NAME

realOTHERauxiliary

SYNOPSIS

Functions

integer function **ilaslc** (M, N, A, LDA)
  ILASLC scans a matrix for its last non-zero column.
integer function **ilaslr** (M, N, A, LDA)
  ILASLR scans a matrix for its last non-zero row.
subroutine **slabrd** (M, N, NB, A, LDA, D, E, TAUQ, TAUP, X, LDX, Y, LDY)
  SLABRD reduces the first nb rows and columns of a general matrix to a bidiagonal form.
subroutine **slacn2** (N, V, X, ISGN, EST, KASE, ISAVE)
  SLACN2 estimates the 1-norm of a square matrix, using reverse communication for evaluating matrix-vector products.
subroutine **slacon** (N, V, X, ISGN, EST, KASE)
  SLACON estimates the 1-norm of a square matrix, using reverse communication for evaluating matrix-vector products.
subroutine **sladiv** (A, B, C, D, P, Q)
  SLADIV performs complex division in real arithmetic, avoiding unnecessary overflow.
subroutine **sladiv1** (A, B, C, D, P, Q)
real function **sladiv2** (A, B, C, D, R, T)
subroutine **slaein** (RIGHTV, NOINIT, N, H, LDH, WR, WI, VR, VI, B, LDB, WORK, EPS3, SMLNUM, BIGNUM, INFO)
  SLAEIN computes a specified right or left eigenvector of an upper Hessenberg matrix by inverse iteration.
subroutine **slaexc** (WANTQ, N, T, LDT, Q, LDQ, J1, N1, N2, WORK, INFO)
  SLAEXC swaps adjacent diagonal blocks of a real upper quasi-triangular matrix in Schur canonical form, by an orthogonal similarity transformation.
subroutine **slag2** (A, LDA, B, LDB, SAFMIN, SCALE1, SCALE2, WR1, WR2, WI)
  SLAG2 computes the eigenvalues of a 2-by-2 generalized eigenvalue problem, with scaling as necessary to avoid over-/underflow.
subroutine **slags2** (UPPER, A1, A2, A3, B1, B2, B3, CSU, SNU, CSV, SNV, CSQ, SNQ)
  SLAGS2 computes 2-by-2 orthogonal matrices U, V, and Q, and applies them to matrices A and B such that the rows of the transformed A and B are parallel.
subroutine **slagtm** (TRANS, N, NRHS, ALPHA, DL, D, DU, X, LDX, BETA, B, LDB)
  SLAGTM performs a matrix-matrix product of the form $C = \alpha AB+\beta C$, where A is a tridiagonal matrix, B and C are rectangular matrices, and α and β are scalars, which may be 0, 1, or -1.
subroutine **slagv2** (A, LDA, B, LDB, ALPHAR, ALPHAI, BETA, CSL, SNL, CSR, SNR)
  SLAGV2 computes the Generalized Schur factorization of a real 2-by-2 matrix pencil (A,B) where B is upper triangular.
subroutine **slahqr** (WANTT, WANTZ, N, ILO, IHI, H, LDH, WR, WI, ILOZ, IHIZ, Z, LDZ, INFO)
  SLAHQR computes the eigenvalues and Schur factorization of an upper Hessenberg matrix, using the double-shift/single-shift QR algorithm.
subroutine **slahr2** (N, K, NB, A, LDA, TAU, T, LDT, Y, LDY)
  SLAHR2 reduces the specified number of first columns of a general rectangular matrix A so that elements below the specified subdiagonal are zero, and returns auxiliary matrices which are needed to apply the transformation to the unreduced part of A.
subroutine **slaic1** (JOB, J, X, SEST, W, GAMMA, SESTPR, S, C)
  SLAIC1 applies one step of incremental condition estimation.
subroutine **slaln2** (LTRANS, NA, NW, SMIN, CA, A, LDA, D1, D2, B, LDB, WR, WI, X, LDX, SCALE, XNORM, INFO)
  SLALN2 solves a 1-by-1 or 2-by-2 linear system of equations of the specified form.
real function **slangt** (NORM, N, DL, D, DU)
  SLANGT returns the value of the 1-norm, Frobenius norm, infinity-norm, or the largest absolute value of any element of a general tridiagonal matrix.
real function **slanhs** (NORM, N, A, LDA, WORK)
  SLANHS returns the value of the 1-norm, Frobenius norm, infinity-norm, or the largest absolute value of any element of an upper Hessenberg matrix.
real function **slansb** (NORM, UPLO, N, K, AB, LDAB, WORK)

