal elements to zero, and for each unreduced block **T<sub>i</sub>**, it finds  
 (a) a suitable shift at one end of the block's spectrum,  
 (b) the base representation,  $T_i - \sigma_i I = L_i D_i L_i^T$ , and  
 (c) eigenvalues of each  $L_i D_i L_i^T$ .

The representations and eigenvalues found are then used by **DSTEMR** to compute the eigenvectors of **T**.

The accuracy varies depending on whether bisection is used to find a few eigenvalues or the dqds algorithm (subroutine **DLASQ2**) to compute all and then discard any unwanted one.

As an added benefit, **DLARRE** also outputs the *n* Gerschgorin intervals for the matrices  $L_i D_i L_i^T$ .

#### Parameters

##### *RANGE*

**RANGE** is CHARACTER\*1  
 = 'A': ("All") all eigenvalues will be found.  
 = 'V': ("Value") all eigenvalues in the half-open interval (VL, VU] will be found.  
 = 'I': ("Index") the IL-th through IU-th eigenvalues (of the entire matrix) will be found.

##### *N*

**N** is INTEGER  
 The order of the matrix.  $N > 0$ .

##### *VL*

**VL** is DOUBLE PRECISION



If RANGE='V', the lower bound for the eigenvalues.  
Eigenvalues less than or equal to VL, or greater than VU,  
will not be returned.  $VL < VU$ .  
If RANGE='I' or 'A', DLARRE computes bounds on the desired  
part of the spectrum.

*VU*

VU is DOUBLE PRECISION  
If RANGE='V', the upper bound for the eigenvalues.  
Eigenvalues less than or equal to VL, or greater than VU,  
will not be returned.  $VL < VU$ .  
If RANGE='I' or 'A', DLARRE computes bounds on the desired  
part of the spectrum.

*IL*

IL is INTEGER  
If RANGE='I', the index of the  
smallest eigenvalue to be returned.  
 $1 \leq IL \leq IU \leq N$ .

*IU*

IU is INTEGER  
If RANGE='I', the index of the  
largest eigenvalue to be returned.  
 $1 \leq IL \leq IU \leq N$ .

*D*

D is DOUBLE PRECISION array, dimension (N)  
On entry, the N diagonal elements of the tridiagonal  
matrix T.  
On exit, the N diagonal elements of the diagonal  
matrices D\_i.

*E*

E is DOUBLE PRECISION array, dimension (N)  
On entry, the first (N-1) entries contain the subdiagonal  
elements of the tridiagonal matrix T; E(N) need not be set.  
On exit, E contains the subdiagonal elements of the unit  
bidiagonal matrices L\_i. The entries E( ISPLIT( I ) ),  
 $1 \leq I \leq NSPLIT$ , contain the base points sigma\_i on output.

*E2*

E2 is DOUBLE PRECISION array, dimension (N)  
On entry, the first (N-1) entries contain the SQUARES of the  
subdiagonal elements of the tridiagonal matrix T;  
E2(N) need not be set.  
On exit, the entries E2( ISPLIT( I ) ),  
 $1 \leq I \leq NSPLIT$ , have been set to zero

*RTOL1*

RTOL1 is DOUBLE PRECISION

*RTOL2*

RTOL2 is DOUBLE PRECISION  
Parameters for bisection.  
An interval [LEFT,RIGHT] has converged if  
 $RIGHT-LEFT < \text{MAX}(RTOL1 * \text{GAP}, RTOL2 * \text{MAX}(|LEFT|, |RIGHT|))$

*SPLTOL*

SPLTOL is DOUBLE PRECISION



The threshold for splitting.

### *NSPLIT*

NSPLIT is INTEGER

The number of blocks T splits into.  $1 \leq \text{NSPLIT} \leq N$ .

### *ISPLIT*

ISPLIT is INTEGER array, dimension (N)

The splitting points, at which T breaks up into blocks.

The first block consists of rows/columns 1 to ISPLIT(1), the second of rows/columns ISPLIT(1)+1 through ISPLIT(2), etc., and the NSPLIT-th consists of rows/columns ISPLIT(NSPLIT-1)+1 through ISPLIT(NSPLIT)=N.

### *M*

M is INTEGER

The total number of eigenvalues (of all  $L_i D_i L_i^T$ ) found.

### *W*

W is DOUBLE PRECISION array, dimension (N)

The first M elements contain the eigenvalues. The eigenvalues of each of the blocks,  $L_i D_i L_i^T$ , are sorted in ascending order (DLARRE may use the remaining N-M elements as workspace).

### *WERR*

WERR is DOUBLE PRECISION array, dimension (N)

The error bound on the corresponding eigenvalue in W.

### *WGAP*

WGAP is DOUBLE PRECISION array, dimension (N)

The separation from the right neighbor eigenvalue in W.

The gap is only with respect to the eigenvalues of the same block as each block has its own representation tree.

Exception: at the right end of a block we store the left gap

### *IBLOCK*

IBLOCK is INTEGER array, dimension (N)

The indices of the blocks (submatrices) associated with the corresponding eigenvalues in W; IBLOCK(i)=1 if eigenvalue W(i) belongs to the first block from the top, =2 if W(i) belongs to the second block, etc.

### *INDEXW*

INDEXW is INTEGER array, dimension (N)

The indices of the eigenvalues within each block (submatrix); for example, INDEXW(i)= 10 and IBLOCK(i)=2 imply that the i-th eigenvalue W(i) is the 10-th eigenvalue in block 2

### *GRS*

GRS is DOUBLE PRECISION array, dimension (2\*N)

The N Gerschgorin intervals (the i-th Gerschgorin interval is (GRS(2\*i-1), GRS(2\*i)).

### *PIVMIN*

PIVMIN is DOUBLE PRECISION

The minimum pivot in the Sturm sequence for T.

### *WORK*

WORK is DOUBLE PRECISION array, dimension (6\*N)



Workspace.

#### *IWORK*

IWORK is INTEGER array, dimension (5\*N)  
Workspace.

#### *INFO*

INFO is INTEGER

= 0: successful exit

> 0: A problem occurred in DLARRE.

< 0: One of the called subroutines signaled an internal problem.  
Needs inspection of the corresponding parameter IINFO  
for further information.

--1: Problem in DLARRD.

= 2: No base representation could be found in MAXTRY iterations.  
Increasing MAXTRY and recompilation might be a remedy.

--3: Problem in DLARRB when computing the refined root  
representation for DLASQ2.

--4: Problem in DLARRB when performing bisection on the  
desired part of the spectrum.

--5: Problem in DLASQ2.

--6: Problem in DLASQ2.

#### **Author**

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

#### **Date**

June 2016

#### **Further Details:**

The base representations are required to suffer very little  
element growth and consequently define all their eigenvalues to  
high relative accuracy.

#### **Contributors:**

Beresford Parlett, University of California, Berkeley, USA

Jim Demmel, University of California, Berkeley, USA

Inderjit Dhillon, University of Texas, Austin, USA

Osni Marques, LBNL/NERSC, USA

Christof Voemel, University of California, Berkeley, USA

**subroutine dlarrf (integer N, double precision, dimension( \* ) D, double precision, dimension( \* ) L,  
double precision, dimension( \* ) LD, integer CLSTRT, integer CLEND, double precision,  
dimension( \* ) W, double precision, dimension( \* ) WGAP, double precision, dimension( \* )  
WERR, double precision SPDIAM, double precision CLGAPL, double precision CLGAPR,  
double precision PIVMIN, double precision SIGMA, double precision, dimension( \* ) DPLUS,  
double precision, dimension( \* ) LPLUS, double precision, dimension( \* ) WORK, integer INFO)**  
DLARRF finds a new relatively robust representation such that at least one of the eigenvalues is  
relatively isolated.

#### **Purpose:**

Given the initial representation  $L D L^T$  and its cluster of close  
eigenvalues (in a relative measure),  $W( CLSTRT )$ ,  $W( CLSTRT+1 )$ , ...  
 $W( CLEND )$ , DLARRF finds a new relatively robust representation  
 $L D L^T - SIGMA I = L(+) D(+) L(+)^T$  such that at least one of the



eigenvalues of  $L(+) D(+) L(+)^T$  is relatively isolated.

### Parameters

*N*

*N* is INTEGER

The order of the matrix (subblock, if the matrix split).

*D*

*D* is DOUBLE PRECISION array, dimension (*N*)

The *N* diagonal elements of the diagonal matrix *D*.

*L*

*L* is DOUBLE PRECISION array, dimension (*N*-1)

The (*N*-1) subdiagonal elements of the unit bidiagonal matrix *L*.

*LD*

*LD* is DOUBLE PRECISION array, dimension (*N*-1)

The (*N*-1) elements  $L(i)*D(i)$ .

*CLSTRT*

*CLSTRT* is INTEGER

The index of the first eigenvalue in the cluster.

*CLEND*

*CLEND* is INTEGER

The index of the last eigenvalue in the cluster.

*W*

*W* is DOUBLE PRECISION array, dimension

dimension is  $\geq (CLEND-CLSTRT+1)$

The eigenvalue APPROXIMATIONS of  $L D L^T$  in ascending order.

$W( CLSTRT )$  through  $W( CLEND )$  form the cluster of relatively close eigenvalues.

*WGAP*

*WGAP* is DOUBLE PRECISION array, dimension

dimension is  $\geq (CLEND-CLSTRT+1)$

The separation from the right neighbor eigenvalue in *W*.

*WERR*

*WERR* is DOUBLE PRECISION array, dimension

dimension is  $\geq (CLEND-CLSTRT+1)$

*WERR* contain the semiwidth of the uncertainty

interval of the corresponding eigenvalue APPROXIMATION in *W*

*SPDIAM*

*SPDIAM* is DOUBLE PRECISION

estimate of the spectral diameter obtained from the

Gerschgorin intervals

*CLGAPL*

*CLGAPL* is DOUBLE PRECISION

*CLGAPR*

*CLGAPR* is DOUBLE PRECISION

absolute gap on each end of the cluster.

Set by the calling routine to protect against shifts too close to eigenvalues outside the cluster.

*PIVMIN*





PIVMIN is DOUBLE PRECISION

The minimum pivot allowed in the Sturm sequence.

#### *SIGMA*

SIGMA is DOUBLE PRECISION

The shift used to form  $L(+) D(+) L(+)^T$ .

#### *DPLUS*

DPLUS is DOUBLE PRECISION array, dimension (N)

The N diagonal elements of the diagonal matrix  $D(+)$ .

#### *LPLUS*

LPLUS is DOUBLE PRECISION array, dimension (N-1)

The first (N-1) elements of LPLUS contain the subdiagonal elements of the unit bidiagonal matrix  $L(+)$ .

#### *WORK*

WORK is DOUBLE PRECISION array, dimension (2\*N)

Workspace.

#### *INFO*

INFO is INTEGER

Signals processing OK (=0) or failure (=1)

#### **Author**

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

#### **Date**

June 2016

#### **Contributors:**

Beresford Parlett, University of California, Berkeley, USA

Jim Demmel, University of California, Berkeley, USA

Inderjit Dhillon, University of Texas, Austin, USA

Osni Marques, LBNL/NERSC, USA

Christof Voemel, University of California, Berkeley, USA

**subroutine dlarrj (integer N, double precision, dimension( \* ) D, double precision, dimension( \* ) E2, integer IFIRST, integer ILAST, double precision RTOL, integer OFFSET, double precision, dimension( \* ) W, double precision, dimension( \* ) WERR, double precision, dimension( \* ) WORK, integer, dimension( \* ) IWORK, double precision PIVMIN, double precision SPDIA, integer INFO)**

**DLARRJ** performs refinement of the initial estimates of the eigenvalues of the matrix T.

#### **Purpose:**

Given the initial eigenvalue approximations of T, DLARRJ

does bisection to refine the eigenvalues of T,

$W( \text{IFIRST-OFFSET} )$  through  $W( \text{ILAST-OFFSET} )$ , to more accuracy. Initial

guesses for these eigenvalues are input in W, the corresponding estimate

of the error in these guesses in WERR. During bisection, intervals

[left, right] are maintained by storing their mid-points and

semi-widths in the arrays W and WERR respectively.

#### **Parameters**

*N*

N is INTEGER

The order of the matrix.



*D*

*D* is DOUBLE PRECISION array, dimension (N)  
The N diagonal elements of *T*.

*E2*

*E2* is DOUBLE PRECISION array, dimension (N-1)  
The Squares of the (N-1) subdiagonal elements of *T*.

*IFIRST*

*IFIRST* is INTEGER  
The index of the first eigenvalue to be computed.

*ILAST*

*ILAST* is INTEGER  
The index of the last eigenvalue to be computed.

*RTOL*

*RTOL* is DOUBLE PRECISION  
Tolerance for the convergence of the bisection intervals.  
An interval [LEFT,RIGHT] has converged if  
 $\text{RIGHT} - \text{LEFT} < \text{RTOL} * \text{MAX}(|\text{LEFT}|, |\text{RIGHT}|)$ .

*OFFSET*

*OFFSET* is INTEGER  
Offset for the arrays *W* and *WERR*, i.e., the *IFIRST*-*OFFSET*  
through *ILAST*-*OFFSET* elements of these arrays are to be used.

*W*

*W* is DOUBLE PRECISION array, dimension (N)  
On input, *W*( *IFIRST*-*OFFSET* ) through *W*( *ILAST*-*OFFSET* ) are  
estimates of the eigenvalues of  $L D L^T$  indexed *IFIRST* through  
*ILAST*.  
On output, these estimates are refined.

*WERR*

*WERR* is DOUBLE PRECISION array, dimension (N)  
On input, *WERR*( *IFIRST*-*OFFSET* ) through *WERR*( *ILAST*-*OFFSET* ) are  
the errors in the estimates of the corresponding elements in *W*.  
On output, these errors are refined.

*WORK*

*WORK* is DOUBLE PRECISION array, dimension (2\*N)  
Workspace.

*IWORK*

*IWORK* is INTEGER array, dimension (2\*N)  
Workspace.

*PIVMIN*

*PIVMIN* is DOUBLE PRECISION  
The minimum pivot in the Sturm sequence for *T*.

*SPDIAM*

*SPDIAM* is DOUBLE PRECISION  
The spectral diameter of *T*.

*INFO*

*INFO* is INTEGER  
Error flag.

**Author**



Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

#### Date

June 2017

#### Contributors:

Beresford Parlett, University of California, Berkeley, USA

Jim Demmel, University of California, Berkeley, USA

Inderjit Dhillon, University of Texas, Austin, USA

Osni Marques, LBNL/NERSC, USA

Christof Voemel, University of California, Berkeley, USA

**subroutine dlarrk (integer N, integer IW, double precision GL, double precision GU, double precision, dimension( \* ) D, double precision, dimension( \* ) E2, double precision PIVMIN, double precision RELTOL, double precision W, double precision WERR, integer INFO)**  
**DLARRK** computes one eigenvalue of a symmetric tridiagonal matrix T to suitable accuracy.

#### Purpose:

DLARRK computes one eigenvalue of a symmetric tridiagonal matrix T to suitable accuracy. This is an auxiliary code to be called from DSTEMR.

To avoid overflow, the matrix must be scaled so that its largest element is no greater than  $\text{overflow}^{**}(1/2) * \text{underflow}^{**}(1/4)$  in absolute value, and for greatest accuracy, it should not be much smaller than that.

See W. Kahan "Accurate Eigenvalues of a Symmetric Tridiagonal Matrix", Report CS41, Computer Science Dept., Stanford University, July 21, 1966.

#### Parameters

*N*

N is INTEGER

The order of the tridiagonal matrix T.  $N \geq 0$ .

*IW*

IW is INTEGER

The index of the eigenvalues to be returned.

*GL*

GL is DOUBLE PRECISION

*GU*

GU is DOUBLE PRECISION

An upper and a lower bound on the eigenvalue.

*D*

D is DOUBLE PRECISION array, dimension (N)

The n diagonal elements of the tridiagonal matrix T.

*E2*

E2 is DOUBLE PRECISION array, dimension (N-1)

The (n-1) squared off-diagonal elements of the tridiagonal matrix T.

*PIVMIN*

PIVMIN is DOUBLE PRECISION

The minimum pivot allowed in the Sturm sequence for T.



*RELTOL*

RELTOL is DOUBLE PRECISION

The minimum relative width of an interval. When an interval is narrower than RELTOL times the larger (in magnitude) endpoint, then it is considered to be sufficiently small, i.e., converged. Note: this should always be at least radix\*machine epsilon.

*W*

W is DOUBLE PRECISION

*WERR*

WERR is DOUBLE PRECISION

The error bound on the corresponding eigenvalue approximation in W.

*INFO*

INFO is INTEGER

= 0: Eigenvalue converged

= -1: Eigenvalue did NOT converge

**Internal Parameters:**

FUDGE DOUBLE PRECISION, default = 2

A "fudge factor" to widen the Gershgorin intervals.

**Author**

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

**Date**

June 2017

**subroutine dlarr (integer N, double precision, dimension( \* ) D, double precision, dimension( \* ) E, integer INFO)**

**DLARRR** performs tests to decide whether the symmetric tridiagonal matrix T warrants expensive computations which guarantee high relative accuracy in the eigenvalues.

**Purpose:**

Perform tests to decide whether the symmetric tridiagonal matrix T warrants expensive computations which guarantee high relative accuracy in the eigenvalues.

**Parameters***N*

N is INTEGER

The order of the matrix.  $N > 0$ .

*D*

D is DOUBLE PRECISION array, dimension (N)

The N diagonal elements of the tridiagonal matrix T.

*E*

E is DOUBLE PRECISION array, dimension (N)

On entry, the first (N-1) entries contain the subdiagonal elements of the tridiagonal matrix T; E(N) is set to ZERO.

*INFO*

INFO is INTEGER

INFO = 0(default) : the matrix warrants computations preserving relative accuracy.

INFO = 1 : the matrix warrants computations guaranteeing only absolute accuracy.

#### Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

#### Date

June 2017

#### Contributors:

Beresford Parlett, University of California, Berkeley, USA

Jim Demmel, University of California, Berkeley, USA

Inderjit Dhillon, University of Texas, Austin, USA

Osni Marques, LBNL/NERSC, USA

Christof Voemel, University of California, Berkeley, USA

**subroutine dlartg (double precision F, double precision G, double precision CS, double precision SN, double precision R)**

**DLARTG** generates a plane rotation with real cosine and real sine.

#### Purpose:

DLARTG generate a plane rotation so that

$$\begin{bmatrix} CS & SN \\ -SN & CS \end{bmatrix} \cdot \begin{bmatrix} F \\ G \end{bmatrix} = \begin{bmatrix} R \\ 0 \end{bmatrix} \quad \text{where } CS^2 + SN^2 = 1.$$

This is a slower, more accurate version of the BLAS1 routine DROTG, with the following other differences:

F and G are unchanged on return.

If G=0, then CS=1 and SN=0.

If F=0 and (G .ne. 0), then CS=0 and SN=1 without doing any floating point operations (saves work in DBDSQR when there are zeros on the diagonal).

If F exceeds G in magnitude, CS will be positive.

#### Parameters

*F*

F is DOUBLE PRECISION

The first component of vector to be rotated.

*G*

G is DOUBLE PRECISION

The second component of vector to be rotated.

*CS*

CS is DOUBLE PRECISION

The cosine of the rotation.

*SN*

SN is DOUBLE PRECISION

The sine of the rotation.

*R*



R is DOUBLE PRECISION

The nonzero component of the rotated vector.

This version has a few statements commented out for thread safety  
(machine parameters are computed on each entry). 10 feb 03, SJH.

#### Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

#### Date

December 2016

**subroutine dlartgp (double precision F, double precision G, double precision CS, double precision SN, double precision R)**

**DLARTGP** generates a plane rotation so that the diagonal is nonnegative.

#### Purpose:

DLARTGP generates a plane rotation so that

$$\begin{bmatrix} CS & SN \\ -SN & CS \end{bmatrix} \cdot \begin{bmatrix} F \\ G \end{bmatrix} = \begin{bmatrix} R \\ 0 \end{bmatrix} \quad \text{where } CS^2 + SN^2 = 1.$$

This is a slower, more accurate version of the Level 1 BLAS routine DROTG,  
with the following other differences:

F and G are unchanged on return.

If G=0, then CS=(+/-)1 and SN=0.

If F=0 and (G .ne. 0), then CS=0 and SN=(+/-)1.

The sign is chosen so that R >= 0.

#### Parameters

*F*

F is DOUBLE PRECISION

The first component of vector to be rotated.

*G*

G is DOUBLE PRECISION

The second component of vector to be rotated.

*CS*

CS is DOUBLE PRECISION

The cosine of the rotation.

*SN*

SN is DOUBLE PRECISION

The sine of the rotation.

*R*

R is DOUBLE PRECISION

The nonzero component of the rotated vector.

This version has a few statements commented out for thread safety  
(machine parameters are computed on each entry). 10 feb 03, SJH.

#### Author

Univ. of Tennessee

Univ. of California Berkeley



Univ. of Colorado Denver

NAG Ltd.

#### Date

December 2016

**subroutine dlaruv (integer, dimension( 4 ) ISEED, integer N, double precision, dimension( n ) X)**

**DLARUV** returns a vector of n random real numbers from a uniform distribution.

#### Purpose:

DLARUV returns a vector of n random real numbers from a uniform (0,1) distribution (n <= 128).

This is an auxiliary routine called by DLARNV and ZLARNV.

#### Parameters

##### *ISEED*

ISEED is INTEGER array, dimension (4)

On entry, the seed of the random number generator; the array elements must be between 0 and 4095, and ISEED(4) must be odd.

On exit, the seed is updated.

##### *N*

N is INTEGER

The number of random numbers to be generated. N <= 128.

##### *X*

X is DOUBLE PRECISION array, dimension (N)

The generated random numbers.

#### Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

#### Date

December 2016

#### Further Details:

This routine uses a multiplicative congruential method with modulus  $2^{*}48$  and multiplier 33952834046453 (see G.S.Fishman, 'Multiplicative congruential random number generators with modulus  $2^{*}b$ : an exhaustive analysis for  $b = 32$  and a partial analysis for  $b = 48$ ', Math. Comp. 189, pp 331-344, 1990).

48-bit integers are stored in 4 integer array elements with 12 bits per element. Hence the routine is portable across machines with integers of 32 bits or more.

**subroutine dlas2 (double precision F, double precision G, double precision H, double precision SSMIN, double precision SSMAX)**

**DLAS2** computes singular values of a 2-by-2 triangular matrix.

#### Purpose:

DLAS2 computes the singular values of the 2-by-2 matrix

[ F G ]



[ 0 H ].

On return, SSMIN is the smaller singular value and SSMAX is the larger singular value.

#### Parameters

*F*

*F* is DOUBLE PRECISION

The (1,1) element of the 2-by-2 matrix.

*G*

*G* is DOUBLE PRECISION

The (1,2) element of the 2-by-2 matrix.

*H*

*H* is DOUBLE PRECISION

The (2,2) element of the 2-by-2 matrix.

*SSMIN*

*SSMIN* is DOUBLE PRECISION

The smaller singular value.

*SSMAX*

*SSMAX* is DOUBLE PRECISION

The larger singular value.

#### Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

#### Date

December 2016

#### Further Details:

Barring over/underflow, all output quantities are correct to within a few units in the last place (ulps), even in the absence of a guard digit in addition/subtraction.

In IEEE arithmetic, the code works correctly if one matrix element is infinite.

Overflow will not occur unless the largest singular value itself overflows, or is within a few ulps of overflow. (On machines with partial overflow, like the Cray, overflow may occur if the largest singular value is within a factor of 2 of overflow.)

Underflow is harmless if underflow is gradual. Otherwise, results may correspond to a matrix modified by perturbations of size near the underflow threshold.

**subroutine dlascl (character TYPE, integer KL, integer KU, double precision CFROM, double precision CTO, integer M, integer N, double precision, dimension( lda, \* ) A, integer LDA, integer INFO)**

**DLASCL** multiplies a general rectangular matrix by a real scalar defined as cto/cfrom.

#### Purpose:

DLASCL multiplies the M by N real matrix A by the real scalar CTO/CFROM. This is done without over/underflow as long as the final





result  $CTO * A(I,J) / CFROM$  does not over/underflow. TYPE specifies that A may be full, upper triangular, lower triangular, upper Hessenberg, or banded.

### Parameters

#### TYPE

TYPE is CHARACTER\*1

TYPE indices the storage type of the input matrix.

= 'G': A is a full matrix.

= 'L': A is a lower triangular matrix.

= 'U': A is an upper triangular matrix.

= 'H': A is an upper Hessenberg matrix.

= 'B': A is a symmetric band matrix with lower bandwidth KL and upper bandwidth KU and with the only the lower half stored.

= 'Q': A is a symmetric band matrix with lower bandwidth KL and upper bandwidth KU and with the only the upper half stored.

= 'Z': A is a band matrix with lower bandwidth KL and upper bandwidth KU. See DGBTRF for storage details.

#### KL

KL is INTEGER

The lower bandwidth of A. Referenced only if TYPE = 'B', 'Q' or 'Z'.

#### KU

KU is INTEGER

The upper bandwidth of A. Referenced only if TYPE = 'B', 'Q' or 'Z'.

#### CFROM

CFROM is DOUBLE PRECISION

#### CTO

CTO is DOUBLE PRECISION

The matrix A is multiplied by CTO/CFROM.  $A(I,J)$  is computed without over/underflow if the final result  $CTO * A(I,J) / CFROM$  can be represented without over/underflow. CFROM must be nonzero.

#### M

M is INTEGER

The number of rows of the matrix A.  $M \geq 0$ .

#### N

N is INTEGER

The number of columns of the matrix A.  $N \geq 0$ .

#### A

A is DOUBLE PRECISION array, dimension (LDA,N)

The matrix to be multiplied by CTO/CFROM. See TYPE for the storage type.

#### LDA

LDA is INTEGER

The leading dimension of the array A.

If TYPE = 'G', 'L', 'U', 'H',  $LDA \geq \max(1,M)$ ;

TYPE = 'B',  $LDA \geq KL+1$ ;



```
TYPE = 'Q', LDA >= KU+1;
TYPE = 'Z', LDA >= 2*KL+KU+1.
```

**INFO**

INFO is INTEGER

0 - successful exit

<0 - if INFO = -i, the i-th argument had an illegal value.

**Author**

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

**Date**

June 2016

**subroutine dlasd0 (integer N, integer SQRE, double precision, dimension( \* ) D, double precision, dimension( \* ) E, double precision, dimension( ldu, \* ) U, integer LDU, double precision, dimension( ldvt, \* ) VT, integer LDVT, integer SMLSIZ, integer, dimension( \* ) IWORK, double precision, dimension( \* ) WORK, integer INFO)**

**DLASD0** computes the singular values of a real upper bidiagonal n-by-m matrix B with diagonal d and off-diagonal e. Used by sbdsdc.

**Purpose:**

Using a divide and conquer approach, DLASD0 computes the singular value decomposition (SVD) of a real upper bidiagonal N-by-M matrix B with diagonal D and offdiagonal E, where  $M = N + SQRE$ . The algorithm computes orthogonal matrices U and VT such that  $B = U * S * VT$ . The singular values S are overwritten on D.

A related subroutine, DLASDA, computes only the singular values, and optionally, the singular vectors in compact form.

**Parameters**

*N*

N is INTEGER

On entry, the row dimension of the upper bidiagonal matrix.

This is also the dimension of the main diagonal array D.

*SQRE*

SQRE is INTEGER

Specifies the column dimension of the bidiagonal matrix.

= 0: The bidiagonal matrix has column dimension  $M = N$ ;

= 1: The bidiagonal matrix has column dimension  $M = N+1$ ;

*D*

D is DOUBLE PRECISION array, dimension (N)

On entry D contains the main diagonal of the bidiagonal matrix.

On exit D, if INFO = 0, contains its singular values.

*E*

E is DOUBLE PRECISION array, dimension (M-1)

Contains the subdiagonal entries of the bidiagonal matrix.

On exit, E has been destroyed.

*U*

U is DOUBLE PRECISION array, dimension (LDU, N)

On exit, U contains the left singular vectors.



*LDU*

LDU is INTEGER  
On entry, leading dimension of U.

*VT*

VT is DOUBLE PRECISION array, dimension (LDVT, M)  
On exit, VT\*\*T contains the right singular vectors.

*LDVT*

LDVT is INTEGER  
On entry, leading dimension of VT.

*SMLSIZ*

SMLSIZ is INTEGER  
On entry, maximum size of the subproblems at the bottom of the computation tree.

*IWORK*

IWORK is INTEGER array, dimension (8\*N)

*WORK*

WORK is DOUBLE PRECISION array, dimension (3\*M\*\*2+2\*M)

*INFO*

INFO is INTEGER  
= 0: successful exit.  
< 0: if INFO = -i, the i-th argument had an illegal value.  
> 0: if INFO = 1, a singular value did not converge

**Author**

Univ. of Tennessee  
Univ. of California Berkeley  
Univ. of Colorado Denver  
NAG Ltd.

**Date**

June 2017

**Contributors:**

Ming Gu and Huan Ren, Computer Science Division, University of California at Berkeley, USA

**subroutine dlasd1 (integer NL, integer NR, integer SQRE, double precision, dimension( \* ) D, double precision ALPHA, double precision BETA, double precision, dimension( ldu, \* ) U, integer LDU, double precision, dimension( ldvt, \* ) VT, integer LDVT, integer, dimension( \* ) IDXQ, integer, dimension( \* ) IWORK, double precision, dimension( \* ) WORK, integer INFO)**

**DLASD1** computes the SVD of an upper bidiagonal matrix B of the specified size. Used by sbdsdc.

**Purpose:**

DLASD1 computes the SVD of an upper bidiagonal N-by-M matrix B, where N = NL + NR + 1 and M = N + SQRE. DLASD1 is called from DLASD0.

A related subroutine DLASD7 handles the case in which the singular values (and the singular vectors in factored form) are desired.

DLASD1 computes the SVD as follows:

$$B = U(\text{in}) * \begin{pmatrix} D1(\text{in}) & 0 & 0 & 0 \\ Z1^{**T} & a & Z2^{**T} & b \\ 0 & 0 & D2(\text{in}) & 0 \end{pmatrix} * VT(\text{in})$$



$$= U(out) * ( D(out) 0 ) * VT(out)$$

where  $Z^{**T} = (Z1^{**T} \text{ a } Z2^{**T} \text{ b}) = u^{**T} VT^{**T}$ , and  $u$  is a vector of dimension  $M$  with  $ALPHA$  and  $BETA$  in the  $NL+1$  and  $NL+2$  th entries and zeros elsewhere; and the entry  $b$  is empty if  $SQRE = 0$ .

The left singular vectors of the original matrix are stored in  $U$ , and the transpose of the right singular vectors are stored in  $VT$ , and the singular values are in  $D$ . The algorithm consists of three stages:

The first stage consists of deflating the size of the problem when there are multiple singular values or when there are zeros in the  $Z$  vector. For each such occurrence the dimension of the secular equation problem is reduced by one. This stage is performed by the routine `DLASD2`.