SLANSB returns the value of the 1-norm, or the Frobenius norm, or the infinity norm, or the element of largest absolute value of a symmetric band matrix.

real function **slansp** (NORM, UPLO, N, AP, WORK)
SLANSP returns the value of the 1-norm, or the Frobenius norm, or the infinity norm, or the element of largest absolute value of a symmetric matrix supplied in packed form.

real function **slantb** (NORM, UPLO, DIAG, N, K, AB, LDAB, WORK)
SLANTB returns the value of the 1-norm, or the Frobenius norm, or the infinity norm, or the element of largest absolute value of a triangular band matrix.

real function **slantp** (NORM, UPLO, DIAG, N, AP, WORK)
SLANTP returns the value of the 1-norm, or the Frobenius norm, or the infinity norm, or the element of largest absolute value of a triangular matrix supplied in packed form.

real function **slantr** (NORM, UPLO, DIAG, M, N, A, LDA, WORK)
SLANTR returns the value of the 1-norm, or the Frobenius norm, or the infinity norm, or the element of largest absolute value of a trapezoidal or triangular matrix.

subroutine **slanv2** (A, B, C, D, RT1R, RT1I, RT2R, RT2I, CS, SN)
SLANV2 computes the Schur factorization of a real 2-by-2 nonsymmetric matrix in standard form.

subroutine **slapll** (N, X, INCX, Y, INCY, SSMIN)
SLAPLL measures the linear dependence of two vectors.

subroutine **slapmr** (FORWRD, M, N, X, LDX, K)
SLAPMR rearranges rows of a matrix as specified by a permutation vector.

subroutine **slapmt** (FORWRD, M, N, X, LDX, K)
SLAPMT performs a forward or backward permutation of the columns of a matrix.

subroutine **slaqp2** (M, N, OFFSET, A, LDA, JPVT, TAU, VN1, VN2, WORK)
SLAQP2 computes a QR factorization with column pivoting of the matrix block.

subroutine **slaqps** (M, N, OFFSET, NB, KB, A, LDA, JPVT, TAU, VN1, VN2, AUXV, F, LDF)
SLAQPS computes a step of QR factorization with column pivoting of a real m-by-n matrix A by using BLAS level 3.

subroutine **slaqr0** (WANTT, WANTZ, N, ILO, IHI, H, LDH, WR, WI, ILOZ, IHIZ, Z, LDZ, WORK, LWORK, INFO)
SLAQR0 computes the eigenvalues of a Hessenberg matrix, and optionally the matrices from the Schur decomposition.

subroutine **slaqr1** (N, H, LDH, SR1, SI1, SR2, SI2, V)
SLAQR1 sets a scalar multiple of the first column of the product of 2-by-2 or 3-by-3 matrix H and specified shifts.

subroutine **slaqr2** (WANTT, WANTZ, N, KTOP, KBOT, NW, H, LDH, ILOZ, IHIZ, Z, LDZ, NS, ND, SR, SI, V, LDV, NH, T, LDT, NV, WV, LDWV, WORK, LWORK)
SLAQR2 performs the orthogonal similarity transformation of a Hessenberg matrix to detect and deflate fully converged eigenvalues from a trailing principal submatrix (aggressive early deflation).

subroutine **slaqr3** (WANTT, WANTZ, N, KTOP, KBOT, NW, H, LDH, ILOZ, IHIZ, Z, LDZ, NS, ND, SR, SI, V, LDV, NH, T, LDT, NV, WV, LDWV, WORK, LWORK)
SLAQR3 performs the orthogonal similarity transformation of a Hessenberg matrix to detect and deflate fully converged eigenvalues from a trailing principal submatrix (aggressive early deflation).

subroutine **slaqr4** (WANTT, WANTZ, N, ILO, IHI, H, LDH, WR, WI, ILOZ, IHIZ, Z, LDZ, WORK, LWORK, INFO)
SLAQR4 computes the eigenvalues of a Hessenberg matrix, and optionally the matrices from the Schur decomposition.

subroutine **slaqr5** (WANTT, WANTZ, KACC22, N, KTOP, KBOT, NSHFTS, SR, SI, H, LDH, ILOZ, IHIZ, Z, LDZ, V, LDV, U, LDU, NV, WV, LDWV, NH, WH, LDWH)
SLAQR5 performs a single small-bulge multi-shift QR sweep.

subroutine **slaqsb** (UPLO, N, KD, AB, LDAB, S, SCOND, AMAX, EQUED)
SLAQSB scales a symmetric/Hermitian band matrix, using scaling factors computed by spbequ.

subroutine **slaqsp** (UPLO, N, AP, S, SCOND, AMAX, EQUED)
SLAQSP scales a symmetric/Hermitian matrix in packed storage, using scaling factors computed by sppequ.

subroutine **slaqtr** (LTRAN, LREAL, N, T, LDT, B, W, SCALE, X, WORK, INFO)
SLAQTR solves a real quasi-triangular system of equations, or a complex quasi-triangular

system of special form, in real arithmetic.

subroutine **slar1v** (N, B1, BN, LAMBDA, D, L, LD, LLD, PIVMIN, GAPTOL, Z, WANTNC, NEGCNT, ZTZ, MINGMA, R, ISUPPZ, NRMINV, RESID, RQCORR, WORK)
SLAR1V computes the (scaled) r-th column of the inverse of the submatrix in rows b1 through bn of the tridiagonal matrix LDLT - λI.

subroutine **slar2v** (N, X, Y, Z, INCX, C, S, INCC)
SLAR2V applies a vector of plane rotations with real cosines and real sines from both sides to a sequence of 2-by-2 symmetric/Hermitian matrices.

subroutine **slarf** (SIDE, M, N, V, INCV, TAU, C, LDC, WORK)
SLARF applies an elementary reflector to a general rectangular matrix.

subroutine **slarfb** (SIDE, TRANS, DIRECT, STOREV, M, N, K, V, LDV, T, LDT, C, LDC, WORK, LDWORK)
SLARFB applies a block reflector or its transpose to a general rectangular matrix.

subroutine **slarfg** (N, ALPHA, X, INCX, TAU)
SLARFG generates an elementary reflector (Householder matrix).

subroutine **slarfgp** (N, ALPHA, X, INCX, TAU)
SLARFGP generates an elementary reflector (Householder matrix) with non-negative beta.

subroutine **slarft** (DIRECT, STOREV, N, K, V, LDV, TAU, T, LDT)
SLARFT forms the triangular factor T of a block reflector H = I - vtvH

subroutine **slarfx** (SIDE, M, N, V, TAU, C, LDC, WORK)
SLARFX applies an elementary reflector to a general rectangular matrix, with loop unrolling when the reflector has order ≤ 10.

subroutine **slarfy** (UPLO, N, V, INCV, TAU, C, LDC, WORK)
SLARFY

subroutine **slargv** (N, X, INCX, Y, INCY, C, INCC)
SLARGV generates a vector of plane rotations with real cosines and real sines.

subroutine **slarrv** (N, VL, VU, D, L, PIVMIN, ISPLIT, M, DOL, DOU, MINRGP, RTOL1, RTOL2, W, WERR, WGAP, IBLOCK, INDEXW, GERS, Z, LDZ, ISUPPZ, WORK, IWORK, INFO)
SLARRV computes the eigenvectors of the tridiagonal matrix T = L D LT given L, D and the eigenvalues of L D LT.

subroutine **slartv** (N, X, INCX, Y, INCY, C, S, INCC)
SLARTV applies a vector of plane rotations with real cosines and real sines to the elements of a pair of vectors.

subroutine **slaswp** (N, A, LDA, K1, K2, IPIV, INCX)
SLASWP performs a series of row interchanges on a general rectangular matrix.

subroutine **slatbs** (UPLO, TRANS, DIAG, NORMIN, N, KD, AB, LDAB, X, SCALE, CNORM, INFO)
SLATBS solves a triangular banded system of equations.

subroutine **slatdf** (IJOB, N, Z, LDZ, RHS, RDSUM, RDSCAL, IPIV, JPIV)
SLATDF uses the LU factorization of the n-by-n matrix computed by sgetc2 and computes a contribution to the reciprocal Dif-estimate.

subroutine **slatps** (UPLO, TRANS, DIAG, NORMIN, N, AP, X, SCALE, CNORM, INFO)
SLATPS solves a triangular system of equations with the matrix held in packed storage.