The second stage consists of calculating the updated singular values. This is done by finding the square roots of the roots of the secular equation via the routine `DLASD4` (as called by `DLASD3`). This routine also calculates the singular vectors of the current problem.

The final stage consists of computing the updated singular vectors directly using the updated singular values. The singular vectors for the current problem are multiplied with the singular vectors from the overall problem.

### Parameters

*NL*

$NL$  is INTEGER

The row dimension of the upper block.  $NL \geq 1$ .

*NR*

$NR$  is INTEGER

The row dimension of the lower block.  $NR \geq 1$ .

*SQRE*

$SQRE$  is INTEGER

= 0: the lower block is an  $NR$ -by- $NR$  square matrix.

= 1: the lower block is an  $NR$ -by- $(NR+1)$  rectangular matrix.

The bidiagonal matrix has row dimension  $N = NL + NR + 1$ , and column dimension  $M = N + SQRE$ .

*D*

$D$  is DOUBLE PRECISION array,  
dimension  $(N = NL+NR+1)$ .

On entry  $D(1:NL,1:NL)$  contains the singular values of the upper block; and  $D(NL+2:N)$  contains the singular values of the lower block. On exit  $D(1:N)$  contains the singular values of the modified matrix.

*ALPHA*

$ALPHA$  is DOUBLE PRECISION

Contains the diagonal element associated with the added row.

*BETA*

$BETA$  is DOUBLE PRECISION

Contains the off-diagonal element associated with the added row.



OTHERauxiliary(3)

LAPACK

OTHERauxiliary(3)

*U*

*U* is DOUBLE PRECISION array, dimension(LDU,N)  
 On entry *U*(1:NL, 1:NL) contains the left singular vectors of the upper block; *U*(NL+2:N, NL+2:N) contains the left singular vectors of the lower block. On exit *U* contains the left singular vectors of the bidiagonal matrix.

*LDU*

*LDU* is INTEGER  
 The leading dimension of the array *U*.  $LDU \geq \max(1, N)$ .

*VT*

*VT* is DOUBLE PRECISION array, dimension(LDVT,M)  
 where  $M = N + \text{SQRE}$ .  
 On entry *VT*(1:NL+1, 1:NL+1)\*\**T* contains the right singular vectors of the upper block; *VT*(NL+2:M, NL+2:M)\*\**T* contains the right singular vectors of the lower block. On exit *VT*\*\**T* contains the right singular vectors of the bidiagonal matrix.

*LDVT*

*LDVT* is INTEGER  
 The leading dimension of the array *VT*.  $LDVT \geq \max(1, M)$ .

*IDXQ*

*IDXQ* is INTEGER array, dimension(N)  
 This contains the permutation which will reintegrate the subproblem just solved back into sorted order, i.e.  
*D*( *IDXQ*( *I* = 1, *N* ) ) will be in ascending order.

*IWORK*

*IWORK* is INTEGER array, dimension( 4 \* *N* )

*WORK*

*WORK* is DOUBLE PRECISION array, dimension( 3\*M\*\*2 + 2\*M )

*INFO*

*INFO* is INTEGER  
 = 0: successful exit.  
 < 0: if *INFO* = -*i*, the *i*-th argument had an illegal value.  
 > 0: if *INFO* = 1, a singular value did not converge

**Author**

Univ. of Tennessee  
 Univ. of California Berkeley  
 Univ. of Colorado Denver  
 NAG Ltd.

**Date**

June 2016

**Contributors:**

Ming Gu and Huan Ren, Computer Science Division, University of California at Berkeley, USA

subroutine **dlasd2** (integer *NL*, integer *NR*, integer *SQRE*, integer *K*, double precision, dimension( \* ) *D*, double precision, dimension( \* ) *Z*, double precision *ALPHA*, double precision *BETA*, double precision, dimension( *ldu*, \* ) *U*, integer *LDU*, double precision, dimension( *ldvt*, \* ) *VT*, integer *LDVT*, double precision, dimension( \* ) *DSIGMA*, double precision, dimension( *ldu2*, \* ) *U2*, integer *LDU2*, double precision, dimension( *ldvt2*, \* ) *VT2*, integer *LDVT2*, integer, dimension( \* ) *IDXP*, integer, dimension( \* ) *IDX*, integer, dimension( \* ) *IDXC*, integer, dimension( \* ) *IDXQ*, integer, dimension( \* ) *COLTYP*, integer *INFO*)



**DLASD2** merges the two sets of singular values together into a single sorted set. Used by sbdsdc.

**Purpose:**

DLASD2 merges the two sets of singular values together into a single sorted set. Then it tries to deflate the size of the problem.

There are two ways in which deflation can occur: when two or more singular values are close together or if there is a tiny entry in the Z vector. For each such occurrence the order of the related secular equation problem is reduced by one.

DLASD2 is called from DLASD1.

**Parameters**

*NL*

NL is INTEGER

The row dimension of the upper block.  $NL \geq 1$ .

*NR*

NR is INTEGER

The row dimension of the lower block.  $NR \geq 1$ .

*SQRE*

SQRE is INTEGER

= 0: the lower block is an NR-by-NR square matrix.

= 1: the lower block is an NR-by-(NR+1) rectangular matrix.

The bidiagonal matrix has  $N = NL + NR + 1$  rows and

$M = N + SQRE \geq N$  columns.

*K*

K is INTEGER

Contains the dimension of the non-deflated matrix,

This is the order of the related secular equation.  $1 \leq K \leq N$ .

*D*

D is DOUBLE PRECISION array, dimension(N)

On entry D contains the singular values of the two submatrices to be combined. On exit D contains the trailing (N-K) updated singular values (those which were deflated) sorted into increasing order.

*Z*

Z is DOUBLE PRECISION array, dimension(N)

On exit Z contains the updating row vector in the secular equation.

*ALPHA*

ALPHA is DOUBLE PRECISION

Contains the diagonal element associated with the added row.

*BETA*

BETA is DOUBLE PRECISION

Contains the off-diagonal element associated with the added row.

*U*

U is DOUBLE PRECISION array, dimension(LDU,N)

On entry U contains the left singular vectors of two submatrices in the two square blocks with corners at (1,1), (NL, NL), and (NL+2, NL+2), (N,N).



On exit U contains the trailing (N-K) updated left singular vectors (those which were deflated) in its last N-K columns.

#### *LDU*

LDU is INTEGER

The leading dimension of the array U.  $LDU \geq N$ .

#### *VT*

VT is DOUBLE PRECISION array, dimension(LDVT,M)

On entry VT\*\*T contains the right singular vectors of two submatrices in the two square blocks with corners at (1,1), (NL+1, NL+1), and (NL+2, NL+2), (M,M).

On exit VT\*\*T contains the trailing (N-K) updated right singular vectors (those which were deflated) in its last N-K columns.

In case SQRE =1, the last row of VT spans the right null space.

#### *LDVT*

LDVT is INTEGER

The leading dimension of the array VT.  $LDVT \geq M$ .

#### *DSIGMA*

DSIGMA is DOUBLE PRECISION array, dimension (N)

Contains a copy of the diagonal elements (K-1 singular values and one zero) in the secular equation.

#### *U2*

U2 is DOUBLE PRECISION array, dimension(LDU2,N)

Contains a copy of the first K-1 left singular vectors which will be used by DLASD3 in a matrix multiply (DGEMM) to solve for the new left singular vectors. U2 is arranged into four blocks. The first block contains a column with 1 at NL+1 and zero everywhere else; the second block contains non-zero entries only at and above NL; the third contains non-zero entries only below NL+1; and the fourth is dense.

#### *LDU2*

LDU2 is INTEGER

The leading dimension of the array U2.  $LDU2 \geq N$ .

#### *VT2*

VT2 is DOUBLE PRECISION array, dimension(LDVT2,N)

VT2\*\*T contains a copy of the first K right singular vectors which will be used by DLASD3 in a matrix multiply (DGEMM) to solve for the new right singular vectors. VT2 is arranged into three blocks. The first block contains a row that corresponds to the special 0 diagonal element in SIGMA; the second block contains non-zeros only at and before NL +1; the third block contains non-zeros only at and after NL +2.

#### *LDVT2*

LDVT2 is INTEGER

The leading dimension of the array VT2.  $LDVT2 \geq M$ .

#### *IDXP*

IDXP is INTEGER array, dimension(N)

This will contain the permutation used to place deflated values of D at the end of the array. On output IDXP(2:K) points to the nondeflated D-values and IDXP(K+1:N) points to the deflated singular values.



*IDX*

IDX is INTEGER array, dimension(N)  
This will contain the permutation used to sort the contents of D into ascending order.

*IDXC*

IDXC is INTEGER array, dimension(N)  
This will contain the permutation used to arrange the columns of the deflated U matrix into three groups: the first group contains non-zero entries only at and above NL, the second contains non-zero entries only below NL+2, and the third is dense.

*IDXQ*

IDXQ is INTEGER array, dimension(N)  
This contains the permutation which separately sorts the two sub-problems in D into ascending order. Note that entries in the first half of this permutation must first be moved one position backward; and entries in the second half must first have NL+1 added to their values.

*COLTYP*

COLTYP is INTEGER array, dimension(N)  
As workspace, this will contain a label which will indicate which of the following types a column in the U2 matrix or a row in the VT2 matrix is:

- 1 : non-zero in the upper half only
- 2 : non-zero in the lower half only
- 3 : dense
- 4 : deflated

On exit, it is an array of dimension 4, with COLTYP(I) being the dimension of the I-th type columns.

*INFO*

INFO is INTEGER  
= 0: successful exit.  
< 0: if INFO = -i, the i-th argument had an illegal value.

**Author**

Univ. of Tennessee  
Univ. of California Berkeley  
Univ. of Colorado Denver  
NAG Ltd.

**Date**

June 2017

**Contributors:**

Ming Gu and Huan Ren, Computer Science Division, University of California at Berkeley, USA

**subroutine dlasd3 (integer NL, integer NR, integer SQRE, integer K, double precision, dimension( \* ) D, double precision, dimension( ldu, \* ) Q, integer LDQ, double precision, dimension( \* ) DSIGMA, double precision, dimension( ldu, \* ) U, integer LDU, double precision, dimension( ldu2, \* ) U2, integer LDU2, double precision, dimension( ldvt, \* ) VT, integer LDVT, double precision, dimension( ldvt2, \* ) VT2, integer LDVT2, integer, dimension( \* ) IDXC, integer, dimension( \* ) CTOT, double precision, dimension( \* ) Z, integer INFO)**

**DLASD3** finds all square roots of the roots of the secular equation, as defined by the values in D and Z, and then updates the singular vectors by matrix multiplication. Used by sbsdsc.

**Purpose:**



DLASD3 finds all the square roots of the roots of the secular equation, as defined by the values in *D* and *Z*. It makes the appropriate calls to DLASD4 and then updates the singular vectors by matrix multiplication.

This code makes very mild assumptions about floating point arithmetic. It will work on machines with a guard digit in add/subtract, or on those binary machines without guard digits which subtract like the Cray XMP, Cray YMP, Cray C 90, or Cray 2. It could conceivably fail on hexadecimal or decimal machines without guard digits, but we know of none.

DLASD3 is called from DLASD1.

### Parameters

*NL*

*NL* is INTEGER

The row dimension of the upper block.  $NL \geq 1$ .

*NR*

*NR* is INTEGER

The row dimension of the lower block.  $NR \geq 1$ .

*SQRE*

*SQRE* is INTEGER

= 0: the lower block is an *NR*-by-*NR* square matrix.

= 1: the lower block is an *NR*-by-(*NR*+1) rectangular matrix.

The bidiagonal matrix has  $N = NL + NR + 1$  rows and  $M = N + SQRE \geq N$  columns.

*K*

*K* is INTEGER

The size of the secular equation,  $1 \leq K \leq N$ .

*D*

*D* is DOUBLE PRECISION array, dimension(*K*)

On exit the square roots of the roots of the secular equation, in ascending order.

*Q*

*Q* is DOUBLE PRECISION array, dimension (LDQ,*K*)

*LDQ*

*LDQ* is INTEGER

The leading dimension of the array *Q*.  $LDQ \geq K$ .

*DSIGMA*

*DSIGMA* is DOUBLE PRECISION array, dimension(*K*)

The first *K* elements of this array contain the old roots of the deflated updating problem. These are the poles of the secular equation.

*U*

*U* is DOUBLE PRECISION array, dimension (LDU, *N*)

The last *N* - *K* columns of this matrix contain the deflated left singular vectors.

*LDU*

*LDU* is INTEGER

The leading dimension of the array *U*.  $LDU \geq N$ .



*U2*

*U2* is DOUBLE PRECISION array, dimension (*LDU2*, *N*)  
 The first *K* columns of this matrix contain the non-deflated  
 left singular vectors for the split problem.

*LDU2*

*LDU2* is INTEGER  
 The leading dimension of the array *U2*. *LDU2*  $\geq$  *N*.

*VT*

*VT* is DOUBLE PRECISION array, dimension (*LDVT*, *M*)  
 The last *M* - *K* columns of *VT\*\*T* contain the deflated  
 right singular vectors.

*LDVT*

*LDVT* is INTEGER  
 The leading dimension of the array *VT*. *LDVT*  $\geq$  *N*.

*VT2*

*VT2* is DOUBLE PRECISION array, dimension (*LDVT2*, *N*)  
 The first *K* columns of *VT2\*\*T* contain the non-deflated  
 right singular vectors for the split problem.

*LDVT2*

*LDVT2* is INTEGER  
 The leading dimension of the array *VT2*. *LDVT2*  $\geq$  *N*.

*IDXC*

*IDXC* is INTEGER array, dimension ( *N* )  
 The permutation used to arrange the columns of *U* (and rows of  
*VT*) into three groups: the first group contains non-zero  
 entries only at and above (or before) *NL* + 1; the second  
 contains non-zero entries only at and below (or after) *NL*+2;  
 and the third is dense. The first column of *U* and the row of  
*VT* are treated separately, however.

The rows of the singular vectors found by *DLASD4*  
 must be likewise permuted before the matrix multiplies can  
 take place.

*CTOT*

*CTOT* is INTEGER array, dimension ( 4 )  
 A count of the total number of the various types of columns  
 in *U* (or rows in *VT*), as described in *IDXC*. The fourth column  
 type is any column which has been deflated.

*Z*

*Z* is DOUBLE PRECISION array, dimension (*K*)  
 The first *K* elements of this array contain the components  
 of the deflation-adjusted updating row vector.

*INFO*

*INFO* is INTEGER  
 = 0: successful exit.  
 < 0: if *INFO* = -*i*, the *i*-th argument had an illegal value.  
 > 0: if *INFO* = 1, a singular value did not converge

**Author**

Univ. of Tennessee

Univ. of California Berkeley



Univ. of Colorado Denver

NAG Ltd.

#### Date

June 2017

#### Contributors:

Ming Gu and Huan Ren, Computer Science Division, University of California at Berkeley, USA

**subroutine dlasd4 (integer N, integer I, double precision, dimension( \*) D, double precision, dimension( \*) Z, double precision, dimension( \*) DELTA, double precision RHO, double precision SIGMA, double precision, dimension( \*) WORK, integer INFO)**

**DLASD4** computes the square root of the i-th updated eigenvalue of a positive symmetric rank-one modification to a positive diagonal matrix. Used by dbdsdc.

#### Purpose:

This subroutine computes the square root of the I-th updated eigenvalue of a positive symmetric rank-one modification to a positive diagonal matrix whose entries are given as the squares of the corresponding entries in the array d, and that

$$0 \leq D(i) < D(j) \text{ for } i < j$$

and that  $RHO > 0$ . This is arranged by the calling routine, and is no loss in generality. The rank-one modified system is thus

$$\text{diag}(D) * \text{diag}(D) + RHO * Z * Z_{\text{transpose}}.$$

where we assume the Euclidean norm of Z is 1.

The method consists of approximating the rational functions in the secular equation by simpler interpolating rational functions.

#### Parameters

*N*

N is INTEGER  
The length of all arrays.

*I*

I is INTEGER  
The index of the eigenvalue to be computed.  $1 \leq I \leq N$ .

*D*

D is DOUBLE PRECISION array, dimension ( N )  
The original eigenvalues. It is assumed that they are in order,  $0 \leq D(I) < D(J)$  for  $I < J$ .

*Z*

Z is DOUBLE PRECISION array, dimension ( N )  
The components of the updating vector.

*DELTA*

DELTA is DOUBLE PRECISION array, dimension ( N )  
If N .ne. 1, DELTA contains  $(D(j) - \text{sigma}_I)$  in its j-th component. If  $N = 1$ , then  $DELTA(1) = 1$ . The vector DELTA contains the information necessary to construct the (singular) eigenvectors.

*RHO*

RHO is DOUBLE PRECISION



The scalar in the symmetric updating formula.

#### *SIGMA*

SIGMA is DOUBLE PRECISION

The computed  $\sigma_I$ , the I-th updated eigenvalue.

#### *WORK*

WORK is DOUBLE PRECISION array, dimension ( N )

If N .ne. 1, WORK contains  $(D(j) + \sigma_I)$  in its j-th component. If N = 1, then  $WORK(1) = 1$ .

#### *INFO*

INFO is INTEGER

= 0: successful exit

> 0: if INFO = 1, the updating process failed.

#### **Internal Parameters:**

Logical variable ORGATI (origin-at-i?) is used for distinguishing whether D(i) or D(i+1) is treated as the origin.

ORGATI = .true. origin at i

ORGATI = .false. origin at i+1

Logical variable SWITCH3 (switch-for-3-poles?) is for noting if we are working with THREE poles!

MAXIT is the maximum number of iterations allowed for each eigenvalue.

#### **Author**

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

#### **Date**

December 2016

#### **Contributors:**

Ren-Cang Li, Computer Science Division, University of California at Berkeley, USA

**subroutine dlasd5 (integer I, double precision, dimension( 2 ) D, double precision, dimension( 2 ) Z, double precision, dimension( 2 ) DELTA, double precision RHO, double precision DSIGMA, double precision, dimension( 2 ) WORK)**

**DLASD5** computes the square root of the i-th eigenvalue of a positive symmetric rank-one modification of a 2-by-2 diagonal matrix. Used by sbsdsc.

#### **Purpose:**

This subroutine computes the square root of the I-th eigenvalue of a positive symmetric rank-one modification of a 2-by-2 diagonal matrix

$$\text{diag}(D) * \text{diag}(D) + RHO * Z * \text{transpose}(Z).$$

The diagonal entries in the array D are assumed to satisfy

$$0 \leq D(i) < D(j) \text{ for } i < j.$$

We also assume  $RHO > 0$  and that the Euclidean norm of the vector



*Z* is one.

### Parameters

*I*

*I* is INTEGER

The index of the eigenvalue to be computed.  $I = 1$  or  $I = 2$ .

*D*

*D* is DOUBLE PRECISION array, dimension ( 2 )

The original eigenvalues. We assume  $0 \leq D(1) < D(2)$ .

*Z*

*Z* is DOUBLE PRECISION array, dimension ( 2 )

The components of the updating vector.

*DELTA*

*DELTA* is DOUBLE PRECISION array, dimension ( 2 )

Contains  $(D(j) - \sigma_I)$  in its  $j$ -th component.

The vector *DELTA* contains the information necessary to construct the eigenvectors.

*RHO*

*RHO* is DOUBLE PRECISION

The scalar in the symmetric updating formula.

*DSIGMA*

*DSIGMA* is DOUBLE PRECISION

The computed  $\sigma_I$ , the  $I$ -th updated eigenvalue.

*WORK*

*WORK* is DOUBLE PRECISION array, dimension ( 2 )

*WORK* contains  $(D(j) + \sigma_I)$  in its  $j$ -th component.

### Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

### Date

December 2016

### Contributors:

Ren-Cang Li, Computer Science Division, University of California at Berkeley, USA

**subroutine dlasd6 (integer ICOMPQ, integer NL, integer NR, integer SQRE, double precision, dimension( \*) D, double precision, dimension( \*) VF, double precision, dimension( \*) VL, double precision ALPHA, double precision BETA, integer, dimension( \*) IDXQ, integer, dimension( \*) PERM, integer GIVPTR, integer, dimension( ldgcol, \*) GIVCOL, integer LDGCOL, double precision, dimension( ldgnum, \*) GIVNUM, integer LDGNUM, double precision, dimension( ldgnum, \*) POLES, double precision, dimension( \*) DIFL, double precision, dimension( \*) DIFR, double precision, dimension( \*) Z, integer K, double precision C, double precision S, double precision, dimension( \*) WORK, integer, dimension( \*) IWORK, integer INFO)**

**DLASD6** computes the SVD of an updated upper bidiagonal matrix obtained by merging two smaller ones by appending a row. Used by sbsdsc.

### Purpose:

**DLASD6** computes the SVD of an updated upper bidiagonal matrix *B* obtained by merging two smaller ones by appending a row. This



routine is used only for the problem which requires all singular values and optionally singular vector matrices in factored form. B is an N-by-M matrix with  $N = NL + NR + 1$  and  $M = N + SQRE$ . A related subroutine, DLASD1, handles the case in which all singular values and singular vectors of the bidiagonal matrix are desired.

DLASD6 computes the SVD as follows:

$$B = U(\text{in}) * \begin{pmatrix} D1(\text{in}) & 0 & 0 & 0 \\ Z1^{**T} & a & Z2^{**T} & b \\ 0 & 0 & D2(\text{in}) & 0 \end{pmatrix} * VT(\text{in})$$

$$= U(\text{out}) * \begin{pmatrix} D(\text{out}) & 0 \end{pmatrix} * VT(\text{out})$$

where  $Z^{**T} = (Z1^{**T} \ a \ Z2^{**T} \ b) = u^{**T} \ VT^{**T}$ , and u is a vector of dimension M with ALPHA and BETA in the NL+1 and NL+2 th entries and zeros elsewhere; and the entry b is empty if  $SQRE = 0$ .

The singular values of B can be computed using D1, D2, the first components of all the right singular vectors of the lower block, and the last components of all the right singular vectors of the upper block. These components are stored and updated in VF and VL, respectively, in DLASD6. Hence U and VT are not explicitly referenced.

The singular values are stored in D. The algorithm consists of two stages:

The first stage consists of deflating the size of the problem when there are multiple singular values or if there is a zero in the Z vector. For each such occurrence the dimension of the secular equation problem is reduced by one. This stage is performed by the routine DLASD7.

The second stage consists of calculating the updated singular values. This is done by finding the roots of the secular equation via the routine DLASD4 (as called by DLASD8). This routine also updates VF and VL and computes the distances between the updated singular values and the old singular values.

DLASD6 is called from DLASDA.

### Parameters

#### *ICOMPQ*

ICOMPQ is INTEGER

Specifies whether singular vectors are to be computed in factored form:

= 0: Compute singular values only.

= 1: Compute singular vectors in factored form as well.

#### *NL*

NL is INTEGER

The row dimension of the upper block.  $NL \geq 1$ .

#### *NR*

NR is INTEGER

The row dimension of the lower block.  $NR \geq 1$ .

#### *SQRE*



SQRE is INTEGER

= 0: the lower block is an NR-by-NR square matrix.

= 1: the lower block is an NR-by-(NR+1) rectangular matrix.

The bidiagonal matrix has row dimension  $N = NL + NR + 1$ ,  
and column dimension  $M = N + SQRE$ .

#### *D*

D is DOUBLE PRECISION array, dimension (  $NL+NR+1$  ).

On entry D(1:NL,1:NL) contains the singular values of the upper block, and D(NL+2:N) contains the singular values of the lower block. On exit D(1:N) contains the singular values of the modified matrix.

#### *VF*

VF is DOUBLE PRECISION array, dimension (  $M$  )

On entry, VF(1:NL+1) contains the first components of all right singular vectors of the upper block; and VF(NL+2:M) contains the first components of all right singular vectors of the lower block. On exit, VF contains the first components of all right singular vectors of the bidiagonal matrix.

#### *VL*

VL is DOUBLE PRECISION array, dimension (  $M$  )

On entry, VL(1:NL+1) contains the last components of all right singular vectors of the upper block; and VL(NL+2:M) contains the last components of all right singular vectors of the lower block. On exit, VL contains the last components of all right singular vectors of the bidiagonal matrix.

#### *ALPHA*

ALPHA is DOUBLE PRECISION

Contains the diagonal element associated with the added row.

#### *BETA*

BETA is DOUBLE PRECISION

Contains the off-diagonal element associated with the added row.

#### *IDXQ*

IDXQ is INTEGER array, dimension (  $N$  )

This contains the permutation which will reintegrate the subproblem just solved back into sorted order, i.e.

D(  $IDXQ(I = 1, N)$  ) will be in ascending order.

#### *PERM*

PERM is INTEGER array, dimension (  $N$  )

The permutations (from deflation and sorting) to be applied to each block. Not referenced if ICOMPQ = 0.

#### *GIVPTR*

GIVPTR is INTEGER

The number of Givens rotations which took place in this subproblem. Not referenced if ICOMPQ = 0.

#### *GIVCOL*

GIVCOL is INTEGER array, dimension (  $LDGCOL, 2$  )

Each pair of numbers indicates a pair of columns to take place in a Givens rotation. Not referenced if ICOMPQ = 0.

#### *LDGCOL*



LDGCOL is INTEGER

leading dimension of GIVCOL, must be at least N.

#### *GIVNUM*

GIVNUM is DOUBLE PRECISION array, dimension ( LDGNUM, 2 )

Each number indicates the C or S value to be used in the corresponding Givens rotation. Not referenced if ICOMPQ = 0.

#### *LDGNUM*

LDGNUM is INTEGER

The leading dimension of GIVNUM and POLES, must be at least N.

#### *POLES*

POLES is DOUBLE PRECISION array, dimension ( LDGNUM, 2 )

On exit, POLES(1,\*) is an array containing the new singular values obtained from solving the secular equation, and POLES(2,\*) is an array containing the poles in the secular equation. Not referenced if ICOMPQ = 0.

#### *DIFL*

DIFL is DOUBLE PRECISION array, dimension ( N )

On exit, DIFL(I) is the distance between I-th updated (undeflated) singular value and the I-th (undeflated) old singular value.

#### *DIFR*

DIFR is DOUBLE PRECISION array,  
dimension ( LDDIFR, 2 ) if ICOMPQ = 1 and  
dimension ( K ) if ICOMPQ = 0.

On exit, DIFR(I,1) = D(I) - DSIGMA(I+1), DIFR(K,1) is not defined and will not be referenced.

If ICOMPQ = 1, DIFR(1:K,2) is an array containing the normalizing factors for the right singular vector matrix.

See DLASD8 for details on DIFL and DIFR.

#### *Z*

Z is DOUBLE PRECISION array, dimension ( M )

The first elements of this array contain the components of the deflation-adjusted updating row vector.

#### *K*

K is INTEGER

Contains the dimension of the non-deflated matrix,  
This is the order of the related secular equation.  $1 \leq K \leq N$ .

#### *C*

C is DOUBLE PRECISION

C contains garbage if SQRE = 0 and the C-value of a Givens rotation related to the right null space if SQRE = 1.

#### *S*

S is DOUBLE PRECISION

S contains garbage if SQRE = 0 and the S-value of a Givens rotation related to the right null space if SQRE = 1.

#### *WORK*

WORK is DOUBLE PRECISION array, dimension ( 4 \* M )

#### *IWORK*





IWORK is INTEGER array, dimension ( 3 \* N )

#### INFO

INFO is INTEGER

= 0: successful exit.

< 0: if INFO = -i, the i-th argument had an illegal value.

> 0: if INFO = 1, a singular value did not converge

#### Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

#### Date

June 2016

#### Contributors:

Ming Gu and Huan Ren, Computer Science Division, University of California at Berkeley, USA

**subroutine dlasd7 (integer ICOMPQ, integer NL, integer NR, integer SQRE, integer K, double precision, dimension( \* ) D, double precision, dimension( \* ) Z, double precision, dimension( \* ) ZW, double precision, dimension( \* ) VF, double precision, dimension( \* ) VFW, double precision, dimension( \* ) VL, double precision, dimension( \* ) VLW, double precision ALPHA, double precision BETA, double precision, dimension( \* ) DSIGMA, integer, dimension( \* ) IDX, integer, dimension( \* ) IDXP, integer, dimension( \* ) IDXQ, integer, dimension( \* ) PERM, integer GIVPTR, integer, dimension( ldgcol, \* ) GIVCOL, integer LDGCOL, double precision, dimension( ldgnum, \* ) GIVNUM, integer LDGNUM, double precision C, double precision S, integer INFO)**

**DLASD7** merges the two sets of singular values together into a single sorted set. Then it tries to deflate the size of the problem. Used by sbdsdc.

#### Purpose:

DLASD7 merges the two sets of singular values together into a single sorted set. Then it tries to deflate the size of the problem. There are two ways in which deflation can occur: when two or more singular values are close together or if there is a tiny entry in the Z vector. For each such occurrence the order of the related secular equation problem is reduced by one.

DLASD7 is called from DLASD6.

#### Parameters

##### ICOMPQ

ICOMPQ is INTEGER

Specifies whether singular vectors are to be computed in compact form, as follows:

= 0: Compute singular values only.

= 1: Compute singular vectors of upper bidiagonal matrix in compact form.

##### NL

NL is INTEGER

The row dimension of the upper block. NL  $\geq$  1.

##### NR

NR is INTEGER

The row dimension of the lower block. NR  $\geq$  1.

##### SQRE



SQRE is INTEGER

= 0: the lower block is an NR-by-NR square matrix.

= 1: the lower block is an NR-by-(NR+1) rectangular matrix.

The bidiagonal matrix has

$N = NL + NR + 1$  rows and

$M = N + SQRE \geq N$  columns.

*K*

K is INTEGER

Contains the dimension of the non-deflated matrix, this is the order of the related secular equation.  $1 \leq K \leq N$ .

*D*

D is DOUBLE PRECISION array, dimension ( N )

On entry D contains the singular values of the two submatrices to be combined. On exit D contains the trailing (N-K) updated singular values (those which were deflated) sorted into increasing order.

*Z*

Z is DOUBLE PRECISION array, dimension ( M )

On exit Z contains the updating row vector in the secular equation.

*ZW*

ZW is DOUBLE PRECISION array, dimension ( M )

Workspace for Z.

*VF*

VF is DOUBLE PRECISION array, dimension ( M )

On entry, VF(1:NL+1) contains the first components of all right singular vectors of the upper block; and VF(NL+2:M) contains the first components of all right singular vectors of the lower block. On exit, VF contains the first components of all right singular vectors of the bidiagonal matrix.

*VFW*

VFW is DOUBLE PRECISION array, dimension ( M )

Workspace for VF.

*VL*

VL is DOUBLE PRECISION array, dimension ( M )

On entry, VL(1:NL+1) contains the last components of all right singular vectors of the upper block; and VL(NL+2:M) contains the last components of all right singular vectors of the lower block. On exit, VL contains the last components of all right singular vectors of the bidiagonal matrix.

*VLW*

VLW is DOUBLE PRECISION array, dimension ( M )

Workspace for VL.

*ALPHA*

ALPHA is DOUBLE PRECISION

Contains the diagonal element associated with the added row.

*BETA*

BETA is DOUBLE PRECISION

Contains the off-diagonal element associated with the added row.



*DSIGMA*

DSIGMA is DOUBLE PRECISION array, dimension ( N )  
Contains a copy of the diagonal elements (K-1 singular values and one zero) in the secular equation.

*IDX*

IDX is INTEGER array, dimension ( N )  
This will contain the permutation used to sort the contents of D into ascending order.

*IDXP*

IDXP is INTEGER array, dimension ( N )  
This will contain the permutation used to place deflated values of D at the end of the array. On output IDXP(2:K) points to the nondeflated D-values and IDXP(K+1:N) points to the deflated singular values.

*IDXQ*

IDXQ is INTEGER array, dimension ( N )  
This contains the permutation which separately sorts the two sub-problems in D into ascending order. Note that entries in the first half of this permutation must first be moved one position backward; and entries in the second half must first have NL+1 added to their values.

*PERM*

PERM is INTEGER array, dimension ( N )  
The permutations (from deflation and sorting) to be applied to each singular block. Not referenced if ICOMPQ = 0.

*GIVPTR*

GIVPTR is INTEGER  
The number of Givens rotations which took place in this subproblem. Not referenced if ICOMPQ = 0.

*GIVCOL*

GIVCOL is INTEGER array, dimension ( LDGCOL, 2 )  
Each pair of numbers indicates a pair of columns to take place in a Givens rotation. Not referenced if ICOMPQ = 0.

*LDGCOL*

LDGCOL is INTEGER  
The leading dimension of GIVCOL, must be at least N.

*GIVNUM*

GIVNUM is DOUBLE PRECISION array, dimension ( LDGNUM, 2 )  
Each number indicates the C or S value to be used in the corresponding Givens rotation. Not referenced if ICOMPQ = 0.

*LDGNUM*

LDGNUM is INTEGER  
The leading dimension of GIVNUM, must be at least N.

*C*

C is DOUBLE PRECISION  
C contains garbage if SQRE = 0 and the C-value of a Givens rotation related to the right null space if SQRE = 1.

*S*

S is DOUBLE PRECISION



S contains garbage if SQRE = 0 and the S-value of a Givens rotation related to the right null space if SQRE = 1.

#### INFO

INFO is INTEGER

= 0: successful exit.

< 0: if INFO = -i, the i-th argument had an illegal value.

#### Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

#### Date

December 2016

#### Contributors:

Ming Gu and Huan Ren, Computer Science Division, University of California at Berkeley, USA

**subroutine dlasd8 (integer ICOMPQ, integer K, double precision, dimension( \* ) D, double precision, dimension( \* ) Z, double precision, dimension( \* ) VF, double precision, dimension( \* ) VL, double precision, dimension( \* ) DIFL, double precision, dimension( lddifr, \* ) DIFR, integer LDDIFR, double precision, dimension( \* ) DSIGMA, double precision, dimension( \* ) WORK, integer INFO)**

**DLASD8** finds the square roots of the roots of the secular equation, and stores, for each element in D, the distance to its two nearest poles. Used by sbdsdc.

#### Purpose:

DLASD8 finds the square roots of the roots of the secular equation, as defined by the values in DSIGMA and Z. It makes the appropriate calls to DLASD4, and stores, for each element in D, the distance to its two nearest poles (elements in DSIGMA). It also updates the arrays VF and VL, the first and last components of all the right singular vectors of the original bidiagonal matrix.

DLASD8 is called from DLASD6.

#### Parameters

##### ICOMPQ

ICOMPQ is INTEGER

Specifies whether singular vectors are to be computed in factored form in the calling routine:

= 0: Compute singular values only.

= 1: Compute singular vectors in factored form as well.

##### K

K is INTEGER

The number of terms in the rational function to be solved by DLASD4.  $K \geq 1$ .

##### D

D is DOUBLE PRECISION array, dimension ( K )

On output, D contains the updated singular values.

##### Z

Z is DOUBLE PRECISION array, dimension ( K )

On entry, the first K elements of this array contain the components of the deflation-adjusted updating row vector.

On exit, Z is updated.



***VF***

VF is DOUBLE PRECISION array, dimension ( K )  
 On entry, VF contains information passed through DBEDE8.  
 On exit, VF contains the first K components of the first components of all right singular vectors of the bidiagonal matrix.

***VL***

VL is DOUBLE PRECISION array, dimension ( K )  
 On entry, VL contains information passed through DBEDE8.  
 On exit, VL contains the first K components of the last components of all right singular vectors of the bidiagonal matrix.

***DIFL***

DIFL is DOUBLE PRECISION array, dimension ( K )  
 On exit,  $DIFL(I) = D(I) - DSIGMA(I)$ .

***DIFR***

DIFR is DOUBLE PRECISION array,  
 dimension ( LDDIFR, 2 ) if ICOMPQ = 1 and  
 dimension ( K ) if ICOMPQ = 0.  
 On exit,  $DIFR(I,1) = D(I) - DSIGMA(I+1)$ ,  $DIFR(K,1)$  is not defined and will not be referenced.

If ICOMPQ = 1,  $DIFR(1:K,2)$  is an array containing the normalizing factors for the right singular vector matrix.

***LDDIFR***

LDDIFR is INTEGER  
 The leading dimension of DIFR, must be at least K.

***DSIGMA***

DSIGMA is DOUBLE PRECISION array, dimension ( K )  
 On entry, the first K elements of this array contain the old roots of the deflated updating problem. These are the poles of the secular equation.  
 On exit, the elements of DSIGMA may be very slightly altered in value.

***WORK***

WORK is DOUBLE PRECISION array, dimension (3\*K)

***INFO***

INFO is INTEGER  
 = 0: successful exit.  
 < 0: if  $INFO = -i$ , the i-th argument had an illegal value.  
 > 0: if  $INFO = 1$ , a singular value did not converge

**Author**

Univ. of Tennessee  
 Univ. of California Berkeley  
 Univ. of Colorado Denver  
 NAG Ltd.

**Date**

June 2017

**Contributors:**

Ming Gu and Huan Ren, Computer Science Division, University of California at Berkeley, USA



**subroutine dlasda** (integer ICOMPQ, integer SMLSIZ, integer N, integer SQRE, double precision, dimension( \* ) D, double precision, dimension( \* ) E, double precision, dimension( ldu, \* ) U, integer LDU, double precision, dimension( ldu, \* ) VT, integer, dimension( \* ) K, double precision, dimension( ldu, \* ) DIFL, double precision, dimension( ldu, \* ) DIFR, double precision, dimension( ldu, \* ) Z, double precision, dimension( ldu, \* ) POLES, integer, dimension( \* ) GIVPTR, integer, dimension( ldgcol, \* ) GIVCOL, integer LDGCOL, integer, dimension( ldgcol, \* ) PERM, double precision, dimension( ldu, \* ) GIVNUM, double precision, dimension( \* ) C, double precision, dimension( \* ) S, double precision, dimension( \* ) WORK, integer, dimension( \* ) IWORK, integer INFO)

**DLASDA** computes the singular value decomposition (SVD) of a real upper bidiagonal matrix with diagonal d and off-diagonal e. Used by sbdsdc.

#### **Purpose:**

Using a divide and conquer approach, DLASDA computes the singular value decomposition (SVD) of a real upper bidiagonal N-by-M matrix B with diagonal D and offdiagonal E, where  $M = N + SQRE$ . The algorithm computes the singular values in the SVD  $B = U * S * VT$ . The orthogonal matrices U and VT are optionally computed in compact form.

A related subroutine, DLASD0, computes the singular values and the singular vectors in explicit form.

#### **Parameters**

##### *ICOMPQ*

ICOMPQ is INTEGER

Specifies whether singular vectors are to be computed in compact form, as follows

= 0: Compute singular values only.

= 1: Compute singular vectors of upper bidiagonal matrix in compact form.

##### *SMLSIZ*

SMLSIZ is INTEGER

The maximum size of the subproblems at the bottom of the computation tree.

##### *N*

N is INTEGER

The row dimension of the upper bidiagonal matrix. This is also the dimension of the main diagonal array D.

##### *SQRE*

SQRE is INTEGER

Specifies the column dimension of the bidiagonal matrix.

= 0: The bidiagonal matrix has column dimension  $M = N$ ;

= 1: The bidiagonal matrix has column dimension  $M = N + 1$ .

##### *D*

D is DOUBLE PRECISION array, dimension ( N )

On entry D contains the main diagonal of the bidiagonal matrix. On exit D, if INFO = 0, contains its singular values.

##### *E*

E is DOUBLE PRECISION array, dimension ( M-1 )

Contains the subdiagonal entries of the bidiagonal matrix.

On exit, E has been destroyed.

##### *U*



U is DOUBLE PRECISION array,  
dimension ( LDU, SMLSIZ ) if ICOMPQ = 1, and not referenced  
if ICOMPQ = 0. If ICOMPQ = 1, on exit, U contains the left  
singular vector matrices of all subproblems at the bottom  
level.

#### LDU

LDU is INTEGER, LDU = > N.  
The leading dimension of arrays U, VT, DIFL, DIFR, POLES,  
GIVNUM, and Z.

#### VT

VT is DOUBLE PRECISION array,  
dimension ( LDU, SMLSIZ+1 ) if ICOMPQ = 1, and not referenced  
if ICOMPQ = 0. If ICOMPQ = 1, on exit, VT\*\*T contains the right  
singular vector matrices of all subproblems at the bottom  
level.

#### K

K is INTEGER array,  
dimension ( N ) if ICOMPQ = 1 and dimension 1 if ICOMPQ = 0.  
If ICOMPQ = 1, on exit, K(I) is the dimension of the I-th  
secular equation on the computation tree.

#### DIFL

DIFL is DOUBLE PRECISION array, dimension ( LDU, NLVL ),  
where NLVL = floor(log<sub>2</sub> (N/SMLSIZ)).

#### DIFR

DIFR is DOUBLE PRECISION array,  
dimension ( LDU, 2 \* NLVL ) if ICOMPQ = 1 and  
dimension ( N ) if ICOMPQ = 0.  
If ICOMPQ = 1, on exit, DIFL(1:N, I) and DIFR(1:N, 2 \* I - 1)  
record distances between singular values on the I-th  
level and singular values on the (I - 1)-th level, and  
DIFR(1:N, 2 \* I) contains the normalizing factors for  
the right singular vector matrix. See DLASD8 for details.

#### Z

Z is DOUBLE PRECISION array,  
dimension ( LDU, NLVL ) if ICOMPQ = 1 and  
dimension ( N ) if ICOMPQ = 0.  
The first K elements of Z(1, I) contain the components of  
the deflation-adjusted updating row vector for subproblems  
on the I-th level.

#### POLES

POLES is DOUBLE PRECISION array,  
dimension ( LDU, 2 \* NLVL ) if ICOMPQ = 1, and not referenced  
if ICOMPQ = 0. If ICOMPQ = 1, on exit, POLES(1, 2\*I - 1) and  
POLES(1, 2\*I) contain the new and old singular values  
involved in the secular equations on the I-th level.

#### GIVPTR

GIVPTR is INTEGER array,  
dimension ( N ) if ICOMPQ = 1, and not referenced if  
ICOMPQ = 0. If ICOMPQ = 1, on exit, GIVPTR( I ) records  
the number of Givens rotations performed on the I-th  
problem on the computation tree.

#### GIVCOL



GIVCOL is INTEGER array,  
dimension ( LDGCOL, 2 \* NLVL ) if ICOMPQ = 1, and not  
referenced if ICOMPQ = 0. If ICOMPQ = 1, on exit, for each I,  
GIVCOL(1, 2 \* I - 1) and GIVCOL(1, 2 \* I) record the locations  
of Givens rotations performed on the I-th level on the  
computation tree.

#### *LDGCOL*

LDGCOL is INTEGER, LDGCOL = > N.  
The leading dimension of arrays GIVCOL and PERM.

#### *PERM*

PERM is INTEGER array,  
dimension ( LDGCOL, NLVL ) if ICOMPQ = 1, and not referenced  
if ICOMPQ = 0. If ICOMPQ = 1, on exit, PERM(1, I) records  
permutations done on the I-th level of the computation tree.

#### *GIVNUM*

GIVNUM is DOUBLE PRECISION array,  
dimension ( LDU, 2 \* NLVL ) if ICOMPQ = 1, and not  
referenced if ICOMPQ = 0. If ICOMPQ = 1, on exit, for each I,  
GIVNUM(1, 2 \* I - 1) and GIVNUM(1, 2 \* I) record the C- and S-  
values of Givens rotations performed on the I-th level on  
the computation tree.

#### *C*

C is DOUBLE PRECISION array,  
dimension ( N ) if ICOMPQ = 1, and dimension 1 if ICOMPQ = 0.  
If ICOMPQ = 1 and the I-th subproblem is not square, on exit,  
C( I ) contains the C-value of a Givens rotation related to  
the right null space of the I-th subproblem.

#### *S*

S is DOUBLE PRECISION array, dimension ( N ) if  
ICOMPQ = 1, and dimension 1 if ICOMPQ = 0. If ICOMPQ = 1  
and the I-th subproblem is not square, on exit, S( I )  
contains the S-value of a Givens rotation related to  
the right null space of the I-th subproblem.

#### *WORK*

WORK is DOUBLE PRECISION array, dimension  
(6 \* N + (SMLSIZ + 1)\*(SMLSIZ + 1)).

#### *IWORK*

IWORK is INTEGER array, dimension (7\*N)

#### *INFO*

INFO is INTEGER  
= 0: successful exit.  
< 0: if INFO = -i, the i-th argument had an illegal value.  
> 0: if INFO = 1, a singular value did not converge

#### **Author**

Univ. of Tennessee  
Univ. of California Berkeley  
Univ. of Colorado Denver  
NAG Ltd.

#### **Date**

June 2017





**Contributors:**

Ming Gu and Huan Ren, Computer Science Division, University of California at Berkeley, USA

**subroutine dlasdq** (character **UPLO**, integer **SQRE**, integer **N**, integer **NCVT**, integer **NRU**, integer **NCC**, double precision, dimension( **\*** ) **D**, double precision, dimension( **\*** ) **E**, double precision, dimension( **ldvt**, **\*** ) **VT**, integer **LDVT**, double precision, dimension( **ldu**, **\*** ) **U**, integer **LDU**, double precision, dimension( **ldc**, **\*** ) **C**, integer **LDC**, double precision, dimension( **\*** ) **WORK**, integer **INFO**)

**DLASDQ** computes the SVD of a real bidiagonal matrix with diagonal **d** and off-diagonal **e**. Used by **sbdsc**.

**Purpose:**

**DLASDQ** computes the singular value decomposition (SVD) of a real (upper or lower) bidiagonal matrix with diagonal **D** and offdiagonal **E**, accumulating the transformations if desired. Letting **B** denote the input bidiagonal matrix, the algorithm computes orthogonal matrices **Q** and **P** such that  $B = Q * S * P^{**T}$  ( $P^{**T}$  denotes the transpose of **P**). The singular values **S** are overwritten on **D**.

The input matrix **U** is changed to  $U * Q$  if desired.

The input matrix **VT** is changed to  $P^{**T} * VT$  if desired.

The input matrix **C** is changed to  $Q^{**T} * C$  if desired.

See "Computing Small Singular Values of Bidiagonal Matrices With Guaranteed High Relative Accuracy," by J. Demmel and W. Kahan, LAPACK Working Note #3, for a detailed description of the algorithm.

**Parameters****UPLO**

**UPLO** is CHARACTER\*1

On entry, **UPLO** specifies whether the input bidiagonal matrix is upper or lower bidiagonal, and whether it is square or not.

**UPLO** = 'U' or 'u' **B** is upper bidiagonal.

**UPLO** = 'L' or 'l' **B** is lower bidiagonal.

**SQRE**

**SQRE** is INTEGER

= 0: then the input matrix is N-by-N.

= 1: then the input matrix is N-by-(N+1) if **UPLO** = 'U' and (N+1)-by-N if **UPLO** = 'L'.

The bidiagonal matrix has

$N = NL + NR + 1$  rows and

$M = N + SQRE \geq N$  columns.

**N**

**N** is INTEGER

On entry, **N** specifies the number of rows and columns in the matrix. **N** must be at least 0.

**NCVT**

**NCVT** is INTEGER

On entry, **NCVT** specifies the number of columns of the matrix **VT**. **NCVT** must be at least 0.

**NRU**

**NRU** is INTEGER

On entry, **NRU** specifies the number of rows of



the matrix U. NRU must be at least 0.

#### *NCC*

NCC is INTEGER

On entry, NCC specifies the number of columns of the matrix C. NCC must be at least 0.

#### *D*

D is DOUBLE PRECISION array, dimension (N)

On entry, D contains the diagonal entries of the bidiagonal matrix whose SVD is desired. On normal exit, D contains the singular values in ascending order.

#### *E*

E is DOUBLE PRECISION array.

dimension is (N-1) if SQRE = 0 and N if SQRE = 1.

On entry, the entries of E contain the offdiagonal entries of the bidiagonal matrix whose SVD is desired. On normal exit, E will contain 0. If the algorithm does not converge, D and E will contain the diagonal and superdiagonal entries of a bidiagonal matrix orthogonally equivalent to the one given as input.

#### *VT*

VT is DOUBLE PRECISION array, dimension (LDVT, NCVT)

On entry, contains a matrix which on exit has been premultiplied by P\*\*T, dimension N-by-NCVT if SQRE = 0 and (N+1)-by-NCVT if SQRE = 1 (not referenced if NCVT=0).

#### *LDVT*

LDVT is INTEGER

On entry, LDVT specifies the leading dimension of VT as declared in the calling (sub) program. LDVT must be at least 1. If NCVT is nonzero LDVT must also be at least N.

#### *U*

U is DOUBLE PRECISION array, dimension (LDU, N)

On entry, contains a matrix which on exit has been postmultiplied by Q, dimension NRU-by-N if SQRE = 0 and NRU-by-(N+1) if SQRE = 1 (not referenced if NRU=0).

#### *LDU*

LDU is INTEGER

On entry, LDU specifies the leading dimension of U as declared in the calling (sub) program. LDU must be at least max( 1, NRU ) .

#### *C*

C is DOUBLE PRECISION array, dimension (LDC, NCC)

On entry, contains an N-by-NCC matrix which on exit has been premultiplied by Q\*\*T dimension N-by-NCC if SQRE = 0 and (N+1)-by-NCC if SQRE = 1 (not referenced if NCC=0).

#### *LDC*

LDC is INTEGER

On entry, LDC specifies the leading dimension of C as declared in the calling (sub) program. LDC must be at least 1. If NCC is nonzero, LDC must also be at least N.

#### *WORK*

WORK is DOUBLE PRECISION array, dimension (4\*N)



Workspace. Only referenced if one of NCVT, NRU, or NCC is nonzero, and if N is at least 2.

#### *INFO*

INFO is INTEGER  
 On exit, a value of 0 indicates a successful exit.  
 If  $INFO < 0$ , argument number  $-INFO$  is illegal.  
 If  $INFO > 0$ , the algorithm did not converge, and INFO specifies how many superdiagonals did not converge.

#### **Author**

Univ. of Tennessee  
 Univ. of California Berkeley  
 Univ. of Colorado Denver  
 NAG Ltd.

#### **Date**

June 2016

#### **Contributors:**

Ming Gu and Huan Ren, Computer Science Division, University of California at Berkeley, USA

**subroutine dlasdt (integer N, integer LVL, integer ND, integer, dimension( \* ) INODE, integer, dimension( \* ) NDIML, integer, dimension( \* ) NDIMR, integer MSUB)**  
**DLASDT** creates a tree of subproblems for bidiagonal divide and conquer. Used by sbdsdc.

#### **Purpose:**

DLASDT creates a tree of subproblems for bidiagonal divide and conquer.

#### **Parameters**

*N*

N is INTEGER  
 On entry, the number of diagonal elements of the bidiagonal matrix.

*LVL*

LVL is INTEGER  
 On exit, the number of levels on the computation tree.

*ND*

ND is INTEGER  
 On exit, the number of nodes on the tree.

*INODE*

INODE is INTEGER array, dimension ( N )  
 On exit, centers of subproblems.

*NDIML*

NDIML is INTEGER array, dimension ( N )  
 On exit, row dimensions of left children.

*NDIMR*

NDIMR is INTEGER array, dimension ( N )  
 On exit, row dimensions of right children.

*MSUB*

MSUB is INTEGER  
 On entry, the maximum row dimension each subproblem at the bottom of the tree can be of.



**Author**

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

**Date**

December 2016

**Contributors:**

Ming Gu and Huan Ren, Computer Science Division, University of California at Berkeley, USA

**subroutine dlaset (character UPLO, integer M, integer N, double precision ALPHA, double precision BETA, double precision, dimension( lda, \* ) A, integer LDA)**

**DLASET** initializes the off-diagonal elements and the diagonal elements of a matrix to given values.

**Purpose:**

DLASET initializes an m-by-n matrix A to BETA on the diagonal and ALPHA on the offdiagonals.

**Parameters***UPLO*

UPLO is CHARACTER\*1

Specifies the part of the matrix A to be set.

= 'U': Upper triangular part is set; the strictly lower triangular part of A is not changed.

= 'L': Lower triangular part is set; the strictly upper triangular part of A is not changed.

Otherwise: All of the matrix A is set.

*M*

M is INTEGER

The number of rows of the matrix A.  $M \geq 0$ .

*N*

N is INTEGER

The number of columns of the matrix A.  $N \geq 0$ .

*ALPHA*

ALPHA is DOUBLE PRECISION

The constant to which the offdiagonal elements are to be set.

*BETA*

BETA is DOUBLE PRECISION

The constant to which the diagonal elements are to be set.

*A*

A is DOUBLE PRECISION array, dimension (LDA,N)

On exit, the leading m-by-n submatrix of A is set as follows:

if UPLO = 'U',  $A(i,j) = ALPHA$ ,  $1 \leq i \leq j-1$ ,  $1 \leq j \leq n$ ,  
 if UPLO = 'L',  $A(i,j) = ALPHA$ ,  $j+1 \leq i \leq m$ ,  $1 \leq j \leq n$ ,  
 otherwise,  $A(i,j) = ALPHA$ ,  $1 \leq i \leq m$ ,  $1 \leq j \leq n$ ,  $i \neq j$ ,

and, for all UPLO,  $A(i,i) = BETA$ ,  $1 \leq i \leq \min(m,n)$ .

*LDA*

LDA is INTEGER

The leading dimension of the array A.  $LDA \geq \max(1,M)$ .



**Author**

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

**Date**

December 2016

**subroutine dlasr** (character **SIDE**, character **PIVOT**, character **DIRECT**, integer **M**, integer **N**, double precision, dimension( \*) **C**, double precision, dimension( \*) **S**, double precision, dimension( **lda**, \*) **A**, integer **LDA**)

**DLASR** applies a sequence of plane rotations to a general rectangular matrix.

**Purpose:**

DLASR applies a sequence of plane rotations to a real matrix **A**, from either the left or the right.

When **SIDE** = 'L', the transformation takes the form

$$A := P * A$$

and when **SIDE** = 'R', the transformation takes the form

$$A := A * P^{**T}$$

where **P** is an orthogonal matrix consisting of a sequence of **z** plane rotations, with **z** = **M** when **SIDE** = 'L' and **z** = **N** when **SIDE** = 'R', and **P\*\*T** is the transpose of **P**.

When **DIRECT** = 'F' (Forward sequence), then

$$P = P(z-1) * \dots * P(2) * P(1)$$

and when **DIRECT** = 'B' (Backward sequence), then

$$P = P(1) * P(2) * \dots * P(z-1)$$

where **P(k)** is a plane rotation matrix defined by the 2-by-2 rotation

$$R(k) = \begin{pmatrix} c(k) & s(k) \\ -s(k) & c(k) \end{pmatrix}$$

When **PIVOT** = 'V' (Variable pivot), the rotation is performed for the plane (k,k+1), i.e., **P(k)** has the form

$$P(k) = \begin{pmatrix} 1 & & & & & \\ & \dots & & & & \\ & & 1 & & & \\ & & & c(k) & s(k) & \\ & & & -s(k) & c(k) & \\ & & & & 1 & \\ & & & & & \dots & \\ & & & & & & 1 \end{pmatrix}$$

where **R(k)** appears as a rank-2 modification to the identity matrix in rows and columns **k** and **k+1**.



When PIVOT = 'T' (Top pivot), the rotation is performed for the plane (1,k+1), so P(k) has the form

$$P(k) = \begin{pmatrix} c(k) & & s(k) & & \\ & 1 & & & \\ & & \dots & & \\ & & & 1 & \\ -s(k) & & c(k) & & \\ & & & 1 & \\ & & & & \dots \\ & & & & & 1 \end{pmatrix}$$

where R(k) appears in rows and columns 1 and k+1.

Similarly, when PIVOT = 'B' (Bottom pivot), the rotation is performed for the plane (k,z), giving P(k) the form

$$P(k) = \begin{pmatrix} 1 & & & & \\ & \dots & & & \\ & & 1 & & \\ & & & c(k) & s(k) \\ & & & 1 & \\ & & & & \dots \\ & & & & & 1 \\ & & -s(k) & & c(k) \end{pmatrix}$$

where R(k) appears in rows and columns k and z. The rotations are performed without ever forming P(k) explicitly.

## Parameters

### SIDE

SIDE is CHARACTER\*1

Specifies whether the plane rotation matrix P is applied to A on the left or the right.

= 'L': Left, compute A := P\*A

= 'R': Right, compute A:= A\*P\*\*T

### PIVOT

PIVOT is CHARACTER\*1

Specifies the plane for which P(k) is a plane rotation matrix.

= 'V': Variable pivot, the plane (k,k+1)

= 'T': Top pivot, the plane (1,k+1)

= 'B': Bottom pivot, the plane (k,z)

### DIRECT

DIRECT is CHARACTER\*1

Specifies whether P is a forward or backward sequence of plane rotations.

= 'F': Forward, P = P(z-1)\*...\*P(2)\*P(1)

= 'B': Backward, P = P(1)\*P(2)\*...\*P(z-1)

### M

M is INTEGER

The number of rows of the matrix A. If m <= 1, an immediate return is effected.

### N

N is INTEGER

The number of columns of the matrix A. If n <= 1, an



immediate return is effected.

*C*

*C* is DOUBLE PRECISION array, dimension

(*M*-1) if *SIDE* = 'L'

(*N*-1) if *SIDE* = 'R'

The cosines *c*(*k*) of the plane rotations.

*S*

*S* is DOUBLE PRECISION array, dimension

(*M*-1) if *SIDE* = 'L'

(*N*-1) if *SIDE* = 'R'

The sines *s*(*k*) of the plane rotations. The 2-by-2 plane rotation part of the matrix *P*(*k*), *R*(*k*), has the form

$$R(k) = \begin{pmatrix} c(k) & s(k) \\ -s(k) & c(k) \end{pmatrix}.$$

*A*

*A* is DOUBLE PRECISION array, dimension (*LDA*,*N*)

The *M*-by-*N* matrix *A*. On exit, *A* is overwritten by *P*\**A* if *SIDE* = 'L' or by *A*\**P*\*\**T* if *SIDE* = 'R'.

*LDA*

*LDA* is INTEGER

The leading dimension of the array *A*. *LDA* ≥ max(1,*M*).

#### Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

#### Date

December 2016

**subroutine dlassq (integer *N*, double precision, dimension( \* ) *X*, integer *INCX*, double precision *SCALE*, double precision *SUMSQ*)**

**DLASSQ** updates a sum of squares represented in scaled form.

#### Purpose:

DLASSQ returns the values *scl* and *sumsq* such that

$$(scl^{**2}) * sumsq = x(1)^{**2} + \dots + x(n)^{**2} + (scale^{**2}) * sumsq,$$

where  $x(i) = X(1 + (i - 1) * INCX)$ . The value of *sumsq* is assumed to be non-negative and *scl* returns the value

$$scl = \max(scale, \text{abs}(x(i))).$$

*scale* and *sumsq* must be supplied in *SCALE* and *SUMSQ* and *scl* and *sumsq* are overwritten on *SCALE* and *SUMSQ* respectively.

The routine makes only one pass through the vector *x*.

#### Parameters

*N*

*N* is INTEGER

The number of elements to be used from the vector *X*.

*X*



$X$  is DOUBLE PRECISION array, dimension  $(1+(N-1)*INCX)$   
 The vector for which a scaled sum of squares is computed.  
 $x(i) = X(1 + (i - 1)*INCX)$ ,  $1 \leq i \leq n$ .

*INCX*

*INCX* is INTEGER  
 The increment between successive values of the vector  $X$ .  
 $INCX > 0$ .

*SCALE*

*SCALE* is DOUBLE PRECISION  
 On entry, the value *scale* in the equation above.  
 On exit, *SCALE* is overwritten with *scl*, the scaling factor  
 for the sum of squares.

*SUMSQ*

*SUMSQ* is DOUBLE PRECISION  
 On entry, the value *sumsq* in the equation above.  
 On exit, *SUMSQ* is overwritten with *smsq*, the basic sum of  
 squares from which *scl* has been factored out.

#### Author

Univ. of Tennessee  
 Univ. of California Berkeley  
 Univ. of Colorado Denver  
 NAG Ltd.

#### Date

December 2016

**subroutine dlasv2 (double precision F, double precision G, double precision H, double precision  
 SSMIN, double precision SSMAX, double precision SNR, double precision CSR, double precision  
 SNL, double precision CSL)**

**DLASV2** computes the singular value decomposition of a 2-by-2 triangular matrix.

#### Purpose:

DLASV2 computes the singular value decomposition of a 2-by-2  
 triangular matrix

$$\begin{bmatrix} F & G \\ 0 & H \end{bmatrix}.$$

On return,  $\text{abs}(\text{SSMAX})$  is the larger singular value,  $\text{abs}(\text{SSMIN})$  is the  
 smaller singular value, and (CSL,SNL) and (CSR,SNR) are the left and  
 right singular vectors for  $\text{abs}(\text{SSMAX})$ , giving the decomposition

$$\begin{bmatrix} \text{CSL} & \text{SNL} \end{bmatrix} \begin{bmatrix} F & G \\ 0 & H \end{bmatrix} \begin{bmatrix} \text{CSR} & -\text{SNR} \end{bmatrix} = \begin{bmatrix} \text{SSMAX} & 0 \\ 0 & \text{SSMIN} \end{bmatrix}.$$

#### Parameters

*F*

*F* is DOUBLE PRECISION  
 The (1,1) element of the 2-by-2 matrix.

*G*

*G* is DOUBLE PRECISION  
 The (1,2) element of the 2-by-2 matrix.

*H*

*H* is DOUBLE PRECISION  
 The (2,2) element of the 2-by-2 matrix.