subroutine **slatrs** (UPLO, TRANS, DIAG, NORMIN, N, A, LDA, X, SCALE, CNORM, INFO)
SLATRS solves a triangular system of equations with the scale factor set to prevent overflow.

subroutine **slauu2** (UPLO, N, A, LDA, INFO)
SLAUU2 computes the product UUH or LHL, where U and L are upper or lower triangular matrices (unblocked algorithm).

subroutine **slauum** (UPLO, N, A, LDA, INFO)
SLAUUM computes the product UUH or LHL, where U and L are upper or lower triangular matrices (blocked algorithm).

subroutine **srscl** (N, SA, SX, INCX)
SRSCL multiplies a vector by the reciprocal of a real scalar.

subroutine **stprfb** (SIDE, TRANS, DIRECT, STOREV, M, N, K, L, V, LDV, T, LDT, A, LDA, B, LDB, WORK, LDWORK)
STPRFB applies a real or complex ’triangular-pentagonal’ blocked reflector to a real or complex matrix, which is composed of two blocks.

Detailed Description

This is the group of real other auxiliary routines

Function Documentation

integer function ilaslc (integer M, integer N, real, dimension(lda, *) A, integer LDA)

ILASLC scans a matrix for its last non-zero column.

Purpose:

ILASLC 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 REAL array, dimension (LDA,N)
The m by n matrix A.

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

June 2017

integer function ilaslr (integer M, integer N, real, dimension(lda, *) A, integer LDA)

ILASLR scans a matrix for its last non-zero row.

Purpose:

ILASLR 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 REAL array, dimension (LDA,N)
The m by n matrix A.

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 slabrd (integer M, integer N, integer NB, real, dimension(lda, *) A, integer LDA, real, dimension(*) D, real, dimension(*) E, real, dimension(*) TAUQ, real, dimension(*) TAUP, real, dimension(ldx, *) X, integer LDX, real, dimension(ldy, *) Y, integer LDY)

SLABRD reduces the first nb rows and columns of a general matrix to a bidiagonal form.

Purpose:

```
SLABRD reduces the first NB rows and columns of a real general
m by n matrix A to upper or lower bidiagonal form by an orthogonal
transformation Q**T * A * P, and returns the matrices X and Y which
are needed to apply the transformation to the unreduced part of A.

If m >= n, A is reduced to upper bidiagonal form; if m < n, to lower
bidiagonal form.

This is an auxiliary routine called by SGEBRD
```

Parameters

M

```
M is INTEGER
The number of rows in the matrix A.
```

N

```
N is INTEGER
The number of columns in the matrix A.
```

NB

```
NB is INTEGER
The number of leading rows and columns of A to be reduced.
```

A

```
A is REAL array, dimension (LDA,N)
On entry, the m by n general matrix to be reduced.
On exit, the first NB rows and columns of the matrix are
overwritten; the rest of the array is unchanged.
If m >= n, elements on and below the diagonal in the first NB
  columns, with the array TAUQ, represent the orthogonal
  matrix Q as a product of elementary reflectors; and
  elements above the diagonal in the first NB rows, with the
  array TAUP, represent the orthogonal matrix P as a product
  of elementary reflectors.
If m < n, elements below the diagonal in the first NB
  columns, with the array TAUQ, represent the orthogonal
  matrix Q as a product of elementary reflectors, and
  elements on and above the diagonal in the first NB rows,
  with the array TAUP, represent the orthogonal matrix P as
  a product of elementary reflectors.
See Further Details.
```

LDA

```
LDA is INTEGER
The leading dimension of the array A.  LDA >= max(1,M).
```

D

D is REAL array, dimension (NB)
The diagonal elements of the first NB rows and columns of
the reduced matrix. D(i) = A(i,i).

E

E is REAL array, dimension (NB)
The off-diagonal elements of the first NB rows and columns of
the reduced matrix.

TAUQ

TAUQ is REAL array, dimension (NB)
The scalar factors of the elementary reflectors which
represent the orthogonal matrix Q. See Further Details.

TAUP

TAUP is REAL array, dimension (NB)
The scalar factors of the elementary reflectors which
represent the orthogonal matrix P. See Further Details.

X

X is REAL array, dimension (LDX,NB)
The m-by-nb matrix X required to update the unreduced part
of A.