*SSMIN*

SSMIN is DOUBLE PRECISION  
abs(SSMIN) is the smaller singular value.

*SSMAX*

SSMAX is DOUBLE PRECISION  
abs(SSMAX) is the larger singular value.

*SNL*

SNL is DOUBLE PRECISION

*CSL*

CSL is DOUBLE PRECISION  
The vector (CSL, SNL) is a unit left singular vector for the singular value abs(SSMAX).

*SNR*

SNR is DOUBLE PRECISION

*CSR*

CSR is DOUBLE PRECISION  
The vector (CSR, SNR) is a unit right singular vector for the singular value abs(SSMAX).

**Author**

Univ. of Tennessee  
Univ. of California Berkeley  
Univ. of Colorado Denver  
NAG Ltd.

**Date**

December 2016

**Further Details:**

Any input parameter may be aliased with any output parameter.

Barring over/underflow and assuming a guard digit in subtraction, all output quantities are correct to within a few units in the last place (ulps).

In IEEE arithmetic, the code works correctly if one matrix element is infinite.

Overflow will not occur unless the largest singular value itself overflows or is within a few ulps of overflow. (On machines with partial overflow, like the Cray, overflow may occur if the largest singular value is within a factor of 2 of overflow.)

Underflow is harmless if underflow is gradual. Otherwise, results may correspond to a matrix modified by perturbations of size near the underflow threshold.

**integer function ieeeck (integer ISPEC, real ZERO, real ONE)****IEEECK****Purpose:**

IEEECK is called from the ILAENV to verify that Infinity and possibly NaN arithmetic is safe (i.e. will not trap).



**Parameters***ISPEC*

ISPEC is INTEGER

Specifies whether to test just for infinity arithmetic or whether to test for infinity and NaN arithmetic.

= 0: Verify infinity arithmetic only.

= 1: Verify infinity and NaN arithmetic.

*ZERO*

ZERO is REAL

Must contain the value 0.0

This is passed to prevent the compiler from optimizing away this code.

*ONE*

ONE is REAL

Must contain the value 1.0

This is passed to prevent the compiler from optimizing away this code.

RETURN VALUE: INTEGER

= 0: Arithmetic failed to produce the correct answers

= 1: Arithmetic produced the correct answers

**Author**

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

**Date**

December 2016

**integer function iladlc (integer M, integer N, double precision, dimension( lda, \* ) A, integer LDA)**

ILADLC scans a matrix for its last non-zero column.

**Purpose:**

ILADLC scans A for its last non-zero column.

**Parameters***M*

M is INTEGER

The number of rows of the matrix A.

*N*

N is INTEGER

The number of columns of the matrix A.

*A*

A is DOUBLE PRECISION array, dimension (LDA,N)

The m by n matrix A.

*LDA*

LDA is INTEGER

The leading dimension of the array A. LDA  $\geq \max(1,M)$ .

**Author**

Univ. of Tennessee

Univ. of California Berkeley



Univ. of Colorado Denver

NAG Ltd.

#### Date

December 2016

**integer function iladlr (integer M, integer N, double precision, dimension( lda, \* ) A, integer LDA)**

**ILADLR** scans a matrix for its last non-zero row.

#### Purpose:

ILADLR scans A for its last non-zero row.

#### Parameters

*M*

M is INTEGER

The number of rows of the matrix A.

*N*

N is INTEGER

The number of columns of the matrix A.

*A*

A is DOUBLE PRECISION array, dimension (LDA,N)

The m by n matrix A.

*LDA*

LDA is INTEGER

The leading dimension of the array A. LDA  $\geq$  max(1,M).

#### Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

#### Date

December 2016

**integer function ilaenv (integer ISPEC, character\*( \*) NAME, character\*( \*) OPTS, integer N1, integer N2, integer N3, integer N4)**

**ILAENV**

#### Purpose:

ILAENV is called from the LAPACK routines to choose problem-dependent parameters for the local environment. See ISPEC for a description of the parameters.

ILAENV returns an INTEGER

if ILAENV  $\geq$  0: ILAENV returns the value of the parameter specified by ISPEC

if ILAENV < 0: if ILAENV = -k, the k-th argument had an illegal value.

This version provides a set of parameters which should give good, but not optimal, performance on many of the currently available computers. Users are encouraged to modify this subroutine to set the tuning parameters for their particular machine using the option and problem size information in the arguments.

This routine will not function correctly if it is converted to all lower case. Converting it to all upper case is allowed.



**Parameters***ISPEC*

ISPEC is INTEGER

Specifies the parameter to be returned as the value of ILAENV.

- = 1: the optimal blocksize; if this value is 1, an unblocked algorithm will give the best performance.
- = 2: the minimum block size for which the block routine should be used; if the usable block size is less than this value, an unblocked routine should be used.
- = 3: the crossover point (in a block routine, for N less than this value, an unblocked routine should be used)
- = 4: the number of shifts, used in the nonsymmetric eigenvalue routines (DEPRECATED)
- = 5: the minimum column dimension for blocking to be used; rectangular blocks must have dimension at least k by m, where k is given by ILAENV(2,...) and m by ILAENV(5,...)
- = 6: the crossover point for the SVD (when reducing an m by n matrix to bidiagonal form, if  $\max(m,n)/\min(m,n)$  exceeds this value, a QR factorization is used first to reduce the matrix to a triangular form.)
- = 7: the number of processors
- = 8: the crossover point for the multishift QR method for nonsymmetric eigenvalue problems (DEPRECATED)
- = 9: maximum size of the subproblems at the bottom of the computation tree in the divide-and-conquer algorithm (used by xGELSD and xGESDD)
- =10: ieee NaN arithmetic can be trusted not to trap
- =11: infinity arithmetic can be trusted not to trap
- 12 <= ISPEC <= 16:  
xHSEQR or related subroutines,  
see IPARMQ for detailed explanation

*NAME*

NAME is CHARACTER\*(\*)

The name of the calling subroutine, in either upper case or lower case.

*OPTS*

OPTS is CHARACTER\*(\*)

The character options to the subroutine NAME, concatenated into a single character string. For example, UPLO = 'U', TRANS = 'T', and DIAG = 'N' for a triangular routine would be specified as OPTS = 'UTN'.

*N1*

N1 is INTEGER

*N2*

N2 is INTEGER

*N3*

N3 is INTEGER

*N4*

N4 is INTEGER

Problem dimensions for the subroutine NAME; these may not all be required.

**Author**

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

#### Date

November 2019

#### Further Details:

The following conventions have been used when calling ILAENV from the LAPACK routines:

- 1) OPTS is a concatenation of all of the character options to subroutine NAME, in the same order that they appear in the argument list for NAME, even if they are not used in determining the value of the parameter specified by ISPEC.
- 2) The problem dimensions N1, N2, N3, N4 are specified in the order that they appear in the argument list for NAME. N1 is used first, N2 second, and so on, and unused problem dimensions are passed a value of -1.
- 3) The parameter value returned by ILAENV is checked for validity in the calling subroutine. For example, ILAENV is used to retrieve the optimal blocksize for STRTRI as follows:

```
NB = ILAENV( 1, 'STRTRI', UPLO // DIAG, N, -1, -1, -1 )
IF( NB.LE.1 ) NB = MAX( 1, N )
```

**integer function ilaenv2stage (integer ISPEC, character\*( \*) NAME, character\*( \*) OPTS, integer N1, integer N2, integer N3, integer N4)  
ILAENV2STAGE**

#### Purpose:

ILAENV2STAGE is called from the LAPACK routines to choose problem-dependent parameters for the local environment. See ISPEC for a description of the parameters.

It sets problem and machine dependent parameters useful for \*\_2STAGE and related subroutines.

ILAENV2STAGE returns an INTEGER

if ILAENV2STAGE  $\geq$  0: ILAENV2STAGE returns the value of the parameter specified by ISPEC

if ILAENV2STAGE  $<$  0: if ILAENV2STAGE = -k, the k-th argument had an illegal value.

This version provides a set of parameters which should give good, but not optimal, performance on many of the currently available computers for the 2-stage solvers. Users are encouraged to modify this subroutine to set the tuning parameters for their particular machine using the option and problem size information in the arguments.

This routine will not function correctly if it is converted to all lower case. Converting it to all upper case is allowed.

#### Parameters

*ISPEC*

ISPEC is INTEGER

Specifies the parameter to be returned as the value of ILAENV2STAGE.



= 1: the optimal blocksize nb for the reduction to BAND

= 2: the optimal blocksize ib for the eigenvectors  
singular vectors update routine

= 3: The length of the array that store the Housholder  
representation for the second stage  
Band to Tridiagonal or Bidiagonal

= 4: The workspace needed for the routine in input.

= 5: For future release.

#### *NAME*

NAME is CHARACTER\*(\*)

The name of the calling subroutine, in either upper case or lower case.

#### *OPTS*

OPTS is CHARACTER\*(\*)

The character options to the subroutine NAME, concatenated into a single character string. For example, UPLO = 'U', TRANS = 'T', and DIAG = 'N' for a triangular routine would be specified as OPTS = 'UTN'.

#### *N1*

N1 is INTEGER

#### *N2*

N2 is INTEGER

#### *N3*

N3 is INTEGER

#### *N4*

N4 is INTEGER

Problem dimensions for the subroutine NAME; these may not all be required.

#### **Author**

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Nick R. Papior

#### **Date**

July 2017

#### **Further Details:**

The following conventions have been used when calling ILAENV2STAGE from the LAPACK routines:

- 1) OPTS is a concatenation of all of the character options to subroutine NAME, in the same order that they appear in the argument list for NAME, even if they are not used in determining the value of the parameter specified by ISPEC.
- 2) The problem dimensions N1, N2, N3, N4 are specified in the order that they appear in the argument list for NAME. N1 is used first, N2 second, and so on, and unused problem dimensions are



passed a value of -1.

- 3) The parameter value returned by ILAENV2STAGE is checked for validity in the calling subroutine.

**integer function iparmq (integer ISPEC, character, dimension( \* ) NAME, character, dimension( \* ) OPTS, integer N, integer ILO, integer IHI, integer LWORK)  
IPARMQ**

#### **Purpose:**

This program sets problem and machine dependent parameters useful for xHSEQR and related subroutines for eigenvalue problems. It is called whenever IPARMQ is called with  $12 \leq \text{ISPEC} \leq 16$

#### **Parameters**

##### *ISPEC*

ISPEC is INTEGER

ISPEC specifies which tunable parameter IPARMQ should return.

ISPEC=12: (INMIN) Matrices of order nmin or less are sent directly to xLAHQZ, the implicit double shift QR algorithm. NMIN must be at least 11.

ISPEC=13: (INWIN) Size of the deflation window. This is best set greater than or equal to the number of simultaneous shifts NS. Larger matrices benefit from larger deflation windows.

ISPEC=14: (INIBL) Determines when to stop nibbling and invest in an (expensive) multi-shift QR sweep. If the aggressive early deflation subroutine finds LD converged eigenvalues from an order NW deflation window and  $LD > (NW * \text{NIBBLE}) / 100$ , then the next QR sweep is skipped and early deflation is applied immediately to the remaining active diagonal block. Setting  $\text{IPARMQ}(\text{ISPEC}=14) = 0$  causes TTQRE to skip a multi-shift QR sweep whenever early deflation finds a converged eigenvalue. Setting  $\text{IPARMQ}(\text{ISPEC}=14)$  greater than or equal to 100 prevents TTQRE from skipping a multi-shift QR sweep.

ISPEC=15: (NSHFTS) The number of simultaneous shifts in a multi-shift QR iteration.

ISPEC=16: (IACC22) IPARMQ is set to 0, 1 or 2 with the following meanings.

- 0: During the multi-shift QR/QZ sweep, blocked eigenvalue reordering, blocked Hessenberg-triangular reduction, reflections and/or rotations are not accumulated when updating the far-from-diagonal matrix entries.
- 1: During the multi-shift QR/QZ sweep, blocked eigenvalue reordering, blocked



Hessenberg-triangular reduction, reflections and/or rotations are accumulated, and matrix-matrix multiplication is used to update the far-from-diagonal matrix entries.

- 2: During the multi-shift QR/QZ sweep, blocked eigenvalue reordering, blocked Hessenberg-triangular reduction, reflections and/or rotations are accumulated, and 2-by-2 block structure is exploited during matrix-matrix multiplies.

(If xTRMM is slower than xGEMM, then IPARMQ(ISPEC=16)=1 may be more efficient than IPARMQ(ISPEC=16)=2 despite the greater level of arithmetic work implied by the latter choice.)

#### *NAME*

NAME is CHARACTER string  
Name of the calling subroutine

#### *OPTS*

OPTS is CHARACTER string  
This is a concatenation of the string arguments to TTQRE.

#### *N*

N is INTEGER  
N is the order of the Hessenberg matrix H.

#### *ILO*

ILO is INTEGER

#### *IHI*

IHI is INTEGER  
It is assumed that H is already upper triangular in rows and columns 1:ILO-1 and IHI+1:N.

#### *LWORK*

LWORK is INTEGER  
The amount of workspace available.

#### **Author**

Univ. of Tennessee  
Univ. of California Berkeley  
Univ. of Colorado Denver  
NAG Ltd.

#### **Date**

June 2017

#### **Further Details:**

Little is known about how best to choose these parameters. It is possible to use different values of the parameters for each of CHSEQR, DHSEQR, SHSEQR and ZHSEQR.

It is probably best to choose different parameters for different matrices and different parameters at different times during the iteration, but this has not been





implemented --- yet.

The best choices of most of the parameters depend in an ill-understood way on the relative execution rate of xLAQR3 and xLAQR5 and on the nature of each particular eigenvalue problem. Experiment may be the only practical way to determine which choices are most effective.

Following is a list of default values supplied by IPARMQ. These defaults may be adjusted in order to attain better performance in any particular computational environment.

IPARMQ(ISPEC=12) The xLAHQR vs xLAQR0 crossover point.  
Default: 75. (Must be at least 11.)

IPARMQ(ISPEC=13) Recommended deflation window size.  
This depends on ILO, IHI and NS, the number of simultaneous shifts returned by IPARMQ(ISPEC=15). The default for  $(IHI-ILO+1) \leq 500$  is NS. The default for  $(IHI-ILO+1) > 500$  is  $3*NS/2$ .

IPARMQ(ISPEC=14) Nibble crossover point. Default: 14.

IPARMQ(ISPEC=15) Number of simultaneous shifts, NS, a multi-shift QR iteration.

If  $IHI-ILO+1$  is ...

greater than	...but less	... the
or equal to ...	than	default is

0	30	NS = 2+
30	60	NS = 4+
60	150	NS = 10
150	590	NS = **
590	3000	NS = 64
3000	6000	NS = 128
6000	infinity	NS = 256

(+) By default matrices of this order are passed to the implicit double shift routine xLAHQR. See IPARMQ(ISPEC=12) above. These values of NS are used only in case of a rare xLAHQR failure.

(\*\*) The asterisks (\*\*) indicate an ad-hoc function increasing from 10 to 64.

IPARMQ(ISPEC=16) Select structured matrix multiply.  
(See ISPEC=16 above for details.)  
Default: 3.

**logical function lsamen (integer N, character\*( \*) CA, character\*( \*) CB)**  
**LSAMEN**

**Purpose:**



LSAMEN tests if the first N letters of CA are the same as the first N letters of CB, regardless of case.  
 LSAMEN returns .TRUE. if CA and CB are equivalent except for case and .FALSE. otherwise. LSAMEN also returns .FALSE. if LEN( CA ) or LEN( CB ) is less than N.

**Parameters***N*

N is INTEGER

The number of characters in CA and CB to be compared.

*CA*

CA is CHARACTER\*(\*)

*CB*

CB is CHARACTER\*(\*)

CA and CB specify two character strings of length at least N.

Only the first N characters of each string will be accessed.

**Author**

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

**Date**

December 2016

**subroutine scombssq (real, dimension( 2 ) V1, real, dimension( 2 ) V2)****SCOMBSSQ** adds two scaled sum of squares quantities**Purpose:**SCOMBSSQ adds two scaled sum of squares quantities,  $V1 := V1 + V2$ .

That is,

$$V1\_scale^{**2} * V1\_sumsq := V1\_scale^{**2} * V1\_sumsq \\ + V2\_scale^{**2} * V2\_sumsq$$

**Parameters***V1*

V1 is REAL array, dimension (2).

The first scaled sum.

 $V1(1) = V1\_scale$ ,  $V1(2) = V1\_sumsq$ .*V2*

V2 is REAL array, dimension (2).

The second scaled sum.

 $V2(1) = V2\_scale$ ,  $V2(2) = V2\_sumsq$ .**Author**

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

**Date**

November 2018



**logical function sisnan (real, intent(in) SIN)**

SISNAN tests input for NaN.

**Purpose:**

SISNAN returns .TRUE. if its argument is NaN, and .FALSE. otherwise. To be replaced by the Fortran 2003 intrinsic in the future.

**Parameters**

*SIN*

SIN is REAL

Input to test for NaN.

**Author**

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

**Date**

June 2017

**subroutine slabad (real SMALL, real LARGE)**

**SLABAD**

**Purpose:**

SLABAD takes as input the values computed by SLAMCH for underflow and overflow, and returns the square root of each of these values if the log of LARGE is sufficiently large. This subroutine is intended to identify machines with a large exponent range, such as the Crays, and redefine the underflow and overflow limits to be the square roots of the values computed by SLAMCH. This subroutine is needed because SLAMCH does not compensate for poor arithmetic in the upper half of the exponent range, as is found on a Cray.

**Parameters**

*SMALL*

SMALL is REAL

On entry, the underflow threshold as computed by SLAMCH.

On exit, if LOG10(LARGE) is sufficiently large, the square root of SMALL, otherwise unchanged.

*LARGE*

LARGE is REAL

On entry, the overflow threshold as computed by SLAMCH.

On exit, if LOG10(LARGE) is sufficiently large, the square root of LARGE, otherwise unchanged.

**Author**

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

**Date**

December 2016



**subroutine slacpy (character UPLO, integer M, integer N, real, dimension( lda, \* ) A, integer LDA, real, dimension( ldb, \* ) B, integer LDB)**

**SLACPY** copies all or part of one two-dimensional array to another.

**Purpose:**

SLACPY copies all or part of a two-dimensional matrix A to another matrix B.

**Parameters**

*UPLO*

UPLO is CHARACTER\*1

Specifies the part of the matrix A to be copied to B.

= 'U': Upper triangular part

= 'L': Lower triangular part

Otherwise: All of the matrix A

*M*

M is INTEGER

The number of rows of the matrix A.  $M \geq 0$ .

*N*

N is INTEGER

The number of columns of the matrix A.  $N \geq 0$ .

*A*

A is REAL array, dimension (LDA,N)

The m by n matrix A. If UPLO = 'U', only the upper triangle or trapezoid is accessed; if UPLO = 'L', only the lower triangle or trapezoid is accessed.

*LDA*

LDA is INTEGER

The leading dimension of the array A.  $LDA \geq \max(1,M)$ .

*B*

B is REAL array, dimension (LDB,N)

On exit,  $B = A$  in the locations specified by UPLO.

*LDB*

LDB is INTEGER

The leading dimension of the array B.  $LDB \geq \max(1,M)$ .

**Author**

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

**Date**

December 2016

**subroutine slae2 (real A, real B, real C, real RT1, real RT2)**

**SLAE2** computes the eigenvalues of a 2-by-2 symmetric matrix.

**Purpose:**

SLAE2 computes the eigenvalues of a 2-by-2 symmetric matrix

$\begin{bmatrix} A & B \\ B & C \end{bmatrix}$

$\begin{bmatrix} B & C \end{bmatrix}$ .

On return, RT1 is the eigenvalue of larger absolute value, and RT2



is the eigenvalue of smaller absolute value.

### Parameters

*A*

*A* is REAL

The (1,1) element of the 2-by-2 matrix.

*B*

*B* is REAL

The (1,2) and (2,1) elements of the 2-by-2 matrix.

*C*

*C* is REAL

The (2,2) element of the 2-by-2 matrix.

*RT1*

*RT1* is REAL

The eigenvalue of larger absolute value.

*RT2*

*RT2* is REAL

The eigenvalue of smaller absolute value.

### Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

### Date

December 2016

### Further Details:

*RT1* is accurate to a few ulps barring over/underflow.

*RT2* may be inaccurate if there is massive cancellation in the determinant  $A*C-B*B$ ; higher precision or correctly rounded or correctly truncated arithmetic would be needed to compute *RT2* accurately in all cases.

Overflow is possible only if *RT1* is within a factor of 5 of overflow.

Underflow is harmless if the input data is 0 or exceeds  
underflow\_threshold / macheps.

**subroutine slaebz (integer IJOB, integer NITMAX, integer N, integer MMAX, integer MINP, integer NBMIN, real ABSTOL, real RELTOL, real PIVMIN, real, dimension( \*) D, real, dimension( \*) E, real, dimension( \*) E2, integer, dimension( \*) NVAL, real, dimension( mmax, \*) AB, real, dimension( \*) C, integer MOUT, integer, dimension( mmax, \*) NAB, real, dimension( \*) WORK, integer, dimension( \*) IWORK, integer INFO)**

**SLAEBZ** computes the number of eigenvalues of a real symmetric tridiagonal matrix which are less than or equal to a given value, and performs other tasks required by the routine **sstebz**.

### Purpose:

**SLAEBZ** contains the iteration loops which compute and use the function  $N(w)$ , which is the count of eigenvalues of a symmetric tridiagonal matrix  $T$  less than or equal to its argument  $w$ . It performs a choice of two types of loops:

$IJOB=1$ , followed by



IJOB=2: It takes as input a list of intervals and returns a list of sufficiently small intervals whose union contains the same eigenvalues as the union of the original intervals. The input intervals are  $(AB(j,1), AB(j,2)]$ ,  $j=1, \dots, MINP$ . The output interval  $(AB(j,1), AB(j,2)]$  will contain eigenvalues  $NAB(j,1)+1, \dots, NAB(j,2)$ , where  $1 \leq j \leq MOUT$ .

IJOB=3: It performs a binary search in each input interval  $(AB(j,1), AB(j,2)]$  for a point  $w(j)$  such that  $N(w(j))=NVAL(j)$ , and uses  $C(j)$  as the starting point of the search. If such a  $w(j)$  is found, then on output  $AB(j,1)=AB(j,2)=w$ . If no such  $w(j)$  is found, then on output  $(AB(j,1), AB(j,2)]$  will be a small interval containing the point where  $N(w)$  jumps through  $NVAL(j)$ , unless that point lies outside the initial interval.

Note that the intervals are in all cases half-open intervals, i.e., of the form  $(a, b]$ , which includes  $b$  but not  $a$ .

To avoid underflow, the matrix should be scaled so that its largest element is no greater than  $\text{overflow}^{**}(1/2) * \text{underflow}^{**}(1/4)$  in absolute value. To assure the most accurate computation of small eigenvalues, the matrix should be scaled to be not much smaller than that, either.

See W. Kahan "Accurate Eigenvalues of a Symmetric Tridiagonal Matrix", Report CS41, Computer Science Dept., Stanford University, July 21, 1966

Note: the arguments are, in general, \*not\* checked for unreasonable values.

## Parameters

### IJOB

IJOB is INTEGER

Specifies what is to be done:

- = 1: Compute NAB for the initial intervals.
- = 2: Perform bisection iteration to find eigenvalues of T.
- = 3: Perform bisection iteration to invert  $N(w)$ , i.e., to find a point which has a specified number of eigenvalues of T to its left.

Other values will cause SLAEBZ to return with INFO=-1.

### NITMAX

NITMAX is INTEGER

The maximum number of "levels" of bisection to be performed, i.e., an interval of width  $W$  will not be made smaller than  $2^{(-NITMAX)} * W$ . If not all intervals have converged after NITMAX iterations, then INFO is set to the number of non-converged intervals.

### N

N is INTEGER

The dimension  $n$  of the tridiagonal matrix T. It must be at least 1.

### MMA

MMA is INTEGER

The maximum number of intervals. If more than MMA intervals



are generated, then SLAEBZ will quit with  $\text{INFO}=\text{MMAX}+1$ .

#### *MINP*

MINP is INTEGER

The initial number of intervals. It may not be greater than MMAX.

#### *NBMIN*

NBMIN is INTEGER

The smallest number of intervals that should be processed using a vector loop. If zero, then only the scalar loop will be used.

#### *ABSTOL*

ABSTOL is REAL

The minimum (absolute) width of an interval. When an interval is narrower than ABSTOL, or than RELTOL times the larger (in magnitude) endpoint, then it is considered to be sufficiently small, i.e., converged. This must be at least zero.

#### *RELTOL*

RELTOL is REAL

The minimum relative width of an interval. When an interval is narrower than ABSTOL, or than RELTOL times the larger (in magnitude) endpoint, then it is considered to be sufficiently small, i.e., converged. Note: this should always be at least  $\text{radix} \times \text{machine epsilon}$ .

#### *PIVMIN*

PIVMIN is REAL

The minimum absolute value of a "pivot" in the Sturm sequence loop.

This must be at least  $\max |e(j)|^2 \times \text{safe\_min}$  and at least  $\text{safe\_min}$ , where  $\text{safe\_min}$  is at least the smallest number that can divide one without overflow.

#### *D*

D is REAL array, dimension (N)

The diagonal elements of the tridiagonal matrix T.

#### *E*

E is REAL array, dimension (N)

The offdiagonal elements of the tridiagonal matrix T in positions 1 through N-1. E(N) is arbitrary.

#### *E2*

E2 is REAL array, dimension (N)

The squares of the offdiagonal elements of the tridiagonal matrix T. E2(N) is ignored.

#### *NVAL*

NVAL is INTEGER array, dimension (MINP)

If IJOB=1 or 2, not referenced.

If IJOB=3, the desired values of N(w). The elements of NVAL will be reordered to correspond with the intervals in AB. Thus, NVAL(j) on output will not, in general be the same as NVAL(j) on input, but it will correspond with the interval (AB(j,1),AB(j,2)] on output.

#### *AB*



AB is REAL array, dimension (MMAX,2)

The endpoints of the intervals. AB(j,1) is a(j), the left endpoint of the j-th interval, and AB(j,2) is b(j), the right endpoint of the j-th interval. The input intervals will, in general, be modified, split, and reordered by the calculation.

### C

C is REAL array, dimension (MMAX)

If IJOB=1, ignored.

If IJOB=2, workspace.

If IJOB=3, then on input C(j) should be initialized to the first search point in the binary search.

### MOUT

MOUT is INTEGER

If IJOB=1, the number of eigenvalues in the intervals.

If IJOB=2 or 3, the number of intervals output.

If IJOB=3, MOUT will equal MINP.

### NAB

NAB is INTEGER array, dimension (MMAX,2)

If IJOB=1, then on output NAB(i,j) will be set to N(AB(i,j)).

If IJOB=2, then on input, NAB(i,j) should be set. It must satisfy the condition:

$N(AB(i,1)) \leq NAB(i,1) \leq NAB(i,2) \leq N(AB(i,2))$ ,

which means that in interval i only eigenvalues

NAB(i,1)+1,...,NAB(i,2) will be considered. Usually,

NAB(i,j)=N(AB(i,j)), from a previous call to SLAEBZ with IJOB=1.

On output, NAB(i,j) will contain

$\max(na(k), \min(nb(k), N(AB(i,j))))$ , where k is the index of the input interval that the output interval

(AB(j,1),AB(j,2)] came from, and na(k) and nb(k) are the input values of NAB(k,1) and NAB(k,2).

If IJOB=3, then on output, NAB(i,j) contains N(AB(i,j)), unless  $N(w) > NVAL(i)$  for all search points w, in which case NAB(i,1) will not be modified, i.e., the output value will be the same as the input value (modulo reorderings -- see NVAL and AB), or unless  $N(w) < NVAL(i)$  for all search points w, in which case NAB(i,2) will not be modified. Normally, NAB should be set to some distinctive value(s) before SLAEBZ is called.

### WORK

WORK is REAL array, dimension (MMAX)

Workspace.

### IWORK

IWORK is INTEGER array, dimension (MMAX)

Workspace.

### INFO

INFO is INTEGER

= 0: All intervals converged.

= 1--MMAX: The last INFO intervals did not converge.

= MMAX+1: More than MMAX intervals were generated.

### Author

Univ. of Tennessee





Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

#### Date

December 2016

#### Further Details:

This routine is intended to be called only by other LAPACK routines, thus the interface is less user-friendly. It is intended for two purposes:

(a) finding eigenvalues. In this case, SLAEBZ should have one or more initial intervals set up in AB, and SLAEBZ should be called with IJOB=1. This sets up NAB, and also counts the eigenvalues. Intervals with no eigenvalues would usually be thrown out at this point. Also, if not all the eigenvalues in an interval  $i$  are desired, NAB( $i,1$ ) can be increased or NAB( $i,2$ ) decreased. For example, set NAB( $i,1$ )=NAB( $i,2$ )-1 to get the largest eigenvalue. SLAEBZ is then called with IJOB=2 and MMAX no smaller than the value of MOUT returned by the call with IJOB=1. After this (IJOB=2) call, eigenvalues NAB( $i,1$ )+1 through NAB( $i,2$ ) are approximately AB( $i,1$ ) (or AB( $i,2$ )) to the tolerance specified by ABSTOL and RELTOL.

(b) finding an interval ( $a',b'$ ) containing eigenvalues  $w(f),\dots,w(l)$ . In this case, start with a Gershgorin interval ( $a,b$ ). Set up AB to contain 2 search intervals, both initially ( $a,b$ ). One NVAL element should contain  $f-1$  and the other should contain 1, while C should contain  $a$  and  $b$ , resp. NAB( $i,1$ ) should be -1 and NAB( $i,2$ ) should be  $N+1$ , to flag an error if the desired interval does not lie in ( $a,b$ ). SLAEBZ is then called with IJOB=3. On exit, if  $w(f-1) < w(f)$ , then one of the intervals --  $j$  -- will have AB( $j,1$ )=AB( $j,2$ ) and NAB( $j,1$ )=NAB( $j,2$ )= $f-1$ , while if, to the specified tolerance,  $w(f-k)=\dots=w(f+r)$ ,  $k > 0$  and  $r \geq 0$ , then the interval will have N(AB( $j,1$ ))=NAB( $j,1$ )= $f-k$  and N(AB( $j,2$ ))=NAB( $j,2$ )= $f+r$ . The cases  $w(l) < w(l+1)$  and  $w(l-r)=\dots=w(l+k)$  are handled similarly.

**subroutine slaev2 (real A, real B, real C, real RT1, real RT2, real CS1, real SN1)**

SLAEV2 computes the eigenvalues and eigenvectors of a 2-by-2 symmetric/Hermitian matrix.

#### Purpose:

SLAEV2 computes the eigendecomposition of a 2-by-2 symmetric matrix

$$\begin{bmatrix} A & B \\ B & C \end{bmatrix}$$

$$\begin{bmatrix} B & C \end{bmatrix}.$$

On return, RT1 is the eigenvalue of larger absolute value, RT2 is the eigenvalue of smaller absolute value, and (CS1,SN1) is the unit right eigenvector for RT1, giving the decomposition

$$\begin{bmatrix} CS1 & SN1 \\ -SN1 & CS1 \end{bmatrix} \begin{bmatrix} A & B \\ B & C \end{bmatrix} \begin{bmatrix} CS1 & -SN1 \\ SN1 & CS1 \end{bmatrix} = \begin{bmatrix} RT1 & 0 \\ 0 & RT2 \end{bmatrix}.$$

#### Parameters

A

A is REAL

The (1,1) element of the 2-by-2 matrix.



*B**B* is REAL

The (1,2) element and the conjugate of the (2,1) element of the 2-by-2 matrix.

*C**C* is REAL

The (2,2) element of the 2-by-2 matrix.

*RT1**RT1* is REAL

The eigenvalue of larger absolute value.

*RT2**RT2* is REAL

The eigenvalue of smaller absolute value.

*CS1**CS1* is REAL*SN1**SN1* is REALThe vector (*CS1*, *SN1*) is a unit right eigenvector for *RT1*.**Author**

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

**Date**

December 2016

**Further Details:***RT1* is accurate to a few ulps barring over/underflow.*RT2* may be inaccurate if there is massive cancellation in the determinant  $A^*C-B^*B$ ; higher precision or correctly rounded or correctly truncated arithmetic would be needed to compute *RT2* accurately in all cases.*CS1* and *SN1* are accurate to a few ulps barring over/underflow.Overflow is possible only if *RT1* is within a factor of 5 of overflow.Underflow is harmless if the input data is 0 or exceeds  
underflow\_threshold / macheps.**subroutine slag2d (integer M, integer N, real, dimension( ldsa, \* ) SA, integer LDSA, double precision, dimension( lda, \* ) A, integer LDA, integer INFO)****SLAG2D** converts a single precision matrix to a double precision matrix.**Purpose:****SLAG2D** converts a SINGLE PRECISION matrix, *SA*, to a DOUBLE PRECISION matrix, *A*.

Note that while it is possible to overflow while converting from double to single, it is not possible to overflow when converting from single to double.



This is an auxiliary routine so there is no argument checking.

### Parameters

*M*

*M* is INTEGER

The number of lines of the matrix *A*.  $M \geq 0$ .

*N*

*N* is INTEGER

The number of columns of the matrix *A*.  $N \geq 0$ .

*SA*

*SA* is REAL array, dimension (LDSA,*N*)

On entry, the *M*-by-*N* coefficient matrix *SA*.

*LDSA*

*LDSA* is INTEGER

The leading dimension of the array *SA*.  $LDSA \geq \max(1,M)$ .

*A*

*A* is DOUBLE PRECISION array, dimension (LDA,*N*)

On exit, the *M*-by-*N* coefficient matrix *A*.

*LDA*

*LDA* is INTEGER

The leading dimension of the array *A*.  $LDA \geq \max(1,M)$ .

*INFO*

*INFO* is INTEGER

= 0: successful exit

### Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

### Date

December 2016

**subroutine slagts (integer JOB, integer N, real, dimension( \*) A, real, dimension( \*) B, real, dimension( \*) C, real, dimension( \*) D, integer, dimension( \*) IN, real, dimension( \*) Y, real TOL, integer INFO)**

**SLAGTS** solves the system of equations  $(T - \lambda I)x = y$  or  $(T - \lambda I)Tx = y$ , where *T* is a general tridiagonal matrix and  $\lambda$  a scalar, using the LU factorization computed by **slagtf**.

### Purpose:

**SLAGTS** may be used to solve one of the systems of equations

$$(T - \lambda I)x = y \quad \text{or} \quad (T - \lambda I)Tx = y,$$

where *T* is an *n* by *n* tridiagonal matrix, for *x*, following the factorization of  $(T - \lambda I)$  as

$$(T - \lambda I) = P * L * U,$$

by routine **SLAGTF**. The choice of equation to be solved is controlled by the argument *JOB*, and in each case there is an option to perturb zero or very small diagonal elements of *U*, this option being intended for use in applications such as inverse iteration.



**Parameters***JOB*

*JOB* is INTEGER

Specifies the job to be performed by SLAGTS as follows:

- = 1: The equations  $(T - \lambda I)x = y$  are to be solved, but diagonal elements of *U* are not to be perturbed.
- = -1: The equations  $(T - \lambda I)x = y$  are to be solved and, if overflow would otherwise occur, the diagonal elements of *U* are to be perturbed. See argument *TOL* below.
- = 2: The equations  $(T - \lambda I)^2 x = y$  are to be solved, but diagonal elements of *U* are not to be perturbed.
- = -2: The equations  $(T - \lambda I)^2 x = y$  are to be solved and, if overflow would otherwise occur, the diagonal elements of *U* are to be perturbed. See argument *TOL* below.

*N*

*N* is INTEGER

The order of the matrix *T*.

*A*

*A* is REAL array, dimension (*N*)

On entry, *A* must contain the diagonal elements of *U* as returned from SLAGTF.

*B*

*B* is REAL array, dimension (*N*-1)

On entry, *B* must contain the first super-diagonal elements of *U* as returned from SLAGTF.

*C*

*C* is REAL array, dimension (*N*-1)

On entry, *C* must contain the sub-diagonal elements of *L* as returned from SLAGTF.

*D*

*D* is REAL array, dimension (*N*-2)

On entry, *D* must contain the second super-diagonal elements of *U* as returned from SLAGTF.

*IN*

*IN* is INTEGER array, dimension (*N*)

On entry, *IN* must contain details of the matrix *P* as returned from SLAGTF.

*Y*

*Y* is REAL array, dimension (*N*)

On entry, the right hand side vector *y*.

On exit, *Y* is overwritten by the solution vector *x*.

*TOL*

*TOL* is REAL

On entry, with *JOB* < 0, *TOL* should be the minimum perturbation to be made to very small diagonal elements of *U*.

*TOL* should normally be chosen as about  $\epsilon \cdot \text{norm}(U)$ , where  $\epsilon$  is the relative machine precision, but if *TOL* is supplied as non-positive, then it is reset to  $\epsilon \cdot \max(|u(i,j)|)$ .

If *JOB* > 0 then *TOL* is not referenced.



On exit, TOL is changed as described above, only if TOL is non-positive on entry. Otherwise TOL is unchanged.

#### INFO

INFO is INTEGER

= 0: successful exit

< 0: if INFO = -i, the i-th argument had an illegal value

> 0: overflow would occur when computing the INFO(th) element of the solution vector x. This can only occur when JOB is supplied as positive and either means that a diagonal element of U is very small, or that the elements of the right-hand side vector y are very large.

#### Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

#### Date

December 2016

#### logical function slaisnan (real, intent(in) SIN1, real, intent(in) SIN2)

SLAISNAN tests input for NaN by comparing two arguments for inequality.

#### Purpose:

This routine is not for general use. It exists solely to avoid over-optimization in SISNAN.

SLAISNAN checks for NaNs by comparing its two arguments for inequality. NaN is the only floating-point value where NaN != NaN returns .TRUE. To check for NaNs, pass the same variable as both arguments.

A compiler must assume that the two arguments are not the same variable, and the test will not be optimized away. Interprocedural or whole-program optimization may delete this test. The ISNAN functions will be replaced by the correct Fortran 03 intrinsic once the intrinsic is widely available.

#### Parameters

*SIN1*

SIN1 is REAL

*SIN2*

SIN2 is REAL

Two numbers to compare for inequality.

#### Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

#### Date

June 2017



**integer function slaneg (integer N, real, dimension( \* ) D, real, dimension( \* ) LLD, real SIGMA, real PIVMIN, integer R)**

**SLANEG** computes the Sturm count.

**Purpose:**

SLANEG computes the Sturm count, the number of negative pivots encountered while factoring tridiagonal  $T - \sigma I = L D L^T$ . This implementation works directly on the factors without forming the tridiagonal matrix  $T$ . The Sturm count is also the number of eigenvalues of  $T$  less than  $\sigma$ .

This routine is called from SLARRB.

The current routine does not use the PIVMIN parameter but rather requires IEEE-754 propagation of Infinities and NaNs. This routine also has no input range restrictions but does require default exception handling such that  $x/0$  produces Inf when  $x$  is non-zero, and Inf/Inf produces NaN. For more information, see:

Marques, Riedy, and Voemel, "Benefits of IEEE-754 Features in Modern Symmetric Tridiagonal Eigensolvers," SIAM Journal on Scientific Computing, v28, n5, 2006. DOI 10.1137/050641624 (Tech report version in LAWN 172 with the same title.)

**Parameters**

*N*

*N* is INTEGER  
The order of the matrix.

*D*

*D* is REAL array, dimension (N)  
The N diagonal elements of the diagonal matrix  $D$ .

*LLD*

*LLD* is REAL array, dimension (N-1)  
The (N-1) elements  $L(i)*L(i)*D(i)$ .

*SIGMA*

*SIGMA* is REAL  
Shift amount in  $T - \sigma I = L D L^T$ .

*PIVMIN*

*PIVMIN* is REAL  
The minimum pivot in the Sturm sequence. May be used when zero pivots are encountered on non-IEEE-754 architectures.

*R*

*R* is INTEGER  
The twist index for the twisted factorization that is used for the negcount.

**Author**

Univ. of Tennessee  
Univ. of California Berkeley  
Univ. of Colorado Denver  
NAG Ltd.

**Date**



December 2016

### Contributors:

Osni Marques, LBNL/NERSC, USA

Christof Voemel, University of California, Berkeley, USA

Jason Riedy, University of California, Berkeley, USA

**real function slanst (character NORM, integer N, real, dimension( \* ) D, real, dimension( \* ) E)**

**SLANST** returns the value of the 1-norm, or the Frobenius norm, or the infinity norm, or the element of largest absolute value of a real symmetric tridiagonal matrix.

### Purpose:

**SLANST** returns the value of the one norm, or the Frobenius norm, or the infinity norm, or the element of largest absolute value of a real symmetric tridiagonal matrix A.

### Returns

**SLANST**

```
SLANST = ( max(abs(A(i,j))), NORM = 'M' or 'm'
          (
            ( norm1(A),      NORM = '1', 'O' or 'o'
              (
                ( normI(A),   NORM = 'I' or 'i'
                  (
                    ( normF(A),   NORM = 'F', 'f', 'E' or 'e'
```

where **norm1** denotes the one norm of a matrix (maximum column sum), **normI** denotes the infinity norm of a matrix (maximum row sum) and **normF** denotes the Frobenius norm of a matrix (square root of sum of squares). Note that **max(abs(A(i,j)))** is not a consistent matrix norm.

### Parameters

**NORM**

**NORM** is CHARACTER\*1

Specifies the value to be returned in **SLANST** as described above.

**N**

**N** is INTEGER

The order of the matrix A.  $N \geq 0$ . When  $N = 0$ , **SLANST** is set to zero.

**D**

**D** is REAL array, dimension (N)

The diagonal elements of A.

**E**

**E** is REAL array, dimension (N-1)

The (n-1) sub-diagonal or super-diagonal elements of A.

### Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

### Date

December 2016



**real function slapy2 (real X, real Y)****SLAPY2** returns  $\sqrt{x^2+y^2}$ .**Purpose:**

SLAPY2 returns  $\sqrt{x^2+y^2}$ , taking care not to cause unnecessary overflow.

**Parameters***X**X* is REAL*Y**Y* is REAL*X* and *Y* specify the values *x* and *y*.**Author**

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

**Date**

June 2017

**real function slapy3 (real X, real Y, real Z)****SLAPY3** returns  $\sqrt{x^2+y^2+z^2}$ .**Purpose:**

SLAPY3 returns  $\sqrt{x^2+y^2+z^2}$ , taking care not to cause unnecessary overflow.

**Parameters***X**X* is REAL*Y**Y* is REAL*Z**Z* is REAL*X*, *Y* and *Z* specify the values *x*, *y* and *z*.**Author**

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

**Date**

December 2016

**subroutine slarnv (integer IDIST, integer, dimension( 4 ) ISEED, integer N, real, dimension( \* ) X)****SLARNV** returns a vector of random numbers from a uniform or normal distribution.**Purpose:**

SLARNV returns a vector of *n* random real numbers from a uniform or normal distribution.

**Parameters**



**IDIST**

IDIST is INTEGER

Specifies the distribution of the random numbers:

= 1: uniform (0,1)

= 2: uniform (-1,1)

= 3: normal (0,1)

**ISEED**

ISEED is INTEGER array, dimension (4)

On entry, the seed of the random number generator; the array elements must be between 0 and 4095, and ISEED(4) must be odd.

On exit, the seed is updated.

**N**

N is INTEGER

The number of random numbers to be generated.

**X**

X is REAL array, dimension (N)

The generated random numbers.

**Author**

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

**Date**

December 2016

**Further Details:**

This routine calls the auxiliary routine SLARUV to generate random real numbers from a uniform (0,1) distribution, in batches of up to 128 using vectorisable code. The Box-Muller method is used to transform numbers from a uniform to a normal distribution.

**subroutine slarra (integer N, real, dimension( \* ) D, real, dimension( \* ) E, real, dimension( \* ) E2, real SPLTOL, real TNRM, integer NSPLIT, integer, dimension( \* ) ISPLIT, integer INFO)**  
**SLARRA** computes the splitting points with the specified threshold.

**Purpose:**

Compute the splitting points with threshold SPLTOL.

SLARRA sets any "small" off-diagonal elements to zero.

**Parameters****N**

N is INTEGER

The order of the matrix.  $N > 0$ .

**D**

D is REAL array, dimension (N)

On entry, the N diagonal elements of the tridiagonal matrix T.

**E**

E is REAL array, dimension (N)

On entry, the first (N-1) entries contain the subdiagonal



elements of the tridiagonal matrix  $T$ ;  $E(N)$  need not be set.  
 On exit, the entries  $E( ISPLIT( I ) )$ ,  $1 \leq I \leq NSPLIT$ ,  
 are set to zero, the other entries of  $E$  are untouched.

***E2***

$E2$  is REAL array, dimension  $(N)$   
 On entry, the first  $(N-1)$  entries contain the SQUARES of the  
 subdiagonal elements of the tridiagonal matrix  $T$ ;  
 $E2(N)$  need not be set.  
 On exit, the entries  $E2( ISPLIT( I ) )$ ,  
 $1 \leq I \leq NSPLIT$ , have been set to zero

***SPLTOL***

$SPLTOL$  is REAL  
 The threshold for splitting. Two criteria can be used:  
 $SPLTOL < 0$  : criterion based on absolute off-diagonal value  
 $SPLTOL > 0$  : criterion that preserves relative accuracy

***TNRM***

$TNRM$  is REAL  
 The norm of the matrix.

***NSPLIT***

$NSPLIT$  is INTEGER  
 The number of blocks  $T$  splits into.  $1 \leq NSPLIT \leq N$ .

***ISPLIT***

$ISPLIT$  is INTEGER array, dimension  $(N)$   
 The splitting points, at which  $T$  breaks up into blocks.  
 The first block consists of rows/columns 1 to  $ISPLIT(1)$ ,  
 the second of rows/columns  $ISPLIT(1)+1$  through  $ISPLIT(2)$ ,  
 etc., and the  $NSPLIT$ -th consists of rows/columns  
 $ISPLIT(NSPLIT-1)+1$  through  $ISPLIT(NSPLIT)=N$ .

***INFO***

$INFO$  is INTEGER  
 $= 0$ : successful exit

**Author**

Univ. of Tennessee  
 Univ. of California Berkeley  
 Univ. of Colorado Denver  
 NAG Ltd.

**Date**

June 2017

**Contributors:**

Beresford Parlett, University of California, Berkeley, USA  
 Jim Demmel, University of California, Berkeley, USA  
 Inderjit Dhillon, University of Texas, Austin, USA  
 Osni Marques, LBNL/NERSC, USA  
 Christof Voemel, University of California, Berkeley, USA

**subroutine slarrb (integer N, real, dimension( \*) D, real, dimension( \*) LLD, integer IFIRST, integer ILAST, real RTOL1, real RTOL2, integer OFFSET, real, dimension( \*) W, real, dimension( \*) WGAP, real, dimension( \*) WERR, real, dimension( \*) WORK, integer, dimension( \*) IWORK, real PIVMIN, real SPDIAM, integer TWIST, integer INFO)**  
**SLARRB** provides limited bisection to locate eigenvalues for more accuracy.

**Purpose:**

Given the relatively robust representation  $(RRR) L D L^T$ , SLARRB does "limited" bisection to refine the eigenvalues of  $L D L^T$ ,  $W( IFIRST-OFFSET )$  through  $W( ILAST-OFFSET )$ , to more accuracy. Initial guesses for these eigenvalues are input in  $W$ , the corresponding estimate of the error in these guesses and their gaps are input in  $WERR$  and  $WGAP$ , respectively. During bisection, intervals  $[left, right]$  are maintained by storing their mid-points and semi-widths in the arrays  $W$  and  $WERR$  respectively.

### Parameters

*N*

*N* is INTEGER  
The order of the matrix.

*D*

*D* is REAL array, dimension (*N*)  
The *N* diagonal elements of the diagonal matrix *D*.

*LLD*

*LLD* is REAL array, dimension (*N*-1)  
The (*N*-1) elements  $L(i)*L(i)*D(i)$ .

*IFIRST*

*IFIRST* is INTEGER  
The index of the first eigenvalue to be computed.

*ILAST*

*ILAST* is INTEGER  
The index of the last eigenvalue to be computed.

*RTOL1*

*RTOL1* is REAL

*RTOL2*

*RTOL2* is REAL  
Tolerance for the convergence of the bisection intervals.  
An interval  $[LEFT, RIGHT]$  has converged if  
 $RIGHT-LEFT < \text{MAX}( RTOL1 * GAP, RTOL2 * \text{MAX}(|LEFT|, |RIGHT|) )$   
where *GAP* is the (estimated) distance to the nearest eigenvalue.

*OFFSET*

*OFFSET* is INTEGER  
Offset for the arrays  $W$ ,  $WGAP$  and  $WERR$ , i.e., the *IFIRST-OFFSET* through *ILAST-OFFSET* elements of these arrays are to be used.

*W*

*W* is REAL array, dimension (*N*)  
On input,  $W( IFIRST-OFFSET )$  through  $W( ILAST-OFFSET )$  are estimates of the eigenvalues of  $L D L^T$  indexed *IFIRST* through *ILAST*.  
On output, these estimates are refined.

*WGAP*

*WGAP* is REAL array, dimension (*N*-1)  
On input, the (estimated) gaps between consecutive eigenvalues of  $L D L^T$ , i.e.,  $WGAP(I-OFFSET)$  is the gap between eigenvalues *I* and *I*+1. Note that if *IFIRST* = *ILAST* then  $WGAP(IFIRST-OFFSET)$  must be set to ZERO.  
On output, these gaps are refined.



**WERR**

WERR is REAL array, dimension (N)  
 On input, WERR( IFIRST-OFFSET ) through WERR( ILAST-OFFSET ) are  
 the errors in the estimates of the corresponding elements in W.  
 On output, these errors are refined.

**WORK**

WORK is REAL array, dimension (2\*N)  
 Workspace.

**IWORK**

IWORK is INTEGER array, dimension (2\*N)  
 Workspace.

**PIVMIN**

PIVMIN is REAL  
 The minimum pivot in the Sturm sequence.

**SPDIAM**

SPDIAM is REAL  
 The spectral diameter of the matrix.

**TWIST**

TWIST is INTEGER  
 The twist index for the twisted factorization that is used  
 for the negcount.  
 TWIST = N: Compute negcount from  $L D L^T - \text{LAMBDA } I = L + D + L^T$   
 TWIST = 1: Compute negcount from  $L D L^T - \text{LAMBDA } I = U - D - U^T$   
 TWIST = R: Compute negcount from  $L D L^T - \text{LAMBDA } I = N(r) D(r) N(r)$

**INFO**

INFO is INTEGER  
 Error flag.

**Author**

Univ. of Tennessee  
 Univ. of California Berkeley  
 Univ. of Colorado Denver  
 NAG Ltd.

**Date**

June 2017

**Contributors:**

Beresford Parlett, University of California, Berkeley, USA  
 Jim Demmel, University of California, Berkeley, USA  
 Inderjit Dhillon, University of Texas, Austin, USA  
 Osni Marques, LBNL/NERSC, USA  
 Christof Voemel, University of California, Berkeley, USA

**subroutine slarrc (character JOBT, integer N, real VL, real VU, real, dimension( \* ) D, real,  
 dimension( \* ) E, real PIVMIN, integer EIGCNT, integer LCNT, integer RCNT, integer INFO)**  
**SLARRC** computes the number of eigenvalues of the symmetric tridiagonal matrix.

**Purpose:**

Find the number of eigenvalues of the symmetric tridiagonal matrix T  
 that are in the interval (VL,VU] if JOBT = 'T', and of  $L D L^T$   
 if JOBT = 'L'.

**Parameters**

*JOBT*

JOBT is CHARACTER\*1

= 'T': Compute Sturm count for matrix T.

= 'L': Compute Sturm count for matrix  $L D L^T$ .

*N*

N is INTEGER

The order of the matrix.  $N > 0$ .

*VL*

VL is REAL

The lower bound for the eigenvalues.

*VU*

VU is REAL

The upper bound for the eigenvalues.

*D*

D is REAL array, dimension (N)

JOBT = 'T': The N diagonal elements of the tridiagonal matrix T.

JOBT = 'L': The N diagonal elements of the diagonal matrix D.

*E*

E is REAL array, dimension (N)

JOBT = 'T': The N-1 offdiagonal elements of the matrix T.

JOBT = 'L': The N-1 offdiagonal elements of the matrix L.

*PIVMIN*

PIVMIN is REAL

The minimum pivot in the Sturm sequence for T.

*EIGCNT*

EIGCNT is INTEGER

The number of eigenvalues of the symmetric tridiagonal matrix T that are in the interval (VL,VU]

*LCNT*

LCNT is INTEGER

*RCNT*

RCNT is INTEGER

The left and right negcounts of the interval.

*INFO*

INFO is INTEGER

**Author**

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

**Date**

June 2016

**Contributors:**

Beresford Parlett, University of California, Berkeley, USA

Jim Demmel, University of California, Berkeley, USA

Inderjit Dhillon, University of Texas, Austin, USA

Osni Marques, LBNL/NERSC, USA

Christof Voemel, University of California, Berkeley, USA



**subroutine slarrd** (character **RANGE**, character **ORDER**, integer **N**, real **VL**, real **VU**, integer **IL**, integer **IU**, real, dimension( \*) **GERS**, real **RELTOL**, real, dimension( \*) **D**, real, dimension( \*) **E**, real, dimension( \*) **E2**, real **PIVMIN**, integer **NSPLIT**, integer, dimension( \*) **ISPLIT**, integer **M**, real, dimension( \*) **W**, real, dimension( \*) **WERR**, real **WL**, real **WU**, integer, dimension( \*) **IBLOCK**, integer, dimension( \*) **INDEXW**, real, dimension( \*) **WORK**, integer, dimension( \*) **IWORK**, integer **INFO**)

**SLARRD** computes the eigenvalues of a symmetric tridiagonal matrix to suitable accuracy.

#### **Purpose:**

SLARRD computes the eigenvalues of a symmetric tridiagonal matrix **T** to suitable accuracy. This is an auxiliary code to be called from **SSTEMR**.

The user may ask for all eigenvalues, all eigenvalues in the half-open interval (**VL**, **VU**], or the **IL**-th through **IU**-th eigenvalues.

To avoid overflow, the matrix must be scaled so that its largest element is no greater than  $\text{overflow}^{1/2} * \text{underflow}^{1/4}$  in absolute value, and for greatest accuracy, it should not be much smaller than that.

See W. Kahan "Accurate Eigenvalues of a Symmetric Tridiagonal Matrix", Report CS41, Computer Science Dept., Stanford University, July 21, 1966.

#### **Parameters**

##### *RANGE*

**RANGE** is CHARACTER\*1  
 = 'A': ("All") all eigenvalues will be found.  
 = 'V': ("Value") all eigenvalues in the half-open interval (**VL**, **VU**] will be found.  
 = 'I': ("Index") the **IL**-th through **IU**-th eigenvalues (of the entire matrix) will be found.

##### *ORDER*

**ORDER** is CHARACTER\*1  
 = 'B': ("By Block") the eigenvalues will be grouped by split-off block (see **IBLOCK**, **ISPLIT**) and ordered from smallest to largest within the block.  
 = 'E': ("Entire matrix") the eigenvalues for the entire matrix will be ordered from smallest to largest.

##### *N*

**N** is INTEGER  
 The order of the tridiagonal matrix **T**.  $N \geq 0$ .

##### *VL*

**VL** is REAL  
 If **RANGE**= 'V', the lower bound of the interval to be searched for eigenvalues. Eigenvalues less than or equal to **VL**, or greater than **VU**, will not be returned.  $VL < VU$ .  
 Not referenced if **RANGE** = 'A' or 'I'.

##### *VU*

**VU** is REAL  
 If **RANGE**= 'V', the upper bound of the interval to be searched for eigenvalues. Eigenvalues less than or equal



to VL, or greater than VU, will not be returned.  $VL < VU$ .  
Not referenced if RANGE = 'A' or 'I'.

*IL*

IL is INTEGER

If RANGE='I', the index of the  
smallest eigenvalue to be returned.

$1 \leq IL \leq IU \leq N$ , if  $N > 0$ ;  $IL = 1$  and  $IU = 0$  if  $N = 0$ .

Not referenced if RANGE = 'A' or 'V'.

*IU*

IU is INTEGER

If RANGE='I', the index of the  
largest eigenvalue to be returned.

$1 \leq IL \leq IU \leq N$ , if  $N > 0$ ;  $IL = 1$  and  $IU = 0$  if  $N = 0$ .

Not referenced if RANGE = 'A' or 'V'.

*GERS*

GERS is REAL array, dimension (2\*N)

The N Gerschgorin intervals (the i-th Gerschgorin interval  
is (GERS(2\*i-1), GERS(2\*i)).

*RELTOL*

RELTOL is REAL

The minimum relative width of an interval. When an interval  
is narrower than RELTOL times the larger (in  
magnitude) endpoint, then it is considered to be  
sufficiently small, i.e., converged. Note: this should  
always be at least radix\*machine epsilon.

*D*

D is REAL array, dimension (N)

The n diagonal elements of the tridiagonal matrix T.

*E*

E is REAL array, dimension (N-1)

The (n-1) off-diagonal elements of the tridiagonal matrix T.

*E2*

E2 is REAL array, dimension (N-1)

The (n-1) squared off-diagonal elements of the tridiagonal matrix T.

*PIVMIN*

PIVMIN is REAL

The minimum pivot allowed in the Sturm sequence for T.

*NSPLIT*

NSPLIT is INTEGER

The number of diagonal blocks in the matrix T.

$1 \leq NSPLIT \leq N$ .

*ISPLIT*

ISPLIT is INTEGER array, dimension (N)

The splitting points, at which T breaks up into submatrices.

The first submatrix consists of rows/columns 1 to ISPLIT(1),  
the second of rows/columns ISPLIT(1)+1 through ISPLIT(2),  
etc., and the NSPLIT-th consists of rows/columns  
ISPLIT(NSPLIT-1)+1 through ISPLIT(NSPLIT)=N.

(Only the first NSPLIT elements will actually be used, but  
since the user cannot know a priori what value NSPLIT will  
have, N words must be reserved for ISPLIT.)



*M*

*M* is INTEGER

The actual number of eigenvalues found.  $0 \leq M \leq N$ .  
(See also the description of INFO=2,3.)

*W*

*W* is REAL array, dimension (N)

On exit, the first *M* elements of *W* will contain the eigenvalue approximations. SLARRD computes an interval  $I_j = (a_j, b_j]$  that includes eigenvalue *j*. The eigenvalue approximation is given as the interval midpoint  $W(j) = (a_j + b_j)/2$ . The corresponding error is bounded by  $WERR(j) = \text{abs}(a_j - b_j)/2$

*WERR*

*WERR* is REAL array, dimension (N)

The error bound on the corresponding eigenvalue approximation in *W*.

*WL*

*WL* is REAL

*WU*

*WU* is REAL

The interval (*WL*, *WU*] contains all the wanted eigenvalues.

If RANGE='V', then *WL*=*VL* and *WU*=*VU*.

If RANGE='A', then *WL* and *WU* are the global Gerschgorin bounds on the spectrum.

If RANGE='I', then *WL* and *WU* are computed by SLAEBZ from the index range specified.

*IBLOCK*

*IBLOCK* is INTEGER array, dimension (N)

At each row/column *j* where *E(j)* is zero or small, the matrix *T* is considered to split into a block diagonal matrix. On exit, if INFO = 0, *IBLOCK(i)* specifies to which block (from 1 to the number of blocks) the eigenvalue *W(i)* belongs. (SLARRD may use the remaining N-M elements as workspace.)

*INDEXW*

*INDEXW* is INTEGER array, dimension (N)

The indices of the eigenvalues within each block (submatrix); for example, *INDEXW(i)=j* and *IBLOCK(i)=k* imply that the *i*-th eigenvalue *W(i)* is the *j*-th eigenvalue in block *k*.

*WORK*

*WORK* is REAL array, dimension (4\*N)

*IWORK*

*IWORK* is INTEGER array, dimension (3\*N)

*INFO*

INFO is INTEGER

= 0: successful exit

< 0: if INFO = -i, the *i*-th argument had an illegal value

> 0: some or all of the eigenvalues failed to converge or were not computed:

=1 or 3: Bisection failed to converge for some eigenvalues; these eigenvalues are flagged by a





negative block number. The effect is that the eigenvalues may not be as accurate as the absolute and relative tolerances. This is generally caused by unexpectedly inaccurate arithmetic.

=2 or 3: RANGE='I' only: Not all of the eigenvalues IL:IU were found.

Effect:  $M < IU + 1 - IL$

Cause: non-monotonic arithmetic, causing the Sturm sequence to be non-monotonic.

Cure: recalculate, using RANGE='A', and pick out eigenvalues IL:IU. In some cases, increasing the PARAMETER "FUDGE" may make things work.

= 4: RANGE='I', and the Gershgorin interval initially used was too small. No eigenvalues were computed.

Probable cause: your machine has sloppy floating-point arithmetic.

Cure: Increase the PARAMETER "FUDGE", recompile, and try again.

### Internal Parameters:

FUDGE REAL, default = 2

A "fudge factor" to widen the Gershgorin intervals. Ideally, a value of 1 should work, but on machines with sloppy arithmetic, this needs to be larger. The default for publicly released versions should be large enough to handle the worst machine around. Note that this has no effect on accuracy of the solution.

### Contributors:

W. Kahan, University of California, Berkeley, USA  
 Beresford Parlett, University of California, Berkeley, USA  
 Jim Demmel, University of California, Berkeley, USA  
 Interjit Dhillon, University of Texas, Austin, USA  
 Osni Marques, LBNL/NERSC, USA  
 Christof Voemel, University of California, Berkeley, USA

### Author

Univ. of Tennessee  
 Univ. of California Berkeley  
 Univ. of Colorado Denver  
 NAG Ltd.