LDX

LDX is INTEGER
The leading dimension of the array X. LDX >= max(1,M).

Y

Y is REAL array, dimension (LDY,NB)
The n-by-nb matrix Y required to update the unreduced part
of A.

LDY

LDY is INTEGER
The leading dimension of the array Y. LDY >= max(1,N).

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

June 2017

Further Details:

The matrices Q and P are represented as products of elementary
reflectors:

Q = H(1) H(2) . . . H(nb) and P = G(1) G(2) . . . G(nb)

Each H(i) and G(i) has the form:

H(i) = I - tauq * v * v**T and G(i) = I - taup * u * u**T

where tauq and taup are real scalars, and v and u are real vectors.

```
If m >= n, v(1:i-1) = 0, v(i) = 1, and v(i:m) is stored on exit in
A(i:m,i); u(1:i) = 0, u(i+1) = 1, and u(i+1:n) is stored on exit in
A(i,i+1:n); tauq is stored in TAUQ(i) and taup in TAUP(i).

If m < n, v(1:i) = 0, v(i+1) = 1, and v(i+1:m) is stored on exit in
A(i+2:m,i); u(1:i-1) = 0, u(i) = 1, and u(i:n) is stored on exit in
A(i,i+1:n); tauq is stored in TAUQ(i) and taup in TAUP(i).

The elements of the vectors v and u together form the m-by-nb matrix
V and the nb-by-n matrix U**T which are needed, with X and Y, to apply
the transformation to the unreduced part of the matrix, using a block
update of the form:  A := A - V*Y**T - X*U**T.

The contents of A on exit are illustrated by the following examples
with nb = 2:

m = 6 and n = 5 (m > n):          m = 5 and n = 6 (m < n):

  (  1   1   u1  u1  u1 )           (  1   u1  u1  u1  u1  u1 )
  (  v1  1   1   u2  u2 )           (  1   1   u2  u2  u2  u2 )
  (  v1  v2  a   a   a  )           (  v1  1   a   a   a   a  )
  (  v1  v2  a   a   a  )           (  v1  v2  a   a   a   a  )
  (  v1  v2  a   a   a  )           (  v1  v2  a   a   a   a  )
  (  v1  v2  a   a   a  )

where a denotes an element of the original matrix which is unchanged,
vi denotes an element of the vector defining H(i), and ui an element
of the vector defining G(i).
```

subroutine slacn2 (integer N, real, dimension(*) V, real, dimension(*) X, integer, dimension(*) ISGN, real EST, integer KASE, integer, dimension(3) ISAVE)

SLACN2 estimates the 1-norm of a square matrix, using reverse communication for evaluating matrix-vector products.

Purpose:

```
SLACN2 estimates the 1-norm of a square, real matrix A.
Reverse communication is used for evaluating matrix-vector products.
```

Parameters

N

```
 N is INTEGER
The order of the matrix.  N >= 1.
```

V

```
 V is REAL array, dimension (N)
On the final return, V = A*W,  where  EST = norm(V)/norm(W)
(W is not returned).
```

X

```
 X is REAL array, dimension (N)
On an intermediate return, X should be overwritten by
      A * X,   if KASE=1,
      A**T * X,  if KASE=2,
and SLACN2 must be re-called with all the other parameters
unchanged.
```

ISGN

```
 ISGN is INTEGER array, dimension (N)
```

EST

EST is REAL
On entry with KASE = 1 or 2 and ISAVE(1) = 3, EST should be
unchanged from the previous call to SLACN2.
On exit, EST is an estimate (a lower bound) for norm(A).

KASE

KASE is INTEGER
On the initial call to SLACN2, KASE should be 0.
On an intermediate return, KASE will be 1 or 2, indicating
whether X should be overwritten by A * X or A**T * X.
On the final return from SLACN2, KASE will again be 0.

ISAVE

ISAVE is INTEGER array, dimension (3)
ISAVE is used to save variables between calls to SLACN2

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

Further Details:

Originally named SONEST, dated March 16, 1988.