### Date

June 2016

subroutine slarre (character RANGE, integer N, real VL, real VU, integer IL, integer IU, real, dimension( \*) D, real, dimension( \*) E, real, dimension( \*) E2, real RTOL1, real RTOL2, real SPLTOL, integer NSPLIT, integer, dimension( \*) ISPLIT, integer M, real, dimension( \*) W, real, dimension( \*) WERR, real, dimension( \*) WGAP, integer, dimension( \*) IBLOCK, integer, dimension( \*) INDEXW, real, dimension( \*) GERS, real PIVMIN, real, dimension( \*) WORK, integer, dimension( \*) IWORK, integer INFO)

SLARRE given the tridiagonal matrix T, sets small off-diagonal elements to zero and for each unreduced block Ti, finds base representations and eigenvalues.

### Purpose:



To find the desired eigenvalues of a given real symmetric tridiagonal matrix  $T$ , SLARRE sets any "small" off-diagonal elements to zero, and for each unreduced block  $T_i$ , it finds

- (a) a suitable shift at one end of the block's spectrum,
- (b) the base representation,  $T_i - \sigma_i I = L_i D_i L_i^T$ , and
- (c) eigenvalues of each  $L_i D_i L_i^T$ .

The representations and eigenvalues found are then used by SSTEMR to compute the eigenvectors of  $T$ .

The accuracy varies depending on whether bisection is used to find a few eigenvalues or the dqds algorithm (subroutine SLASQ2) to compute all and then discard any unwanted one.

As an added benefit, SLARRE also outputs the  $n$  Gerschgorin intervals for the matrices  $L_i D_i L_i^T$ .

### Parameters

#### *RANGE*

RANGE is CHARACTER\*1  
 = 'A': ("All") all eigenvalues will be found.  
 = 'V': ("Value") all eigenvalues in the half-open interval  $(VL, VU]$  will be found.  
 = 'I': ("Index") the  $IL$ -th through  $IU$ -th eigenvalues (of the entire matrix) will be found.

#### *N*

$N$  is INTEGER  
 The order of the matrix.  $N > 0$ .

#### *VL*

$VL$  is REAL  
 If RANGE='V', the lower bound for the eigenvalues.  
 Eigenvalues less than or equal to  $VL$ , or greater than  $VU$ , will not be returned.  $VL < VU$ .  
 If RANGE='I' or 'A', SLARRE computes bounds on the desired part of the spectrum.

#### *VU*

$VU$  is REAL  
 If RANGE='V', the upper bound for the eigenvalues.  
 Eigenvalues less than or equal to  $VL$ , or greater than  $VU$ , will not be returned.  $VL < VU$ .  
 If RANGE='I' or 'A', SLARRE computes bounds on the desired part of the spectrum.

#### *IL*

$IL$  is INTEGER  
 If RANGE='I', the index of the smallest eigenvalue to be returned.  
 $1 \leq IL \leq IU \leq N$ .

#### *IU*

$IU$  is INTEGER  
 If RANGE='I', the index of the largest eigenvalue to be returned.  
 $1 \leq IL \leq IU \leq N$ .

#### *D*

$D$  is REAL array, dimension ( $N$ )  
 On entry, the  $N$  diagonal elements of the tridiagonal matrix  $T$ .  
 On exit, the  $N$  diagonal elements of the diagonal



matrices  $D_i$ .

*E*

*E* is REAL array, dimension (N)

On entry, the first (N-1) entries contain the subdiagonal elements of the tridiagonal matrix *T*; *E*(N) need not be set.

On exit, *E* contains the subdiagonal elements of the unit bidiagonal matrices  $L_i$ . The entries *E*( ISPLIT( *I* ) ),  $1 \leq I \leq \text{NSPLIT}$ , contain the base points  $\sigma_i$  on output.

*E2*

*E2* is REAL array, dimension (N)

On entry, the first (N-1) entries contain the SQUARES of the subdiagonal elements of the tridiagonal matrix *T*;

*E2*(N) need not be set.

On exit, the entries *E2*( ISPLIT( *I* ) ),

$1 \leq I \leq \text{NSPLIT}$ , have been set to zero

*RTOL1*

*RTOL1* is REAL

*RTOL2*

*RTOL2* is REAL

Parameters for bisection.

An interval [LEFT,RIGHT] has converged if

$\text{RIGHT} - \text{LEFT} < \text{MAX}( \text{RTOL1} * \text{GAP}, \text{RTOL2} * \text{MAX}(|\text{LEFT}|, |\text{RIGHT}|) )$

*SPLTOL*

*SPLTOL* is REAL

The threshold for splitting.

*NSPLIT*

*NSPLIT* is INTEGER

The number of blocks *T* splits into.  $1 \leq \text{NSPLIT} \leq N$ .

*ISPLIT*

*ISPLIT* is INTEGER array, dimension (N)

The splitting points, at which *T* breaks up into blocks.

The first block consists of rows/columns 1 to *ISPLIT*(1), the second of rows/columns *ISPLIT*(1)+1 through *ISPLIT*(2), etc., and the *NSPLIT*-th consists of rows/columns *ISPLIT*(*NSPLIT*-1)+1 through *ISPLIT*(*NSPLIT*)=N.

*M*

*M* is INTEGER

The total number of eigenvalues (of all  $L_i D_i L_i^T$ ) found.

*W*

*W* is REAL array, dimension (N)

The first *M* elements contain the eigenvalues. The eigenvalues of each of the blocks,  $L_i D_i L_i^T$ , are sorted in ascending order ( SLARRE may use the remaining N-M elements as workspace).

*WERR*

*WERR* is REAL array, dimension (N)

The error bound on the corresponding eigenvalue in *W*.

*WGAP*

*WGAP* is REAL array, dimension (N)



The separation from the right neighbor eigenvalue in  $W$ .  
 The gap is only with respect to the eigenvalues of the same block  
 as each block has its own representation tree.  
 Exception: at the right end of a block we store the left gap

**IBLOCK**

IBLOCK is INTEGER array, dimension (N)  
 The indices of the blocks (submatrices) associated with the  
 corresponding eigenvalues in  $W$ ; IBLOCK(i)=1 if eigenvalue  
 $W(i)$  belongs to the first block from the top, =2 if  $W(i)$   
 belongs to the second block, etc.

**INDEXW**

INDEXW is INTEGER array, dimension (N)  
 The indices of the eigenvalues within each block (submatrix);  
 for example, INDEXW(i)= 10 and IBLOCK(i)=2 imply that the  
 i-th eigenvalue  $W(i)$  is the 10-th eigenvalue in block 2

**GERS**

GERS is REAL array, dimension (2\*N)  
 The N Gerschgorin intervals (the i-th Gerschgorin interval  
 is (GERS(2\*i-1), GERS(2\*i)).

**PIVMIN**

PIVMIN is REAL  
 The minimum pivot in the Sturm sequence for  $T$ .

**WORK**

WORK is REAL array, dimension (6\*N)  
 Workspace.

**IWORK**

IWORK is INTEGER array, dimension (5\*N)  
 Workspace.

**INFO**

INFO is INTEGER  
 = 0: successful exit  
 > 0: A problem occurred in SLARRE.  
 < 0: One of the called subroutines signaled an internal problem.  
       Needs inspection of the corresponding parameter IINFO  
       for further information.  
 =-1: Problem in SLARRD.  
 = 2: No base representation could be found in MAXTRY iterations.  
       Increasing MAXTRY and recompilation might be a remedy.  
 =-3: Problem in SLARRB when computing the refined root  
       representation for SLASQ2.  
 =-4: Problem in SLARRB when performing bisection on the  
       desired part of the spectrum.  
 =-5: Problem in SLASQ2.  
 =-6: Problem in SLASQ2.

**Author**

Univ. of Tennessee  
 Univ. of California Berkeley  
 Univ. of Colorado Denver  
 NAG Ltd.

**Date**

June 2016

**Further Details:**

The base representations are required to suffer very little element growth and consequently define all their eigenvalues to high relative accuracy.

**Contributors:**

Beresford Parlett, University of California, Berkeley, USA  
 Jim Demmel, University of California, Berkeley, USA  
 Inderjit Dhillon, University of Texas, Austin, USA  
 Osni Marques, LBNL/NERSC, USA  
 Christof Voemel, University of California, Berkeley, USA

**subroutine slarrf (integer N, real, dimension( \* ) D, real, dimension( \* ) L, real, dimension( \* ) LD, integer CLSTRT, integer CLEND, real, dimension( \* ) W, real, dimension( \* ) WGAP, real, dimension( \* ) WERR, real SPDIA, real CLGAPL, real CLGAPR, real PIVMIN, real SIGMA, real, dimension( \* ) DPLUS, real, dimension( \* ) LPLUS, real, dimension( \* ) WORK, integer INFO)**

**SLARRF** finds a new relatively robust representation such that at least one of the eigenvalues is relatively isolated.

**Purpose:**

Given the initial representation  $L D L^T$  and its cluster of close eigenvalues (in a relative measure),  $W( CLSTRT )$ ,  $W( CLSTRT+1 )$ , ...,  $W( CLEND )$ , **SLARRF** finds a new relatively robust representation  $L D L^T - SIGMA I = L(+) D(+) L(+)^T$  such that at least one of the eigenvalues of  $L(+) D(+) L(+)^T$  is relatively isolated.

**Parameters***N*

*N* is INTEGER  
 The order of the matrix (subblock, if the matrix split).

*D*

*D* is REAL array, dimension (N)  
 The N diagonal elements of the diagonal matrix *D*.

*L*

*L* is REAL array, dimension (N-1)  
 The (N-1) subdiagonal elements of the unit bidiagonal matrix *L*.

*LD*

*LD* is REAL array, dimension (N-1)  
 The (N-1) elements  $L(i)*D(i)$ .

*CLSTRT*

*CLSTRT* is INTEGER  
 The index of the first eigenvalue in the cluster.

*CLEND*

*CLEND* is INTEGER  
 The index of the last eigenvalue in the cluster.

*W*

*W* is REAL array, dimension  
 dimension is  $\geq (CLEND-CLSTRT+1)$   
 The eigenvalue APPROXIMATIONS of  $L D L^T$  in ascending order.  
 $W( CLSTRT )$  through  $W( CLEND )$  form the cluster of relatively



close eigenvalues.

#### *WGAP*

WGAP is REAL array, dimension  
dimension is  $\geq (\text{CLEND}-\text{CLSTRT}+1)$   
The separation from the right neighbor eigenvalue in W.

#### *WERR*

WERR is REAL array, dimension  
dimension is  $\geq (\text{CLEND}-\text{CLSTRT}+1)$   
WERR contain the semiwidth of the uncertainty  
interval of the corresponding eigenvalue APPROXIMATION in W

#### *SPDIAM*

SPDIAM is REAL  
estimate of the spectral diameter obtained from the  
Gerschgorin intervals

#### *CLGAPL*

CLGAPL is REAL

#### *CLGAPR*

CLGAPR is REAL  
absolute gap on each end of the cluster.  
Set by the calling routine to protect against shifts too close  
to eigenvalues outside the cluster.

#### *PIVMIN*

PIVMIN is REAL  
The minimum pivot allowed in the Sturm sequence.

#### *SIGMA*

SIGMA is REAL  
The shift used to form  $L(+) D(+) L(+)^T$ .

#### *DPLUS*

DPLUS is REAL array, dimension (N)  
The N diagonal elements of the diagonal matrix D(+).

#### *LPLUS*

LPLUS is REAL array, dimension (N-1)  
The first (N-1) elements of LPLUS contain the subdiagonal  
elements of the unit bidiagonal matrix L(+).

#### *WORK*

WORK is REAL array, dimension (2\*N)  
Workspace.

#### *INFO*

INFO is INTEGER  
Signals processing OK (=0) or failure (=1)

#### **Author**

Univ. of Tennessee  
Univ. of California Berkeley  
Univ. of Colorado Denver  
NAG Ltd.

#### **Date**

June 2016



**Contributors:**

Beresford Parlett, University of California, Berkeley, USA  
 Jim Demmel, University of California, Berkeley, USA  
 Inderjit Dhillon, University of Texas, Austin, USA  
 Osni Marques, LBNL/NERSC, USA  
 Christof Voemel, University of California, Berkeley, USA

**subroutine slarrj (integer N, real, dimension( \* ) D, real, dimension( \* ) E2, integer IFIRST, integer ILAST, real RTOL, integer OFFSET, real, dimension( \* ) W, real, dimension( \* ) WERR, real, dimension( \* ) WORK, integer, dimension( \* ) IWORK, real PIVMIN, real SPDIAM, integer INFO)**

**SLARRJ** performs refinement of the initial estimates of the eigenvalues of the matrix T.

**Purpose:**

Given the initial eigenvalue approximations of T, SLARRJ does bisection to refine the eigenvalues of T, W( IFIRST-OFFSET ) through W( ILAST-OFFSET ), to more accuracy. Initial guesses for these eigenvalues are input in W, the corresponding estimate of the error in these guesses in WERR. During bisection, intervals [left, right] are maintained by storing their mid-points and semi-widths in the arrays W and WERR respectively.

**Parameters**

*N*

N is INTEGER  
 The order of the matrix.

*D*

D is REAL array, dimension (N)  
 The N diagonal elements of T.

*E2*

E2 is REAL array, dimension (N-1)  
 The Squares of the (N-1) subdiagonal elements of T.

*IFIRST*

IFIRST is INTEGER  
 The index of the first eigenvalue to be computed.

*ILAST*

ILAST is INTEGER  
 The index of the last eigenvalue to be computed.

*RTOL*

RTOL is REAL  
 Tolerance for the convergence of the bisection intervals.  
 An interval [LEFT,RIGHT] has converged if  
 $RIGHT-LEFT < RTOL * MAX(|LEFT|, |RIGHT|)$ .

*OFFSET*

OFFSET is INTEGER  
 Offset for the arrays W and WERR, i.e., the IFIRST-OFFSET through ILAST-OFFSET elements of these arrays are to be used.

*W*

W is REAL array, dimension (N)  
 On input, W( IFIRST-OFFSET ) through W( ILAST-OFFSET ) are estimates of the eigenvalues of  $L D L^T$  indexed IFIRST through ILAST.  
 On output, these estimates are refined.



*WERR*

WERR is REAL array, dimension (N)

On input, WERR( IFIRST-OFFSET ) through WERR( ILAST-OFFSET ) are the errors in the estimates of the corresponding elements in W.

On output, these errors are refined.

*WORK*

WORK is REAL array, dimension (2\*N)

Workspace.

*IWORK*

IWORK is INTEGER array, dimension (2\*N)

Workspace.

*PIVMIN*

PIVMIN is REAL

The minimum pivot in the Sturm sequence for T.

*SPDIAM*

SPDIAM is REAL

The spectral diameter of T.

*INFO*

INFO is INTEGER

Error flag.

**Author**

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

**Date**

June 2017

**Contributors:**

Beresford Parlett, University of California, Berkeley, USA

Jim Demmel, University of California, Berkeley, USA

Inderjit Dhillon, University of Texas, Austin, USA

Osni Marques, LBNL/NERSC, USA

Christof Voemel, University of California, Berkeley, USA

**subroutine slarrk (integer N, integer IW, real GL, real GU, real, dimension( \* ) D, real, dimension( \* )**

**E2, real PIVMIN, real RELTOL, real W, real WERR, integer INFO)**

**SLARRK** computes one eigenvalue of a symmetric tridiagonal matrix T to suitable accuracy.

**Purpose:**

SLARRK computes one eigenvalue of a symmetric tridiagonal matrix T to suitable accuracy. This is an auxiliary code to be called from SSTEMR.

To avoid overflow, the matrix must be scaled so that its largest element is no greater than  $\text{overflow}^{**}(1/2) * \text{underflow}^{**}(1/4)$  in absolute value, and for greatest accuracy, it should not be much smaller than that.

See W. Kahan "Accurate Eigenvalues of a Symmetric Tridiagonal Matrix", Report CS41, Computer Science Dept., Stanford University, July 21, 1966.

**Parameters**



OTHERauxiliary(3)

LAPACK

OTHERauxiliary(3)

*N**N* is INTEGERThe order of the tridiagonal matrix *T*.  $N \geq 0$ .*IW**IW* is INTEGER

The index of the eigenvalues to be returned.

*GL**GL* is REAL*GU**GU* is REAL

An upper and a lower bound on the eigenvalue.

*D**D* is REAL array, dimension (*N*)The *n* diagonal elements of the tridiagonal matrix *T*.*E2**E2* is REAL array, dimension (*N*-1)The (*n*-1) squared off-diagonal elements of the tridiagonal matrix *T*.*PIVMIN**PIVMIN* is REALThe minimum pivot allowed in the Sturm sequence for *T*.*RELTOL**RELTOL* is REALThe minimum relative width of an interval. When an interval is narrower than *RELTOL* times the larger (in magnitude) endpoint, then it is considered to be sufficiently small, i.e., converged. Note: this should always be at least radix\*machine epsilon.*W**W* is REAL*WERR**WERR* is REALThe error bound on the corresponding eigenvalue approximation in *W*.*INFO**INFO* is INTEGER

= 0: Eigenvalue converged

= -1: Eigenvalue did NOT converge

**Internal Parameters:***FUDGE* REAL, default = 2

A "fudge factor" to widen the Gershgorin intervals.

**Author**

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

**Date**

June 2017

**subroutine slarr (integer N, real, dimension( \* ) D, real, dimension( \* ) E, integer INFO)**

**SLARRR** performs tests to decide whether the symmetric tridiagonal matrix T warrants expensive computations which guarantee high relative accuracy in the eigenvalues.

**Purpose:**

Perform tests to decide whether the symmetric tridiagonal matrix T warrants expensive computations which guarantee high relative accuracy in the eigenvalues.

**Parameters**

*N*

N is INTEGER  
The order of the matrix.  $N > 0$ .

*D*

D is REAL array, dimension (N)  
The N diagonal elements of the tridiagonal matrix T.

*E*

E is REAL array, dimension (N)  
On entry, the first (N-1) entries contain the subdiagonal elements of the tridiagonal matrix T; E(N) is set to ZERO.

*INFO*

INFO is INTEGER  
INFO = 0(default) : the matrix warrants computations preserving relative accuracy.  
INFO = 1 : the matrix warrants computations guaranteeing only absolute accuracy.

**Author**

Univ. of Tennessee  
Univ. of California Berkeley  
Univ. of Colorado Denver  
NAG Ltd.

**Date**

June 2017

**Contributors:**

Beresford Parlett, University of California, Berkeley, USA  
Jim Demmel, University of California, Berkeley, USA  
Inderjit Dhillon, University of Texas, Austin, USA  
Osni Marques, LBNL/NERSC, USA  
Christof Voemel, University of California, Berkeley, USA

**subroutine slartg (real F, real G, real CS, real SN, real R)**

**SLARTG** generates a plane rotation with real cosine and real sine.

**Purpose:**

SLARTG generate a plane rotation so that

$$\begin{bmatrix} CS & SN \\ -SN & CS \end{bmatrix} \cdot \begin{bmatrix} F \\ G \end{bmatrix} = \begin{bmatrix} R \\ 0 \end{bmatrix} \quad \text{where } CS^2 + SN^2 = 1.$$

This is a slower, more accurate version of the BLAS1 routine SROTG, with the following other differences:

F and G are unchanged on return.



If  $G=0$ , then  $CS=1$  and  $SN=0$ .

If  $F=0$  and ( $G \neq 0$ ), then  $CS=0$  and  $SN=1$  without doing any floating point operations (saves work in SBDSQR when there are zeros on the diagonal).

If  $F$  exceeds  $G$  in magnitude,  $CS$  will be positive.

### Parameters

*F*

*F* is REAL

The first component of vector to be rotated.

*G*

*G* is REAL

The second component of vector to be rotated.

*CS*

*CS* is REAL

The cosine of the rotation.

*SN*

*SN* is REAL

The sine of the rotation.

*R*

*R* is REAL

The nonzero component of the rotated vector.

This version has a few statements commented out for thread safety (machine parameters are computed on each entry). 10 feb 03, SJH.

### Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

### Date

December 2016

**subroutine slartgp (real F, real G, real CS, real SN, real R)**

**SLARTGP** generates a plane rotation so that the diagonal is nonnegative.

### Purpose:

SLARTGP generates a plane rotation so that

$$\begin{bmatrix} CS & SN \\ -SN & CS \end{bmatrix} \cdot \begin{bmatrix} F \\ G \end{bmatrix} = \begin{bmatrix} R \\ 0 \end{bmatrix} \quad \text{where } CS^2 + SN^2 = 1.$$

This is a slower, more accurate version of the Level 1 BLAS routine SROTG, with the following other differences:

*F* and *G* are unchanged on return.

If  $G=0$ , then  $CS=(+/-)1$  and  $SN=0$ .

If  $F=0$  and ( $G \neq 0$ ), then  $CS=0$  and  $SN=(+/-)1$ .

The sign is chosen so that  $R \geq 0$ .

### Parameters

*F*

*F* is REAL



The first component of vector to be rotated.

*G*

*G* is REAL

The second component of vector to be rotated.

*CS*

*CS* is REAL

The cosine of the rotation.

*SN*

*SN* is REAL

The sine of the rotation.

*R*

*R* is REAL

The nonzero component of the rotated vector.

This version has a few statements commented out for thread safety  
(machine parameters are computed on each entry). 10 feb 03, SJH.

#### Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

#### Date

December 2016

**subroutine slaruv (integer, dimension( 4 ) ISEED, integer N, real, dimension( n ) X)**

**SLARUV** returns a vector of n random real numbers from a uniform distribution.

#### Purpose:

SLARUV returns a vector of n random real numbers from a uniform (0,1)  
distribution (n <= 128).

This is an auxiliary routine called by SLARNV and CLARNV.

#### Parameters

*ISEED*

*ISEED* is INTEGER array, dimension (4)

On entry, the seed of the random number generator; the array  
elements must be between 0 and 4095, and *ISEED*(4) must be  
odd.

On exit, the seed is updated.

*N*

*N* is INTEGER

The number of random numbers to be generated. *N* <= 128.

*X*

*X* is REAL array, dimension (*N*)

The generated random numbers.

#### Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver



NAG Ltd.

# Date

December 2016

# Further Details:

This routine uses a multiplicative congruential method with modulus  $2^{**}48$  and multiplier 33952834046453 (see G.S.Fishman, 'Multiplicative congruential random number generators with modulus  $2^{**}b$ : an exhaustive analysis for  $b = 32$  and a partial analysis for  $b = 48$ ', Math. Comp. 189, pp 331-344, 1990).

48-bit integers are stored in 4 integer array elements with 12 bits per element. Hence the routine is portable across machines with integers of 32 bits or more.

**subroutine slas2 (real F, real G, real H, real SSMIN, real SSMAX)**

**SLAS2** computes singular values of a 2-by-2 triangular matrix.

# Purpose:

**SLAS2** computes the singular values of the 2-by-2 matrix

$$\begin{bmatrix} F & G \\ 0 & H \end{bmatrix}$$

On return, **SSMIN** is the smaller singular value and **SSMAX** is the larger singular value.

# Parameters

*F*

*F* is REAL

The (1,1) element of the 2-by-2 matrix.

*G*

*G* is REAL

The (1,2) element of the 2-by-2 matrix.

*H*

*H* is REAL

The (2,2) element of the 2-by-2 matrix.

*SSMIN*

*SSMIN* is REAL

The smaller singular value.

*SSMAX*

*SSMAX* is REAL

The larger singular value.

# Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

# Date

December 2016

# Further Details:

Barring over/underflow, all output quantities are correct to within



a few units in the last place (ulps), even in the absence of a guard digit in addition/subtraction.

In IEEE arithmetic, the code works correctly if one matrix element is infinite.

Overflow will not occur unless the largest singular value itself overflows, or is within a few ulps of overflow. (On machines with partial overflow, like the Cray, overflow may occur if the largest singular value is within a factor of 2 of overflow.)

Underflow is harmless if underflow is gradual. Otherwise, results may correspond to a matrix modified by perturbations of size near the underflow threshold.

**subroutine slascl (character TYPE, integer KL, integer KU, real CFROM, real CTO, integer M, integer N, real, dimension( lda, \* ) A, integer LDA, integer INFO)**  
**SLASCL** multiplies a general rectangular matrix by a real scalar defined as cto/cfrom.

**Purpose:**

SLASCL multiplies the M by N real matrix A by the real scalar CTO/CFROM. This is done without over/underflow as long as the final result CTO\*A(I,J)/CFROM does not over/underflow. TYPE specifies that A may be full, upper triangular, lower triangular, upper Hessenberg, or banded.

**Parameters**

*TYPE*

TYPE is CHARACTER\*1

TYPE indices the storage type of the input matrix.

= 'G': A is a full matrix.

= 'L': A is a lower triangular matrix.

= 'U': A is an upper triangular matrix.

= 'H': A is an upper Hessenberg matrix.

= 'B': A is a symmetric band matrix with lower bandwidth KL and upper bandwidth KU and with the only the lower half stored.

= 'Q': A is a symmetric band matrix with lower bandwidth KL and upper bandwidth KU and with the only the upper half stored.

= 'Z': A is a band matrix with lower bandwidth KL and upper bandwidth KU. See SGBTRF for storage details.

*KL*

KL is INTEGER

The lower bandwidth of A. Referenced only if TYPE = 'B', 'Q' or 'Z'.

*KU*

KU is INTEGER

The upper bandwidth of A. Referenced only if TYPE = 'B', 'Q' or 'Z'.

*CFROM*

CFROM is REAL

*CTO*

CTO is REAL



The matrix A is multiplied by CTO/CFROM. A(I,J) is computed without over/underflow if the final result CTO\*A(I,J)/CFROM can be represented without over/underflow. CFROM must be nonzero.

*M*

M is INTEGER

The number of rows of the matrix A.  $M \geq 0$ .

*N*

N is INTEGER

The number of columns of the matrix A.  $N \geq 0$ .

*A*

A is REAL array, dimension (LDA,N)

The matrix to be multiplied by CTO/CFROM. See TYPE for the storage type.

*LDA*

LDA is INTEGER

The leading dimension of the array A.

If TYPE = 'G', 'L', 'U', 'H',  $LDA \geq \max(1,M)$ ;

TYPE = 'B',  $LDA \geq KL+1$ ;

TYPE = 'Q',  $LDA \geq KU+1$ ;

TYPE = 'Z',  $LDA \geq 2*KL+KU+1$ .

*INFO*

INFO is INTEGER

0 - successful exit

<0 - if INFO = -i, the i-th argument had an illegal value.

#### Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

#### Date

June 2016

**subroutine slasd0 (integer N, integer SQRE, real, dimension( \*) D, real, dimension( \*) E, real, dimension( ldu, \*) U, integer LDU, real, dimension( ldvt, \*) VT, integer LDVT, integer SMLSIZ, integer, dimension( \*) IWORK, real, dimension( \*) WORK, integer INFO)**

**SLASD0** computes the singular values of a real upper bidiagonal n-by-m matrix B with diagonal d and off-diagonal e. Used by sbdsdc.

#### Purpose:

Using a divide and conquer approach, SLASD0 computes the singular value decomposition (SVD) of a real upper bidiagonal N-by-M matrix B with diagonal D and offdiagonal E, where  $M = N + SQRE$ . The algorithm computes orthogonal matrices U and VT such that  $B = U * S * VT$ . The singular values S are overwritten on D.

A related subroutine, SLASDA, computes only the singular values, and optionally, the singular vectors in compact form.

#### Parameters

*N*

N is INTEGER

On entry, the row dimension of the upper bidiagonal matrix.



This is also the dimension of the main diagonal array D.

### *SQRE*

SQRE is INTEGER

Specifies the column dimension of the bidiagonal matrix.

= 0: The bidiagonal matrix has column dimension  $M = N$ ;

= 1: The bidiagonal matrix has column dimension  $M = N+1$ ;

### *D*

D is REAL array, dimension (N)

On entry D contains the main diagonal of the bidiagonal matrix.

On exit D, if INFO = 0, contains its singular values.

### *E*

E is REAL array, dimension (M-1)

Contains the subdiagonal entries of the bidiagonal matrix.

On exit, E has been destroyed.

### *U*

U is REAL array, dimension (LDU, N)

On exit, U contains the left singular vectors.

### *LDU*

LDU is INTEGER

On entry, leading dimension of U.

### *VT*

VT is REAL array, dimension (LDVT, M)

On exit, VT\*\*T contains the right singular vectors.

### *LDVT*

LDVT is INTEGER

On entry, leading dimension of VT.

### *SMLSIZ*

SMLSIZ is INTEGER

On entry, maximum size of the subproblems at the bottom of the computation tree.

### *IWORK*

IWORK is INTEGER array, dimension (8\*N)

### *WORK*

WORK is REAL array, dimension (3\*M\*\*2+2\*M)

### *INFO*

INFO is INTEGER

= 0: successful exit.

< 0: if INFO = -i, the i-th argument had an illegal value.

> 0: if INFO = 1, a singular value did not converge

### **Author**

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

### **Date**

June 2017





**Contributors:**

Ming Gu and Huan Ren, Computer Science Division, University of California at Berkeley, USA

**subroutine slasd1 (integer NL, integer NR, integer SQRE, real, dimension( \* ) D, real ALPHA, real BETA, real, dimension( ldu, \* ) U, integer LDU, real, dimension( ldvt, \* ) VT, integer LDVT, integer, dimension( \* ) IDXQ, integer, dimension( \* ) IWORK, real, dimension( \* ) WORK, integer INFO)**

**SLASD1** computes the SVD of an upper bidiagonal matrix B of the specified size. Used by sbdsdc.

**Purpose:**

SLASD1 computes the SVD of an upper bidiagonal N-by-M matrix B, where  $N = NL + NR + 1$  and  $M = N + SQRE$ . SLASD1 is called from SLASD0.

A related subroutine SLASD7 handles the case in which the singular values (and the singular vectors in factored form) are desired.

SLASD1 computes the SVD as follows:

$$B = U(\text{in}) * \begin{pmatrix} D1(\text{in}) & 0 & 0 & 0 \\ Z1^{**T} & a & Z2^{**T} & b \\ 0 & 0 & D2(\text{in}) & 0 \end{pmatrix} * VT(\text{in})$$

$$= U(\text{out}) * \begin{pmatrix} D(\text{out}) & 0 \end{pmatrix} * VT(\text{out})$$

where  $Z^{**T} = (Z1^{**T} \ a \ Z2^{**T} \ b) = u^{**T} \ VT^{**T}$ , and u is a vector of dimension M with ALPHA and BETA in the NL+1 and NL+2 th entries and zeros elsewhere; and the entry b is empty if  $SQRE = 0$ .

The left singular vectors of the original matrix are stored in U, and the transpose of the right singular vectors are stored in VT, and the singular values are in D. The algorithm consists of three stages:

The first stage consists of deflating the size of the problem when there are multiple singular values or when there are zeros in the Z vector. For each such occurrence the dimension of the secular equation problem is reduced by one. This stage is performed by the routine SLASD2.