This is a thread safe version of SLACON, which uses the array ISAVE
in place of a SAVE statement, as follows:

```
   SLACON     SLACN2
    JUMP     ISAVE(1)
    J        ISAVE(2)
    ITER     ISAVE(3)
```

Contributors:

Nick Higham, University of Manchester

References:

N.J. Higham, 'FORTRAN codes for estimating the one-norm of
a real or complex matrix, with applications to condition estimation', ACM Trans. Math. Soft., vol. 14, no. 4, pp. 381-396, December 1988.

subroutine slacon (integer N, real, dimension(*) V, real, dimension(*) X, integer, dimension(*) ISGN, real EST, integer KASE)

SLACON estimates the 1-norm of a square matrix, using reverse communication for evaluating matrix-vector products.

Purpose:

SLACON estimates the 1-norm of a square, real matrix A.
Reverse communication is used for evaluating matrix-vector products.

Parameters

N

N is INTEGER
The order of the matrix. N >= 1.

V

V is REAL array, dimension (N)

On the final return, V = A*W, where EST = norm(V)/norm(W)
(W is not returned).

X

X is REAL array, dimension (N)
On an intermediate return, X should be overwritten by
A * X, if KASE=1,
A**T * X, if KASE=2,
and SLACON must be re-called with all the other parameters
unchanged.

ISGN

ISGN is INTEGER array, dimension (N)

EST

EST is REAL
On entry with KASE = 1 or 2 and JUMP = 3, EST should be
unchanged from the previous call to SLACON.
On exit, EST is an estimate (a lower bound) for norm(A).

KASE

KASE is INTEGER
On the initial call to SLACON, KASE should be 0.
On an intermediate return, KASE will be 1 or 2, indicating
whether X should be overwritten by A * X or A**T * X.
On the final return from SLACON, KASE will again be 0.

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

Contributors:

Nick Higham, University of Manchester.
Originally named SONEST, dated March 16, 1988.

References:

N.J. Higham, 'FORTRAN codes for estimating the one-norm of
a real or complex matrix, with applications to condition estimation', ACM Trans. Math. Soft.,
vol. 14, no. 4, pp. 381-396, December 1988.

subroutine sladiv (real A, real B, real C, real D, real P, real Q)

SLADIV performs complex division in real arithmetic, avoiding unnecessary overflow.

Purpose:

SLADIV performs complex division in real arithmetic

```
          a + i*b
p + i*q = ---------
          c + i*d
```

The algorithm is due to Michael Baudin and Robert L. Smith
and can be found in the paper
"A Robust Complex Division in Scilab"

Parameters

A

A is REAL

B

B is REAL

C

C is REAL

D

D is REAL
The scalars a, b, c, and d in the above expression.

P

P is REAL

Q

Q is REAL
The scalars p and q in the above expression.

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

January 2013

subroutine slaein (logical RIGHTV, logical NOINIT, integer N, real, dimension(ldh, *) H, integer LDH, real WR, real WI, real, dimension(*) VR, real, dimension(*) VI, real, dimension(ldb, *) B, integer LDB, real, dimension(*) WORK, real EPS3, real SMLNUM, real BIGNUM, integer INFO)

SLAEIN computes a specified right or left eigenvector of an upper Hessenberg matrix by inverse iteration.

Purpose:

SLAEIN uses inverse iteration to find a right or left eigenvector
corresponding to the eigenvalue (WR,WI) of a real upper Hessenberg
matrix H.

Parameters

RIGHTV

RIGHTV is LOGICAL
= .TRUE. : compute right eigenvector;
= .FALSE.: compute left eigenvector.

NOINIT

NOINIT is LOGICAL
= .TRUE. : no initial vector supplied in (VR,VI).
= .FALSE.: initial vector supplied in (VR,VI).

N

N is INTEGER
The order of the matrix H. N >= 0.

H

H is REAL array, dimension (LDH,N)
The upper Hessenberg matrix H.

LDH

LDH is INTEGER
The leading dimension of the array H. LDH >= max(1,N).

WR

WR is REAL

WI

WI is REAL
The real and imaginary parts of the eigenvalue of H whose corresponding right or left eigenvector is to be computed.