The second stage consists of calculating the updated singular values. This is done by finding the square roots of the roots of the secular equation via the routine SLASD4 (as called by SLASD3). This routine also calculates the singular vectors of the current problem.

The final stage consists of computing the updated singular vectors directly using the updated singular values. The singular vectors for the current problem are multiplied with the singular vectors from the overall problem.

**Parameters**

*NL*

NL is INTEGER

The row dimension of the upper block.  $NL \geq 1$ .

*NR*

NR is INTEGER

The row dimension of the lower block.  $NR \geq 1$ .

*SQRE*



SQRE is INTEGER

= 0: the lower block is an NR-by-NR square matrix.

= 1: the lower block is an NR-by-(NR+1) rectangular matrix.

The bidiagonal matrix has row dimension  $N = NL + NR + 1$ ,  
and column dimension  $M = N + SQRE$ .

*D*

D is REAL array, dimension (NL+NR+1).

$N = NL + NR + 1$

On entry D(1:NL,1:NL) contains the singular values of the upper block; and D(NL+2:N) contains the singular values of the lower block. On exit D(1:N) contains the singular values of the modified matrix.

*ALPHA*

ALPHA is REAL

Contains the diagonal element associated with the added row.

*BETA*

BETA is REAL

Contains the off-diagonal element associated with the added row.

*U*

U is REAL array, dimension (LDU,N)

On entry U(1:NL, 1:NL) contains the left singular vectors of the upper block; U(NL+2:N, NL+2:N) contains the left singular vectors of the lower block. On exit U contains the left singular vectors of the bidiagonal matrix.

*LDU*

LDU is INTEGER

The leading dimension of the array U.  $LDU \geq \max(1, N)$ .

*VT*

VT is REAL array, dimension (LDVT,M)

where  $M = N + SQRE$ .

On entry VT(1:NL+1, 1:NL+1)\*\*T contains the right singular vectors of the upper block; VT(NL+2:M, NL+2:M)\*\*T contains the right singular vectors of the lower block. On exit VT\*\*T contains the right singular vectors of the bidiagonal matrix.

*LDVT*

LDVT is INTEGER

The leading dimension of the array VT.  $LDVT \geq \max(1, M)$ .

*IDXQ*

IDXQ is INTEGER array, dimension (N)

This contains the permutation which will reintegrate the subproblem just solved back into sorted order, i.e.

D( IDXQ( I = 1, N ) ) will be in ascending order.

*IWORK*

IWORK is INTEGER array, dimension (4\*N)

*WORK*

WORK is REAL array, dimension (3\*M\*\*2+2\*M)

*INFO*



INFO is INTEGER

= 0: successful exit.

< 0: if INFO = -i, the i-th argument had an illegal value.

> 0: if INFO = 1, a singular value did not converge

#### Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

#### Date

June 2016

#### Contributors:

Ming Gu and Huan Ren, Computer Science Division, University of California at Berkeley, USA

**subroutine slasd2 (integer NL, integer NR, integer SQRE, integer K, real, dimension( \*) D, real, dimension( \*) Z, real ALPHA, real BETA, real, dimension( ldu, \*) U, integer LDU, real, dimension( ldvt, \*) VT, integer LDVT, real, dimension( \*) DSIGMA, real, dimension( ldu2, \*) U2, integer LDU2, real, dimension( ldvt2, \*) VT2, integer LDVT2, integer, dimension( \*) IDXP, integer, dimension( \*) IDX, integer, dimension( \*) IDXC, integer, dimension( \*) IDXQ, integer, dimension( \*) COLTYP, integer INFO)**

SLASD2 merges the two sets of singular values together into a single sorted set. Used by sbdsdc.

#### Purpose:

SLASD2 merges the two sets of singular values together into a single sorted set. Then it tries to deflate the size of the problem.

There are two ways in which deflation can occur: when two or more singular values are close together or if there is a tiny entry in the Z vector. For each such occurrence the order of the related secular equation problem is reduced by one.

SLASD2 is called from SLASD1.

#### Parameters

*NL*

NL is INTEGER

The row dimension of the upper block. NL >= 1.

*NR*

NR is INTEGER

The row dimension of the lower block. NR >= 1.

*SQRE*

SQRE is INTEGER

= 0: the lower block is an NR-by-NR square matrix.

= 1: the lower block is an NR-by-(NR+1) rectangular matrix.

The bidiagonal matrix has N = NL + NR + 1 rows and

M = N + SQRE >= N columns.

*K*

K is INTEGER

Contains the dimension of the non-deflated matrix,

This is the order of the related secular equation. 1 <= K <= N.

*D*

D is REAL array, dimension (N)

On entry D contains the singular values of the two submatrices



to be combined. On exit D contains the trailing (N-K) updated singular values (those which were deflated) sorted into increasing order.

*Z*

*Z* is REAL array, dimension (N)

On exit *Z* contains the updating row vector in the secular equation.

*ALPHA*

*ALPHA* is REAL

Contains the diagonal element associated with the added row.

*BETA*

*BETA* is REAL

Contains the off-diagonal element associated with the added row.

*U*

*U* is REAL array, dimension (LDU,N)

On entry *U* contains the left singular vectors of two submatrices in the two square blocks with corners at (1,1), (NL, NL), and (NL+2, NL+2), (N,N).

On exit *U* contains the trailing (N-K) updated left singular vectors (those which were deflated) in its last N-K columns.

*LDU*

*LDU* is INTEGER

The leading dimension of the array *U*.  $LDU \geq N$ .

*VT*

*VT* is REAL array, dimension (LDVT,M)

On entry *VT\*\*T* contains the right singular vectors of two submatrices in the two square blocks with corners at (1,1), (NL+1, NL+1), and (NL+2, NL+2), (M,M).

On exit *VT\*\*T* contains the trailing (N-K) updated right singular vectors (those which were deflated) in its last N-K columns.

In case  $SQRE = 1$ , the last row of *VT* spans the right null space.

*LDVT*

*LDVT* is INTEGER

The leading dimension of the array *VT*.  $LDVT \geq M$ .

*DSIGMA*

*DSIGMA* is REAL array, dimension (N)

Contains a copy of the diagonal elements (K-1 singular values and one zero) in the secular equation.

*U2*

*U2* is REAL array, dimension (LDU2,N)

Contains a copy of the first K-1 left singular vectors which will be used by SLASD3 in a matrix multiply (SGEMM) to solve for the new left singular vectors. *U2* is arranged into four blocks. The first block contains a column with 1 at NL+1 and zero everywhere else; the second block contains non-zero entries only at and above NL; the third contains non-zero entries only below NL+1; and the fourth is dense.

*LDU2*

*LDU2* is INTEGER



The leading dimension of the array U2.  $LDU2 \geq N$ .

#### *VT2*

VT2 is REAL array, dimension (LDVT2,N)

VT2\*\*T contains a copy of the first K right singular vectors which will be used by SLASD3 in a matrix multiply (SGEMM) to solve for the new right singular vectors. VT2 is arranged into three blocks. The first block contains a row that corresponds to the special 0 diagonal element in SIGMA; the second block contains non-zeros only at and before NL +1; the third block contains non-zeros only at and after NL +2.

#### *LDVT2*

LDVT2 is INTEGER

The leading dimension of the array VT2.  $LDVT2 \geq M$ .

#### *IDXP*

IDXP is INTEGER array, dimension (N)

This will contain the permutation used to place deflated values of D at the end of the array. On output IDXP(2:K) points to the nondeflated D-values and IDXP(K+1:N) points to the deflated singular values.

#### *IDX*

IDX is INTEGER array, dimension (N)

This will contain the permutation used to sort the contents of D into ascending order.

#### *IDXC*

IDXC is INTEGER array, dimension (N)

This will contain the permutation used to arrange the columns of the deflated U matrix into three groups: the first group contains non-zero entries only at and above NL, the second contains non-zero entries only below NL+2, and the third is dense.

#### *IDXQ*

IDXQ is INTEGER array, dimension (N)

This contains the permutation which separately sorts the two sub-problems in D into ascending order. Note that entries in the first half of this permutation must first be moved one position backward; and entries in the second half must first have NL+1 added to their values.

#### *COLTYP*

COLTYP is INTEGER array, dimension (N)

As workspace, this will contain a label which will indicate which of the following types a column in the U2 matrix or a row in the VT2 matrix is:

- 1 : non-zero in the upper half only
- 2 : non-zero in the lower half only
- 3 : dense
- 4 : deflated

On exit, it is an array of dimension 4, with COLTYP(I) being the dimension of the I-th type columns.

#### *INFO*

INFO is INTEGER

= 0: successful exit.



< 0: if INFO = -i, the i-th argument had an illegal value.

#### Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

#### Date

December 2016

#### Contributors:

Ming Gu and Huan Ren, Computer Science Division, University of California at Berkeley, USA

**subroutine slasd3 (integer NL, integer NR, integer SQRE, integer K, real, dimension( \*) D, real, dimension( ldq, \*) Q, integer LDQ, real, dimension( \*) DSIGMA, real, dimension( ldu, \*) U, integer LDU, real, dimension( ldu2, \*) U2, integer LDU2, real, dimension( ldvt, \*) VT, integer LDVT, real, dimension( ldvt2, \*) VT2, integer LDVT2, integer, dimension( \*) IDXC, integer, dimension( \*) CTOT, real, dimension( \*) Z, integer INFO)**

SLASD3 finds all square roots of the roots of the secular equation, as defined by the values in D and Z, and then updates the singular vectors by matrix multiplication. Used by sbdsdc.

#### Purpose:

SLASD3 finds all the square roots of the roots of the secular equation, as defined by the values in D and Z. It makes the appropriate calls to SLASD4 and then updates the singular vectors by matrix multiplication.

This code makes very mild assumptions about floating point arithmetic. It will work on machines with a guard digit in add/subtract, or on those binary machines without guard digits which subtract like the Cray XMP, Cray YMP, Cray C 90, or Cray 2. It could conceivably fail on hexadecimal or decimal machines without guard digits, but we know of none.

SLASD3 is called from SLASD1.

#### Parameters

*NL*

NL is INTEGER

The row dimension of the upper block. NL >= 1.

*NR*

NR is INTEGER

The row dimension of the lower block. NR >= 1.

*SQRE*

SQRE is INTEGER

= 0: the lower block is an NR-by-NR square matrix.

= 1: the lower block is an NR-by-(NR+1) rectangular matrix.

The bidiagonal matrix has N = NL + NR + 1 rows and

M = N + SQRE >= N columns.

*K*

K is INTEGER

The size of the secular equation, 1 <= K < N.

*D*

D is REAL array, dimension(K)



On exit the square roots of the roots of the secular equation, in ascending order.

*Q*

*Q* is REAL array, dimension (LDQ,K)

*LDQ*

LDQ is INTEGER

The leading dimension of the array *Q*. LDQ  $\geq$  K.

*DSIGMA*

DSIGMA is REAL array, dimension(K)

The first K elements of this array contain the old roots of the deflated updating problem. These are the poles of the secular equation.

*U*

*U* is REAL array, dimension (LDU, N)

The last N - K columns of this matrix contain the deflated left singular vectors.

*LDU*

LDU is INTEGER

The leading dimension of the array *U*. LDU  $\geq$  N.

*U2*

*U2* is REAL array, dimension (LDU2, N)

The first K columns of this matrix contain the non-deflated left singular vectors for the split problem.

*LDU2*

LDU2 is INTEGER

The leading dimension of the array *U2*. LDU2  $\geq$  N.

*VT*

*VT* is REAL array, dimension (LDVT, M)

The last M - K columns of *VT*\*\**T* contain the deflated right singular vectors.

*LDVT*

LDVT is INTEGER

The leading dimension of the array *VT*. LDVT  $\geq$  N.

*VT2*

*VT2* is REAL array, dimension (LDVT2, N)

The first K columns of *VT2*\*\**T* contain the non-deflated right singular vectors for the split problem.

*LDVT2*

LDVT2 is INTEGER

The leading dimension of the array *VT2*. LDVT2  $\geq$  N.

*IDXC*

IDXC is INTEGER array, dimension (N)

The permutation used to arrange the columns of *U* (and rows of *VT*) into three groups: the first group contains non-zero entries only at and above (or before) NL + 1; the second contains non-zero entries only at and below (or after) NL+2; and the third is dense. The first column of *U* and the row of *VT* are treated separately, however.



The rows of the singular vectors found by SLASD4 must be likewise permuted before the matrix multiplies can take place.

#### *CTOT*

CTOT is INTEGER array, dimension (4)  
A count of the total number of the various types of columns in U (or rows in VT), as described in IDXC. The fourth column type is any column which has been deflated.

#### *Z*

Z is REAL array, dimension (K)  
The first K elements of this array contain the components of the deflation-adjusted updating row vector.

#### *INFO*

INFO is INTEGER  
= 0: successful exit.  
< 0: if INFO = -i, the i-th argument had an illegal value.  
> 0: if INFO = 1, a singular value did not converge

#### **Author**

Univ. of Tennessee  
Univ. of California Berkeley  
Univ. of Colorado Denver  
NAG Ltd.

#### **Date**

June 2017

#### **Contributors:**

Ming Gu and Huan Ren, Computer Science Division, University of California at Berkeley, USA

**subroutine slasd4 (integer N, integer I, real, dimension( \*) D, real, dimension( \*) Z, real, dimension( \*) DELTA, real RHO, real SIGMA, real, dimension( \*) WORK, integer INFO)**

**SLASD4** computes the square root of the i-th updated eigenvalue of a positive symmetric rank-one modification to a positive diagonal matrix. Used by sbdsdc.

#### **Purpose:**

This subroutine computes the square root of the I-th updated eigenvalue of a positive symmetric rank-one modification to a positive diagonal matrix whose entries are given as the squares of the corresponding entries in the array d, and that

$$0 \leq D(i) < D(j) \text{ for } i < j$$

and that  $RHO > 0$ . This is arranged by the calling routine, and is no loss in generality. The rank-one modified system is thus

$$\text{diag}(D) * \text{diag}(D) + RHO * Z * Z_{\text{transpose}}.$$

where we assume the Euclidean norm of Z is 1.

The method consists of approximating the rational functions in the secular equation by simpler interpolating rational functions.

#### **Parameters**

*N*

N is INTEGER  
The length of all arrays.





*I**I* is INTEGERThe index of the eigenvalue to be computed.  $1 \leq I \leq N$ .*D**D* is REAL array, dimension ( *N* )The original eigenvalues. It is assumed that they are in order,  $0 \leq D(I) < D(J)$  for  $I < J$ .*Z**Z* is REAL array, dimension ( *N* )

The components of the updating vector.

*DELTA**DELTA* is REAL array, dimension ( *N* )If *N* .ne. 1, *DELTA* contains (*D*(*j*) - *sigma*<sub>*I*</sub>) in its *j*-th component. If *N* = 1, then *DELTA*(1) = 1. The vector *DELTA* contains the information necessary to construct the (singular) eigenvectors.*RHO**RHO* is REAL

The scalar in the symmetric updating formula.

*SIGMA**SIGMA* is REALThe computed *sigma*<sub>*I*</sub>, the *I*-th updated eigenvalue.*WORK**WORK* is REAL array, dimension ( *N* )If *N* .ne. 1, *WORK* contains (*D*(*j*) + *sigma*<sub>*I*</sub>) in its *j*-th component. If *N* = 1, then *WORK*( 1 ) = 1.*INFO**INFO* is INTEGER

= 0: successful exit

> 0: if *INFO* = 1, the updating process failed.**Internal Parameters:**

Logical variable *ORGATI* (origin-at-i?) is used for distinguishing whether *D*(*i*) or *D*(*i*+1) is treated as the origin.

*ORGATI* = .true. origin at *i**ORGATI* = .false. origin at *i*+1

Logical variable *SWTCH3* (switch-for-3-poles?) is for noting if we are working with *THREE* poles!

*MAXIT* is the maximum number of iterations allowed for each eigenvalue.

**Author**

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

**Date**

December 2016



**Contributors:**

Ren-Cang Li, Computer Science Division, University of California at Berkeley, USA

**subroutine slasd5 (integer I, real, dimension( 2 ) D, real, dimension( 2 ) Z, real, dimension( 2 ) DELTA, real RHO, real DSIGMA, real, dimension( 2 ) WORK)**

**SLASD5** computes the square root of the i-th eigenvalue of a positive symmetric rank-one modification of a 2-by-2 diagonal matrix. Used by sbdsdc.

**Purpose:**

This subroutine computes the square root of the I-th eigenvalue of a positive symmetric rank-one modification of a 2-by-2 diagonal matrix

$$\text{diag}(D) * \text{diag}(D) + RHO * Z * \text{transpose}(Z).$$

The diagonal entries in the array D are assumed to satisfy

$$0 \leq D(i) < D(j) \text{ for } i < j.$$

We also assume  $RHO > 0$  and that the Euclidean norm of the vector Z is one.

**Parameters**

*I*

I is INTEGER

The index of the eigenvalue to be computed. I = 1 or I = 2.

*D*

D is REAL array, dimension (2)

The original eigenvalues. We assume  $0 \leq D(1) < D(2)$ .

*Z*

Z is REAL array, dimension (2)

The components of the updating vector.

*DELTA*

DELTA is REAL array, dimension (2)

Contains  $(D(j) - \text{sigma}_I)$  in its j-th component.

The vector DELTA contains the information necessary to construct the eigenvectors.

*RHO*

RHO is REAL

The scalar in the symmetric updating formula.

*DSIGMA*

DSIGMA is REAL

The computed  $\text{sigma}_I$ , the I-th updated eigenvalue.

*WORK*

WORK is REAL array, dimension (2)

WORK contains  $(D(j) + \text{sigma}_I)$  in its j-th component.

**Author**

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

**Date**

December 2016

**Contributors:**

Ren-Cang Li, Computer Science Division, University of California at Berkeley, USA

**subroutine slasd6 (integer ICOMPQ, integer NL, integer NR, integer SQRE, real, dimension( \* ) D, real, dimension( \* ) VF, real, dimension( \* ) VL, real ALPHA, real BETA, integer, dimension( \* ) IDXQ, integer, dimension( \* ) PERM, integer GIVPTR, integer, dimension( ldgcol, \* ) GIVCOL, integer LDGCOL, real, dimension( ldgnum, \* ) GIVNUM, integer LDGNUM, real, dimension( ldgnum, \* ) POLES, real, dimension( \* ) DIFL, real, dimension( \* ) DIFR, real, dimension( \* ) Z, integer K, real C, real S, real, dimension( \* ) WORK, integer, dimension( \* ) IWORK, integer INFO)**

**SLASD6** computes the SVD of an updated upper bidiagonal matrix obtained by merging two smaller ones by appending a row. Used by sbdsdc.

**Purpose:**

SLASD6 computes the SVD of an updated upper bidiagonal matrix B obtained by merging two smaller ones by appending a row. This routine is used only for the problem which requires all singular values and optionally singular vector matrices in factored form. B is an N-by-M matrix with  $N = NL + NR + 1$  and  $M = N + SQRE$ . A related subroutine, SLASD1, handles the case in which all singular values and singular vectors of the bidiagonal matrix are desired.

SLASD6 computes the SVD as follows:

$$B = U(\text{in}) * \begin{pmatrix} D1(\text{in}) & 0 & 0 & 0 \\ Z1^{**T} & a & Z2^{**T} & b \\ 0 & 0 & D2(\text{in}) & 0 \end{pmatrix} * VT(\text{in})$$

$$= U(\text{out}) * (D(\text{out}) \ 0) * VT(\text{out})$$

where  $Z^{**T} = (Z1^{**T} \ a \ Z2^{**T} \ b) = u^{**T} \ VT^{**T}$ , and u is a vector of dimension M with ALPHA and BETA in the NL+1 and NL+2 th entries and zeros elsewhere; and the entry b is empty if SQRE = 0.

The singular values of B can be computed using D1, D2, the first components of all the right singular vectors of the lower block, and the last components of all the right singular vectors of the upper block. These components are stored and updated in VF and VL, respectively, in SLASD6. Hence U and VT are not explicitly referenced.

The singular values are stored in D. The algorithm consists of two stages:

The first stage consists of deflating the size of the problem when there are multiple singular values or if there is a zero in the Z vector. For each such occurrence the dimension of the secular equation problem is reduced by one. This stage is performed by the routine SLASD7.

The second stage consists of calculating the updated singular values. This is done by finding the roots of the secular equation via the routine SLASD4 (as called by SLASD8). This routine also updates VF and VL and computes the distances between the updated singular values and the old singular values.



SLASD6 is called from SLASDA.

### Parameters

#### *ICOMPQ*

ICOMPQ is INTEGER

Specifies whether singular vectors are to be computed in factored form:

= 0: Compute singular values only.

= 1: Compute singular vectors in factored form as well.

#### *NL*

NL is INTEGER

The row dimension of the upper block.  $NL \geq 1$ .

#### *NR*

NR is INTEGER

The row dimension of the lower block.  $NR \geq 1$ .

#### *SQRE*

SQRE is INTEGER

= 0: the lower block is an NR-by-NR square matrix.

= 1: the lower block is an NR-by-(NR+1) rectangular matrix.

The bidiagonal matrix has row dimension  $N = NL + NR + 1$ , and column dimension  $M = N + SQRE$ .

#### *D*

D is REAL array, dimension (NL+NR+1).

On entry D(1:NL,1:NL) contains the singular values of the upper block, and D(NL+2:N) contains the singular values of the lower block. On exit D(1:N) contains the singular values of the modified matrix.

#### *VF*

VF is REAL array, dimension (M)

On entry, VF(1:NL+1) contains the first components of all right singular vectors of the upper block; and VF(NL+2:M) contains the first components of all right singular vectors of the lower block. On exit, VF contains the first components of all right singular vectors of the bidiagonal matrix.

#### *VL*

VL is REAL array, dimension (M)

On entry, VL(1:NL+1) contains the last components of all right singular vectors of the upper block; and VL(NL+2:M) contains the last components of all right singular vectors of the lower block. On exit, VL contains the last components of all right singular vectors of the bidiagonal matrix.

#### *ALPHA*

ALPHA is REAL

Contains the diagonal element associated with the added row.

#### *BETA*

BETA is REAL

Contains the off-diagonal element associated with the added row.

#### *IDXQ*

IDXQ is INTEGER array, dimension (N)

This contains the permutation which will reintegrate the



subproblem just solved back into sorted order, i.e.  
 $D(\text{IDXQ}(I = 1, N))$  will be in ascending order.

#### *PERM*

PERM is INTEGER array, dimension ( N )  
 The permutations (from deflation and sorting) to be applied  
 to each block. Not referenced if ICOMPQ = 0.

#### *GIVPTR*

GIVPTR is INTEGER  
 The number of Givens rotations which took place in this  
 subproblem. Not referenced if ICOMPQ = 0.

#### *GIVCOL*

GIVCOL is INTEGER array, dimension ( LDGCOL, 2 )  
 Each pair of numbers indicates a pair of columns to take place  
 in a Givens rotation. Not referenced if ICOMPQ = 0.

#### *LDGCOL*

LDGCOL is INTEGER  
 leading dimension of GIVCOL, must be at least N.

#### *GIVNUM*

GIVNUM is REAL array, dimension ( LDGNUM, 2 )  
 Each number indicates the C or S value to be used in the  
 corresponding Givens rotation. Not referenced if ICOMPQ = 0.

#### *LDGNUM*

LDGNUM is INTEGER  
 The leading dimension of GIVNUM and POLES, must be at least N.

#### *POLES*

POLES is REAL array, dimension ( LDGNUM, 2 )  
 On exit, POLES(1,\*) is an array containing the new singular  
 values obtained from solving the secular equation, and  
 POLES(2,\*) is an array containing the poles in the secular  
 equation. Not referenced if ICOMPQ = 0.

#### *DIFL*

DIFL is REAL array, dimension ( N )  
 On exit, DIFL(I) is the distance between I-th updated  
 (undeflated) singular value and the I-th (undeflated) old  
 singular value.

#### *DIFR*

DIFR is REAL array,  
 dimension ( LDDIFR, 2 ) if ICOMPQ = 1 and  
 dimension ( K ) if ICOMPQ = 0.  
 On exit, DIFR(I,1) = D(I) - DSIGMA(I+1), DIFR(K,1) is not  
 defined and will not be referenced.

If ICOMPQ = 1, DIFR(1:K,2) is an array containing the  
 normalizing factors for the right singular vector matrix.

See SLASD8 for details on DIFL and DIFR.

#### *Z*

Z is REAL array, dimension ( M )  
 The first elements of this array contain the components  
 of the deflation-adjusted updating row vector.



*K**K* is INTEGER

Contains the dimension of the non-deflated matrix,  
This is the order of the related secular equation.  $1 \leq K \leq N$ .

*C**C* is REAL

*C* contains garbage if *SQRE* = 0 and the *C*-value of a Givens  
rotation related to the right null space if *SQRE* = 1.

*S**S* is REAL

*S* contains garbage if *SQRE* = 0 and the *S*-value of a Givens  
rotation related to the right null space if *SQRE* = 1.

*WORK**WORK* is REAL array, dimension ( 4 \* *M* )*IWORK**IWORK* is INTEGER array, dimension ( 3 \* *N* )*INFO**INFO* is INTEGER

= 0: successful exit.

< 0: if *INFO* = -*i*, the *i*-th argument had an illegal value.> 0: if *INFO* = 1, a singular value did not converge**Author**

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

**Date**

June 2016

**Contributors:**

Ming Gu and Huan Ren, Computer Science Division, University of California at Berkeley, USA

**subroutine slasd7 (integer ICOMPQ, integer NL, integer NR, integer SQRE, integer K, real, dimension( \*) D, real, dimension( \*) Z, real, dimension( \*) ZW, real, dimension( \*) VF, real, dimension( \*) VFW, real, dimension( \*) VL, real, dimension( \*) VLW, real ALPHA, real BETA, real, dimension( \*) DSIGMA, integer, dimension( \*) IDX, integer, dimension( \*) IDXP, integer, dimension( \*) IDXQ, integer, dimension( \*) PERM, integer GIVPTR, integer, dimension( ldgcol, \*) GIVCOL, integer LDGCOL, real, dimension( ldgnum, \*) GIVNUM, integer LDGNUM, real C, real S, integer INFO)**

**SLASD7** merges the two sets of singular values together into a single sorted set. Then it tries to deflate the size of the problem. Used by *sbsdsc*.

**Purpose:**

**SLASD7** merges the two sets of singular values together into a single sorted set. Then it tries to deflate the size of the problem. There are two ways in which deflation can occur: when two or more singular values are close together or if there is a tiny entry in the *Z* vector. For each such occurrence the order of the related secular equation problem is reduced by one.

**SLASD7** is called from **SLASD6**.

**Parameters**

*ICOMPQ*

ICOMPQ is INTEGER

Specifies whether singular vectors are to be computed in compact form, as follows:

= 0: Compute singular values only.

= 1: Compute singular vectors of upper bidiagonal matrix in compact form.

*NL*

NL is INTEGER

The row dimension of the upper block.  $NL \geq 1$ .

*NR*

NR is INTEGER

The row dimension of the lower block.  $NR \geq 1$ .

*SQRE*

SQRE is INTEGER

= 0: the lower block is an NR-by-NR square matrix.

= 1: the lower block is an NR-by-(NR+1) rectangular matrix.

The bidiagonal matrix has

$N = NL + NR + 1$  rows and

$M = N + SQRE \geq N$  columns.

*K*

K is INTEGER

Contains the dimension of the non-deflated matrix, this is the order of the related secular equation.  $1 \leq K \leq N$ .

*D*

D is REAL array, dimension ( N )

On entry D contains the singular values of the two submatrices to be combined. On exit D contains the trailing (N-K) updated singular values (those which were deflated) sorted into increasing order.

*Z*

Z is REAL array, dimension ( M )

On exit Z contains the updating row vector in the secular equation.

*ZW*

ZW is REAL array, dimension ( M )

Workspace for Z.

*VF*

VF is REAL array, dimension ( M )

On entry, VF(1:NL+1) contains the first components of all right singular vectors of the upper block; and VF(NL+2:M) contains the first components of all right singular vectors of the lower block. On exit, VF contains the first components of all right singular vectors of the bidiagonal matrix.

*VFW*

VFW is REAL array, dimension ( M )

Workspace for VF.

*VL*

VL is REAL array, dimension ( M )



On entry, VL(1:NL+1) contains the last components of all right singular vectors of the upper block; and VL(NL+2:M) contains the last components of all right singular vectors of the lower block. On exit, VL contains the last components of all right singular vectors of the bidiagonal matrix.

*VLW*

VLW is REAL array, dimension ( M )  
Workspace for VL.

*ALPHA*

ALPHA is REAL  
Contains the diagonal element associated with the added row.

*BETA*

BETA is REAL  
Contains the off-diagonal element associated with the added row.

*DSIGMA*

DSIGMA is REAL array, dimension ( N )  
Contains a copy of the diagonal elements (K-1 singular values and one zero) in the secular equation.

*IDX*

IDX is INTEGER array, dimension ( N )  
This will contain the permutation used to sort the contents of D into ascending order.

*IDXP*

IDXP is INTEGER array, dimension ( N )  
This will contain the permutation used to place deflated values of D at the end of the array. On output IDXP(2:K) points to the nondeflated D-values and IDXP(K+1:N) points to the deflated singular values.

*IDXQ*

IDXQ is INTEGER array, dimension ( N )  
This contains the permutation which separately sorts the two sub-problems in D into ascending order. Note that entries in the first half of this permutation must first be moved one position backward; and entries in the second half must first have NL+1 added to their values.

*PERM*

PERM is INTEGER array, dimension ( N )  
The permutations (from deflation and sorting) to be applied to each singular block. Not referenced if ICOMPQ = 0.

*GIVPTR*

GIVPTR is INTEGER  
The number of Givens rotations which took place in this subproblem. Not referenced if ICOMPQ = 0.

*GIVCOL*

GIVCOL is INTEGER array, dimension ( LDGCOL, 2 )  
Each pair of numbers indicates a pair of columns to take place in a Givens rotation. Not referenced if ICOMPQ = 0.

*LDGCOL*

LDGCOL is INTEGER





The leading dimension of GIVCOL, must be at least N.

#### *GIVNUM*

GIVNUM is REAL array, dimension ( LDGNUM, 2 )

Each number indicates the C or S value to be used in the corresponding Givens rotation. Not referenced if ICOMPQ = 0.

#### *LDGNUM*

LDGNUM is INTEGER

The leading dimension of GIVNUM, must be at least N.

#### *C*

C is REAL

C contains garbage if SQRE = 0 and the C-value of a Givens rotation related to the right null space if SQRE = 1.

#### *S*

S is REAL

S contains garbage if SQRE = 0 and the S-value of a Givens rotation related to the right null space if SQRE = 1.

#### *INFO*

INFO is INTEGER

= 0: successful exit.

< 0: if INFO = -i, the i-th argument had an illegal value.

#### **Author**

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

#### **Date**

December 2016

#### **Contributors:**

Ming Gu and Huan Ren, Computer Science Division, University of California at Berkeley, USA

**subroutine slasd8 (integer ICOMPQ, integer K, real, dimension( \* ) D, real, dimension( \* ) Z, real, dimension( \* ) VF, real, dimension( \* ) VL, real, dimension( \* ) DIFL, real, dimension( lddifr, \* ) DIFR, integer LDDIFR, real, dimension( \* ) DSIGMA, real, dimension( \* ) WORK, integer INFO)**

**SLASD8** finds the square roots of the roots of the secular equation, and stores, for each element in D, the distance to its two nearest poles. Used by sbdsdc.

#### **Purpose:**

SLASD8 finds the square roots of the roots of the secular equation, as defined by the values in DSIGMA and Z. It makes the appropriate calls to SLASD4, and stores, for each element in D, the distance to its two nearest poles (elements in DSIGMA). It also updates the arrays VF and VL, the first and last components of all the right singular vectors of the original bidiagonal matrix.

SLASD8 is called from SLASD6.

#### **Parameters**

##### *ICOMPQ*

ICOMPQ is INTEGER

Specifies whether singular vectors are to be computed in factored form in the calling routine:



= 0: Compute singular values only.  
 = 1: Compute singular vectors in factored form as well.

*K*

*K* is INTEGER

The number of terms in the rational function to be solved by SLASD4.  $K \geq 1$ .

*D*

*D* is REAL array, dimension ( *K* )

On output, *D* contains the updated singular values.

*Z*

*Z* is REAL array, dimension ( *K* )

On entry, the first *K* elements of this array contain the components of the deflation-adjusted updating row vector.  
 On exit, *Z* is updated.

*VF*

*VF* is REAL array, dimension ( *K* )

On entry, *VF* contains information passed through DBEDE8.  
 On exit, *VF* contains the first *K* components of the first components of all right singular vectors of the bidiagonal matrix.

*VL*

*VL* is REAL array, dimension ( *K* )

On entry, *VL* contains information passed through DBEDE8.  
 On exit, *VL* contains the first *K* components of the last components of all right singular vectors of the bidiagonal matrix.

*DIFL*

*DIFL* is REAL array, dimension ( *K* )

On exit,  $DIFL(I) = D(I) - DSIGMA(I)$ .

*DIFR*

*DIFR* is REAL array,

dimension ( *LDDIFR*, 2 ) if *ICOMPQ* = 1 and

dimension ( *K* ) if *ICOMPQ* = 0.

On exit,  $DIFR(I,1) = D(I) - DSIGMA(I+1)$ , *DIFR*(*K*,1) is not defined and will not be referenced.

If *ICOMPQ* = 1, *DIFR*(1:*K*,2) is an array containing the normalizing factors for the right singular vector matrix.

*LDDIFR*

*LDDIFR* is INTEGER

The leading dimension of *DIFR*, must be at least *K*.

*DSIGMA*

*DSIGMA* is REAL array, dimension ( *K* )

On entry, the first *K* elements of this array contain the old roots of the deflated updating problem. These are the poles of the secular equation.

On exit, the elements of *DSIGMA* may be very slightly altered in value.

*WORK*

*WORK* is REAL array, dimension (3\**K*)



**INFO**

INFO is INTEGER

= 0: successful exit.

< 0: if INFO = -i, the i-th argument had an illegal value.

> 0: if INFO = 1, a singular value did not converge

**Author**

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

**Date**

June 2017

**Contributors:**

Ming Gu and Huan Ren, Computer Science Division, University of California at Berkeley, USA

**subroutine slasda (integer ICOMPQ, integer SMLSIZ, integer N, integer SQRE, real, dimension( \* ) D, real, dimension( \* ) E, real, dimension( ldu, \* ) U, integer LDU, real, dimension( ldu, \* ) VT, integer, dimension( \* ) K, real, dimension( ldu, \* ) DIFL, real, dimension( ldu, \* ) DIFR, real, dimension( ldu, \* ) Z, real, dimension( ldu, \* ) POLES, integer, dimension( \* ) GIVPTR, integer, dimension( ldgcol, \* ) GIVCOL, integer LDGCOL, integer, dimension( ldgcol, \* ) PERM, real, dimension( ldu, \* ) GIVNUM, real, dimension( \* ) C, real, dimension( \* ) S, real, dimension( \* ) WORK, integer, dimension( \* ) IWORK, integer INFO)**

**SLASDA** computes the singular value decomposition (SVD) of a real upper bidiagonal matrix with diagonal *d* and off-diagonal *e*. Used by *sbdscd*.

**Purpose:**

Using a divide and conquer approach, **SLASDA** computes the singular value decomposition (SVD) of a real upper bidiagonal *N*-by-*M* matrix *B* with diagonal *D* and offdiagonal *E*, where  $M = N + SQRE$ . The algorithm computes the singular values in the SVD  $B = U * S * VT$ . The orthogonal matrices *U* and *VT* are optionally computed in compact form.

A related subroutine, **SLASD0**, computes the singular values and the singular vectors in explicit form.

**Parameters****ICOMPQ**

ICOMPQ is INTEGER

Specifies whether singular vectors are to be computed in compact form, as follows

= 0: Compute singular values only.

= 1: Compute singular vectors of upper bidiagonal matrix in compact form.

**SMLSIZ**

SMLSIZ is INTEGER

The maximum size of the subproblems at the bottom of the computation tree.

**N**

N is INTEGER

The row dimension of the upper bidiagonal matrix. This is also the dimension of the main diagonal array *D*.

**SQRE**

SQRE is INTEGER

Specifies the column dimension of the bidiagonal matrix.

= 0: The bidiagonal matrix has column dimension  $M = N$ ;

= 1: The bidiagonal matrix has column dimension  $M = N + 1$ .

*D*

*D* is REAL array, dimension ( *N* )

On entry *D* contains the main diagonal of the bidiagonal matrix. On exit *D*, if *INFO* = 0, contains its singular values.

*E*

*E* is REAL array, dimension ( *M*-1 )

Contains the subdiagonal entries of the bidiagonal matrix.

On exit, *E* has been destroyed.

*U*

*U* is REAL array,

dimension ( *LDU*, *SMLSIZ* ) if *ICOMPQ* = 1, and not referenced

if *ICOMPQ* = 0. If *ICOMPQ* = 1, on exit, *U* contains the left

singular vector matrices of all subproblems at the bottom level.

*LDU*

*LDU* is INTEGER, *LDU* = > *N*.

The leading dimension of arrays *U*, *VT*, *DIFL*, *DIFR*, *POLES*, *GIVNUM*, and *Z*.

*VT*

*VT* is REAL array,

dimension ( *LDU*, *SMLSIZ*+1 ) if *ICOMPQ* = 1, and not referenced

if *ICOMPQ* = 0. If *ICOMPQ* = 1, on exit, *VT*\*\**T* contains the right

singular vector matrices of all subproblems at the bottom level.

*K*

*K* is INTEGER array, dimension ( *N* )

if *ICOMPQ* = 1 and dimension 1 if *ICOMPQ* = 0.

If *ICOMPQ* = 1, on exit, *K*(*I*) is the dimension of the *I*-th secular equation on the computation tree.

*DIFL*

*DIFL* is REAL array, dimension ( *LDU*, *NLVL* ),

where *NLVL* = floor(log<sub>2</sub> ( *N*/*SMLSIZ* )).

*DIFR*

*DIFR* is REAL array,

dimension ( *LDU*, 2 \* *NLVL* ) if *ICOMPQ* = 1 and

dimension ( *N* ) if *ICOMPQ* = 0.

If *ICOMPQ* = 1, on exit, *DIFL*(1:*N*, *I*) and *DIFR*(1:*N*, 2 \* *I* - 1)

record distances between singular values on the *I*-th

level and singular values on the (*I* - 1)-th level, and

*DIFR*(1:*N*, 2 \* *I*) contains the normalizing factors for

the right singular vector matrix. See SLASD8 for details.

*Z*

*Z* is REAL array,

dimension ( *LDU*, *NLVL* ) if *ICOMPQ* = 1 and

dimension ( *N* ) if *ICOMPQ* = 0.

The first *K* elements of *Z*(1, *I*) contain the components of

the deflation-adjusted updating row vector for subproblems



on the I-th level.

#### *POLES*

POLES is REAL array,  
dimension ( LDU, 2 \* NLVL ) if ICOMPQ = 1, and not referenced  
if ICOMPQ = 0. If ICOMPQ = 1, on exit, POLES(1, 2\*I - 1) and  
POLES(1, 2\*I) contain the new and old singular values  
involved in the secular equations on the I-th level.

#### *GIVPTR*

GIVPTR is INTEGER array,  
dimension ( N ) if ICOMPQ = 1, and not referenced if  
ICOMPQ = 0. If ICOMPQ = 1, on exit, GIVPTR( I ) records  
the number of Givens rotations performed on the I-th  
problem on the computation tree.

#### *GIVCOL*

GIVCOL is INTEGER array,  
dimension ( LDGCOL, 2 \* NLVL ) if ICOMPQ = 1, and not  
referenced if ICOMPQ = 0. If ICOMPQ = 1, on exit, for each I,  
GIVCOL(1, 2 \* I - 1) and GIVCOL(1, 2 \* I) record the locations  
of Givens rotations performed on the I-th level on the  
computation tree.

#### *LDGCOL*

LDGCOL is INTEGER, LDGCOL = > N.  
The leading dimension of arrays GIVCOL and PERM.

#### *PERM*

PERM is INTEGER array, dimension ( LDGCOL, NLVL )  
if ICOMPQ = 1, and not referenced  
if ICOMPQ = 0. If ICOMPQ = 1, on exit, PERM(1, I) records  
permutations done on the I-th level of the computation tree.

#### *GIVNUM*

GIVNUM is REAL array,  
dimension ( LDU, 2 \* NLVL ) if ICOMPQ = 1, and not  
referenced if ICOMPQ = 0. If ICOMPQ = 1, on exit, for each I,  
GIVNUM(1, 2 \* I - 1) and GIVNUM(1, 2 \* I) record the C- and S-  
values of Givens rotations performed on the I-th level on  
the computation tree.

#### *C*

C is REAL array,  
dimension ( N ) if ICOMPQ = 1, and dimension 1 if ICOMPQ = 0.  
If ICOMPQ = 1 and the I-th subproblem is not square, on exit,  
C( I ) contains the C-value of a Givens rotation related to  
the right null space of the I-th subproblem.

#### *S*

S is REAL array, dimension ( N ) if  
ICOMPQ = 1, and dimension 1 if ICOMPQ = 0. If ICOMPQ = 1  
and the I-th subproblem is not square, on exit, S( I )  
contains the S-value of a Givens rotation related to  
the right null space of the I-th subproblem.

#### *WORK*

WORK is REAL array, dimension  
(6 \* N + (SMLSIZ + 1)\*(SMLSIZ + 1)).

#### *IWORK*



IWORK is INTEGER array, dimension (7\*N).

#### INFO

INFO is INTEGER

= 0: successful exit.

< 0: if INFO = -i, the i-th argument had an illegal value.

> 0: if INFO = 1, a singular value did not converge

#### Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

#### Date

December 2016

#### Contributors:

Ming Gu and Huan Ren, Computer Science Division, University of California at Berkeley, USA

**subroutine slasdq (character UPLO, integer SQRE, integer N, integer NCVT, integer NRU, integer NCC, real, dimension( \*) D, real, dimension( \*) E, real, dimension( ldvt, \*) VT, integer LDVT, real, dimension( ldu, \*) U, integer LDU, real, dimension( ldc, \*) C, integer LDC, real, dimension( \*) WORK, integer INFO)**

**SLASDQ** computes the SVD of a real bidiagonal matrix with diagonal d and off-diagonal e. Used by sbdsdc.

#### Purpose:

SLASDQ computes the singular value decomposition (SVD) of a real (upper or lower) bidiagonal matrix with diagonal D and offdiagonal E, accumulating the transformations if desired. Letting B denote the input bidiagonal matrix, the algorithm computes orthogonal matrices Q and P such that  $B = Q * S * P^{*T}$  ( $P^{*T}$  denotes the transpose of P). The singular values S are overwritten on D.

The input matrix U is changed to  $U * Q$  if desired.

The input matrix VT is changed to  $P^{*T} * VT$  if desired.

The input matrix C is changed to  $Q^{*T} * C$  if desired.

See "Computing Small Singular Values of Bidiagonal Matrices With Guaranteed High Relative Accuracy," by J. Demmel and W. Kahan, LAPACK Working Note #3, for a detailed description of the algorithm.

#### Parameters

##### UPLO

UPLO is CHARACTER\*1

On entry, UPLO specifies whether the input bidiagonal matrix is upper or lower bidiagonal, and whether it is square or not.

UPLO = 'U' or 'u' B is upper bidiagonal.

UPLO = 'L' or 'l' B is lower bidiagonal.

##### SQRE

SQRE is INTEGER

= 0: then the input matrix is N-by-N.

= 1: then the input matrix is N-by-(N+1) if UPLU = 'U' and (N+1)-by-N if UPLU = 'L'.

The bidiagonal matrix has

$N = NL + NR + 1$  rows and



$M = N + \text{SQRE} \geq N$  columns.

*N*

*N* is INTEGER

On entry, *N* specifies the number of rows and columns in the matrix. *N* must be at least 0.

*NCVT*

*NCVT* is INTEGER

On entry, *NCVT* specifies the number of columns of the matrix *VT*. *NCVT* must be at least 0.

*NRU*

*NRU* is INTEGER

On entry, *NRU* specifies the number of rows of the matrix *U*. *NRU* must be at least 0.

*NCC*

*NCC* is INTEGER

On entry, *NCC* specifies the number of columns of the matrix *C*. *NCC* must be at least 0.

*D*

*D* is REAL array, dimension (*N*)

On entry, *D* contains the diagonal entries of the bidiagonal matrix whose SVD is desired. On normal exit, *D* contains the singular values in ascending order.

*E*

*E* is REAL array.

dimension is (*N*-1) if *SQRE* = 0 and *N* if *SQRE* = 1. On entry, the entries of *E* contain the offdiagonal entries of the bidiagonal matrix whose SVD is desired. On normal exit, *E* will contain 0. If the algorithm does not converge, *D* and *E* will contain the diagonal and superdiagonal entries of a bidiagonal matrix orthogonally equivalent to the one given as input.

*VT*

*VT* is REAL array, dimension (*LDVT*, *NCVT*)

On entry, contains a matrix which on exit has been premultiplied by  $P^*T$ , dimension *N*-by-*NCVT* if *SQRE* = 0 and (*N*+1)-by-*NCVT* if *SQRE* = 1 (not referenced if *NCVT*=0).

*LDVT*

*LDVT* is INTEGER

On entry, *LDVT* specifies the leading dimension of *VT* as declared in the calling (sub) program. *LDVT* must be at least 1. If *NCVT* is nonzero *LDVT* must also be at least *N*.

*U*

*U* is REAL array, dimension (*LDU*, *N*)

On entry, contains a matrix which on exit has been postmultiplied by *Q*, dimension *NRU*-by-*N* if *SQRE* = 0 and *NRU*-by-(*N*+1) if *SQRE* = 1 (not referenced if *NRU*=0).

*LDU*

*LDU* is INTEGER

On entry, *LDU* specifies the leading dimension of *U* as declared in the calling (sub) program. *LDU* must be at least  $\max(1, \text{NRU})$ .



*C*

*C* is REAL array, dimension (LDC, NCC)  
 On entry, contains an N-by-NCC matrix which on exit  
 has been premultiplied by  $Q^*T$  dimension N-by-NCC if SQRE = 0  
 and (N+1)-by-NCC if SQRE = 1 (not referenced if NCC=0).

*LDC*

LDC is INTEGER  
 On entry, LDC specifies the leading dimension of *C* as  
 declared in the calling (sub) program. LDC must be at  
 least 1. If NCC is nonzero, LDC must also be at least N.

*WORK*

WORK is REAL array, dimension (4\*N)  
 Workspace. Only referenced if one of NCVT, NRU, or NCC is  
 nonzero, and if N is at least 2.

*INFO*

INFO is INTEGER  
 On exit, a value of 0 indicates a successful exit.  
 If INFO < 0, argument number -INFO is illegal.  
 If INFO > 0, the algorithm did not converge, and INFO  
 specifies how many superdiagonals did not converge.

**Author**

Univ. of Tennessee  
 Univ. of California Berkeley  
 Univ. of Colorado Denver  
 NAG Ltd.

**Date**

June 2016

**Contributors:**

Ming Gu and Huan Ren, Computer Science Division, University of California at Berkeley, USA

**subroutine slasdt (integer N, integer LVL, integer ND, integer, dimension( \* ) INODE, integer,  
 dimension( \* ) NDIML, integer, dimension( \* ) NDIMR, integer MSUB)**  
**SLASDT** creates a tree of subproblems for bidiagonal divide and conquer. Used by sbdsdc.

**Purpose:**

SLASDT creates a tree of subproblems for bidiagonal divide and  
 conquer.

**Parameters***N*

*N* is INTEGER  
 On entry, the number of diagonal elements of the  
 bidiagonal matrix.

*LVL*

LVL is INTEGER  
 On exit, the number of levels on the computation tree.

*ND*

ND is INTEGER  
 On exit, the number of nodes on the tree.

*INODE*

INODE is INTEGER array, dimension ( N )





On exit, centers of subproblems.

#### *NDIML*

NDIML is INTEGER array, dimension ( N )

On exit, row dimensions of left children.

#### *NDIMR*

NDIMR is INTEGER array, dimension ( N )

On exit, row dimensions of right children.

#### *MSUB*

MSUB is INTEGER

On entry, the maximum row dimension each subproblem at the bottom of the tree can be of.

#### **Author**

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

#### **Date**

December 2016

#### **Contributors:**

Ming Gu and Huan Ren, Computer Science Division, University of California at Berkeley, USA

**subroutine slaset (character UPLO, integer M, integer N, real ALPHA, real BETA, real, dimension( lda, \* ) A, integer LDA)**

SLASET initializes the off-diagonal elements and the diagonal elements of a matrix to given values.

#### **Purpose:**

SLASET initializes an m-by-n matrix A to BETA on the diagonal and ALPHA on the offdiagonals.

#### **Parameters**

##### *UPLO*

UPLO is CHARACTER\*1

Specifies the part of the matrix A to be set.

= 'U': Upper triangular part is set; the strictly lower triangular part of A is not changed.

= 'L': Lower triangular part is set; the strictly upper triangular part of A is not changed.

Otherwise: All of the matrix A is set.

##### *M*

M is INTEGER

The number of rows of the matrix A.  $M \geq 0$ .

##### *N*

N is INTEGER

The number of columns of the matrix A.  $N \geq 0$ .

##### *ALPHA*

ALPHA is REAL

The constant to which the offdiagonal elements are to be set.

##### *BETA*

BETA is REAL

The constant to which the diagonal elements are to be set.



*A*

*A* is REAL array, dimension (LDA,N)

On exit, the leading m-by-n submatrix of *A* is set as follows:

if UPLO = 'U',  $A(i,j) = \text{ALPHA}$ ,  $1 \leq i \leq j-1$ ,  $1 \leq j \leq n$ ,  
 if UPLO = 'L',  $A(i,j) = \text{ALPHA}$ ,  $j+1 \leq i \leq m$ ,  $1 \leq j \leq n$ ,  
 otherwise,  $A(i,j) = \text{ALPHA}$ ,  $1 \leq i \leq m$ ,  $1 \leq j \leq n$ ,  $i \neq j$ ,

and, for all UPLO,  $A(i,i) = \text{BETA}$ ,  $1 \leq i \leq \min(m,n)$ .

*LDA*

LDA is INTEGER

The leading dimension of the array *A*.  $LDA \geq \max(1,M)$ .

#### Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

#### Date

December 2016

**subroutine slasr (character SIDE, character PIVOT, character DIRECT, integer M, integer N, real, dimension( \* ) C, real, dimension( \* ) S, real, dimension( lda, \* ) A, integer LDA)**

**SLASR** applies a sequence of plane rotations to a general rectangular matrix.

#### Purpose:

SLASR applies a sequence of plane rotations to a real matrix *A*, from either the left or the right.

When SIDE = 'L', the transformation takes the form

$$A := P * A$$

and when SIDE = 'R', the transformation takes the form

$$A := A * P^{**T}$$

where *P* is an orthogonal matrix consisting of a sequence of *z* plane rotations, with *z* = *M* when SIDE = 'L' and *z* = *N* when SIDE = 'R', and *P\*\*T* is the transpose of *P*.

When DIRECT = 'F' (Forward sequence), then

$$P = P(z-1) * \dots * P(2) * P(1)$$

and when DIRECT = 'B' (Backward sequence), then

$$P = P(1) * P(2) * \dots * P(z-1)$$

where *P*(*k*) is a plane rotation matrix defined by the 2-by-2 rotation

$$R(k) = \begin{pmatrix} c(k) & s(k) \\ -s(k) & c(k) \end{pmatrix}$$

When PIVOT = 'V' (Variable pivot), the rotation is performed for the plane (*k*,*k*+1), i.e., *P*(*k*) has the form



$$P(k) = \begin{pmatrix} 1 & & & \\ & \dots & & \\ & & 1 & \\ & & & c(k) & s(k) \\ & & & -s(k) & c(k) \\ & & & & 1 & \\ & & & & & \dots \\ & & & & & & 1 \end{pmatrix}$$

where  $R(k)$  appears as a rank-2 modification to the identity matrix in rows and columns  $k$  and  $k+1$ .

When  $PIVOT = 'T'$  (Top pivot), the rotation is performed for the plane  $(1, k+1)$ , so  $P(k)$  has the form

$$P(k) = \begin{pmatrix} c(k) & & s(k) & & \\ & 1 & & & \\ & & \dots & & \\ & & & 1 & \\ -s(k) & & & c(k) & \\ & & & & 1 & \\ & & & & & \dots \\ & & & & & & 1 \end{pmatrix}$$

where  $R(k)$  appears in rows and columns  $1$  and  $k+1$ .

Similarly, when  $PIVOT = 'B'$  (Bottom pivot), the rotation is performed for the plane  $(k, z)$ , giving  $P(k)$  the form

$$P(k) = \begin{pmatrix} 1 & & & & \\ & \dots & & & \\ & & 1 & & \\ & & & c(k) & s(k) \\ & & & 1 & \\ & & & & \dots \\ & & & & & 1 \\ & & -s(k) & & c(k) \end{pmatrix}$$

where  $R(k)$  appears in rows and columns  $k$  and  $z$ . The rotations are performed without ever forming  $P(k)$  explicitly.

## Parameters

### *SIDE*

*SIDE* is CHARACTER\*1

Specifies whether the plane rotation matrix  $P$  is applied to  $A$  on the left or the right.

= 'L': Left, compute  $A := P * A$

= 'R': Right, compute  $A := A * P^T$

### *PIVOT*

*PIVOT* is CHARACTER\*1

Specifies the plane for which  $P(k)$  is a plane rotation matrix.

= 'V': Variable pivot, the plane  $(k, k+1)$

= 'T': Top pivot, the plane  $(1, k+1)$

= 'B': Bottom pivot, the plane  $(k, z)$

### *DIRECT*

*DIRECT* is CHARACTER\*1



Specifies whether P is a forward or backward sequence of plane rotations.

= 'F': Forward,  $P = P(z-1)*...*P(2)*P(1)$

= 'B': Backward,  $P = P(1)*P(2)*...*P(z-1)$

*M*

*M* is INTEGER

The number of rows of the matrix A. If  $m \leq 1$ , an immediate return is effected.

*N*

*N* is INTEGER

The number of columns of the matrix A. If  $n \leq 1$ , an immediate return is effected.

*C*

*C* is REAL array, dimension

(*M*-1) if *SIDE* = 'L'

(*N*-1) if *SIDE* = 'R'

The cosines *c*(*k*) of the plane rotations.

*S*

*S* is REAL array, dimension

(*M*-1) if *SIDE* = 'L'

(*N*-1) if *SIDE* = 'R'

The sines *s*(*k*) of the plane rotations. The 2-by-2 plane rotation part of the matrix *P*(*k*), *R*(*k*), has the form

$R(k) = \begin{pmatrix} c(k) & s(k) \\ -s(k) & c(k) \end{pmatrix}$

*A*

*A* is REAL array, dimension (*LDA*,*N*)

The *M*-by-*N* matrix A. On exit, A is overwritten by *P*\*A if *SIDE* = 'R' or by *A*\**P*\*\**T* if *SIDE* = 'L'.

*LDA*

*LDA* is INTEGER

The leading dimension of the array A.  $LDA \geq \max(1, M)$ .

#### Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

#### Date

December 2016

**subroutine slassq (integer N, real, dimension( \* ) X, integer INCX, real SCALE, real SUMSQ)**

**SLASSQ** updates a sum of squares represented in scaled form.

#### Purpose:

SLASSQ returns the values *scl* and *sumsq* such that

$$(scl**2)*sumsq = x(1)**2 + ... + x(n)**2 + (scale**2)*sumsq,$$

where  $x(i) = X(1 + (i-1)*INCX)$ . The value of *sumsq* is assumed to be non-negative and *scl* returns the value

$$scl = \max(scale, \max(|x(i)|)).$$



scale and sumsq must be supplied in SCALE and SUMSQ and scl and smsq are overwritten on SCALE and SUMSQ respectively.

The routine makes only one pass through the vector x.

### Parameters

*N*

*N* is INTEGER

The number of elements to be used from the vector X.

*X*

*X* is REAL array, dimension (1+(*N*-1)\**INCX*)

The vector for which a scaled sum of squares is computed.

$x(i) = X(1 + (i - 1) * INCX)$ ,  $1 \leq i \leq n$ .

*INCX*

*INCX* is INTEGER

The increment between successive values of the vector X.

*INCX* > 0.

*SCALE*

*SCALE* is REAL

On entry, the value *scale* in the equation above.

On exit, *SCALE* is overwritten with *scl*, the scaling factor for the sum of squares.

*SUMSQ*

*SUMSQ* is REAL

On entry, the value *sumsq* in the equation above.

On exit, *SUMSQ* is overwritten with *smsq*, the basic sum of squares from which *scl* has been factored out.

### Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

### Date

December 2016

**subroutine slasv2 (real F, real G, real H, real SSMIN, real SSMAX, real SNR, real CSR, real SNL, real CSL)**

**SLASV2** computes the singular value decomposition of a 2-by-2 triangular matrix.

### Purpose:

SLASV2 computes the singular value decomposition of a 2-by-2 triangular matrix

$\begin{bmatrix} F & G \\ 0 & H \end{bmatrix}$ .

On return, *abs(SSMAX)* is the larger singular value, *abs(SSMIN)* is the smaller singular value, and (*CSL*,*SNL*) and (*CSR*,*SNR*) are the left and right singular vectors for *abs(SSMAX)*, giving the decomposition

$\begin{bmatrix} CSL & SNL \end{bmatrix} \begin{bmatrix} F & G \\ 0 & H \end{bmatrix} \begin{bmatrix} CSR & -SNR \end{bmatrix} = \begin{bmatrix} SSMAX & 0 \\ 0 & SSMIN \end{bmatrix}$ .

### Parameters

*F*

*F* is REAL



The (1,1) element of the 2-by-2 matrix.

*G*

*G* is REAL

The (1,2) element of the 2-by-2 matrix.

*H*

*H* is REAL

The (2,2) element of the 2-by-2 matrix.

*SSMIN*

*SSMIN* is REAL

abs(*SSMIN*) is the smaller singular value.

*SSMAX*

*SSMAX* is REAL

abs(*SSMAX*) is the larger singular value.

*SNL*

*SNL* is REAL

*CSL*

*CSL* is REAL

The vector (*CSL*, *SNL*) is a unit left singular vector for the singular value abs(*SSMAX*).

*SNR*

*SNR* is REAL

*CSR*

*CSR* is REAL

The vector (*CSR*, *SNR*) is a unit right singular vector for the singular value abs(*SSMAX*).

#### Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

#### Date

December 2016

#### Further Details:

Any input parameter may be aliased with any output parameter.

Barring over/underflow and assuming a guard digit in subtraction, all output quantities are correct to within a few units in the last place (ulps).

In IEEE arithmetic, the code works correctly if one matrix element is infinite.

Overflow will not occur unless the largest singular value itself overflows or is within a few ulps of overflow. (On machines with partial overflow, like the Cray, overflow may occur if the largest singular value is within a factor of 2 of overflow.)

Underflow is harmless if underflow is gradual. Otherwise, results



may correspond to a matrix modified by perturbations of size near the underflow threshold.

**subroutine xerbla (character\*(\*) SRNAME, integer INFO)**

**XERBLA**

**Purpose:**

XERBLA is an error handler for the LAPACK routines. It is called by an LAPACK routine if an input parameter has an invalid value. A message is printed and execution stops.

Installers may consider modifying the STOP statement in order to call system-specific exception-handling facilities.

**Parameters**

*SRNAME*

SRNAME is CHARACTER\*(\*)

The name of the routine which called XERBLA.

*INFO*

INFO is INTEGER

The position of the invalid parameter in the parameter list of the calling routine.

**Author**

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

**Date**

December 2016

**subroutine xerbla\_array (character(1), dimension(srname\_len) SRNAME\_ARRAY, integer SRNAME\_LEN, integer INFO)**

**XERBLA\_ARRAY**

**Purpose:**

XERBLA\_ARRAY assists other languages in calling XERBLA, the LAPACK and BLAS error handler. Rather than taking a Fortran string argument as the function's name, XERBLA\_ARRAY takes an array of single characters along with the array's length. XERBLA\_ARRAY then copies up to 32 characters of that array into a Fortran string and passes that to XERBLA. If called with a non-positive SRNAME\_LEN, XERBLA\_ARRAY will call XERBLA with a string of all blank characters.

Say some macro or other device makes XERBLA\_ARRAY available to C99 by a name lapack\_xerbla and with a common Fortran calling convention. Then a C99 program could invoke XERBLA via:

```
{
    int flen = strlen(__func__);
    lapack_xerbla(__func__, &flen, &info);
}
```

Providing XERBLA\_ARRAY is not necessary for intercepting LAPACK errors. XERBLA\_ARRAY calls XERBLA.

**Parameters**

*SRNAME\_ARRAY*



SRNAME\_ARRAY is CHARACTER(1) array, dimension (SRNAME\_LEN)  
The name of the routine which called XERBLA\_ARRAY.

*SRNAME\_LEN*

SRNAME\_LEN is INTEGER  
The length of the name in SRNAME\_ARRAY.

*INFO*

INFO is INTEGER  
The position of the invalid parameter in the parameter list  
of the calling routine.

**Author**

Univ. of Tennessee  
Univ. of California Berkeley  
Univ. of Colorado Denver  
NAG Ltd.

**Date**

December 2016

**Author**

Generated automatically by Doxygen for LAPACK from the source code.