VR

VR is REAL array, dimension (N)

VI

VI is REAL array, dimension (N)
On entry, if NOINIT = .FALSE. and WI = 0.0, VR must contain a real starting vector for inverse iteration using the real eigenvalue WR; if NOINIT = .FALSE. and WI.ne.0.0, VR and VI must contain the real and imaginary parts of a complex starting vector for inverse iteration using the complex eigenvalue (WR,WI); otherwise VR and VI need not be set.
On exit, if WI = 0.0 (real eigenvalue), VR contains the computed real eigenvector; if WI.ne.0.0 (complex eigenvalue), VR and VI contain the real and imaginary parts of the computed complex eigenvector. The eigenvector is normalized so that the component of largest magnitude has magnitude 1; here the magnitude of a complex number (x,y) is taken to be |x| + |y|.
VI is not referenced if WI = 0.0.

B

B is REAL array, dimension (LDB,N)

LDB

LDB is INTEGER
The leading dimension of the array B. LDB >= N+1.

WORK

WORK is REAL array, dimension (N)

EPS3

EPS3 is REAL
A small machine-dependent value which is used to perturb close eigenvalues, and to replace zero pivots.

SMLNUM

SMLNUM is REAL
A machine-dependent value close to the underflow threshold.

BIGNUM

BIGNUM is REAL
A machine-dependent value close to the overflow threshold.

INFO

INFO is INTEGER
= 0: successful exit
= 1: inverse iteration did not converge; VR is set to the
last iterate, and so is VI if WI.ne.0.0.

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

subroutine slaexc (logical WANTQ, integer N, real, dimension(ldt, *) T, integer LDT, real, dimension(ldq, *) Q, integer LDQ, integer J1, integer N1, integer N2, real, dimension(*) WORK, integer INFO)

SLAEXC swaps adjacent diagonal blocks of a real upper quasi-triangular matrix in Schur canonical form, by an orthogonal similarity transformation.

Purpose:

SLAEXC swaps adjacent diagonal blocks T11 and T22 of order 1 or 2 in
an upper quasi-triangular matrix T by an orthogonal similarity
transformation.

T must be in Schur canonical form, that is, block upper triangular
with 1-by-1 and 2-by-2 diagonal blocks; each 2-by-2 diagonal block
has its diagonal elemnts equal and its off-diagonal elements of
opposite sign.

Parameters

WANTQ

WANTQ is LOGICAL
= .TRUE. : accumulate the transformation in the matrix Q;
= .FALSE.: do not accumulate the transformation.

N

N is INTEGER
The order of the matrix T. N >= 0.

T

T is REAL array, dimension (LDT,N)
On entry, the upper quasi-triangular matrix T, in Schur
canonical form.
On exit, the updated matrix T, again in Schur canonical form.

LDT

LDT is INTEGER
The leading dimension of the array T. LDT >= max(1,N).

Q

Q is REAL array, dimension (LDQ,N)
On entry, if WANTQ is .TRUE., the orthogonal matrix Q.
On exit, if WANTQ is .TRUE., the updated matrix Q.
If WANTQ is .FALSE., Q is not referenced.

LDQ

LDQ is INTEGER
The leading dimension of the array Q.
LDQ >= 1; and if WANTQ is .TRUE., LDQ >= N.

J1

J1 is INTEGER
The index of the first row of the first block T11.

N1

N1 is INTEGER
The order of the first block T11. N1 = 0, 1 or 2.

N2

N2 is INTEGER
The order of the second block T22. N2 = 0, 1 or 2.

WORK

WORK is REAL array, dimension (N)

INFO

INFO is INTEGER
= 0: successful exit
= 1: the transformed matrix T would be too far from Schur
form; the blocks are not swapped and T and Q are
unchanged.

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

subroutine slag2 (real, dimension(lda, *) A, integer LDA, real, dimension(ldb, *) B, integer LDB, real SAFMIN, real SCALE1, real SCALE2, real WR1, real WR2, real WI)

SLAG2 computes the eigenvalues of a 2-by-2 generalized eigenvalue problem, with scaling as necessary to avoid over-/underflow.

Purpose:

```
SLAG2 computes the eigenvalues of a 2 x 2 generalized eigenvalue
problem  A - w B, with scaling as necessary to avoid over-/underflow.

The scaling factor "s" results in a modified eigenvalue equation

    s A - w B

where  s  is a non-negative scaling factor chosen so that  w,  w B,
and  s A  do not overflow and, if possible, do not underflow, either.
```

Parameters

A

A is REAL array, dimension (LDA, 2)
On entry, the 2 x 2 matrix A. It is assumed that its 1-norm
is less than 1/SAFMIN. Entries less than
sqrt(SAFMIN)*norm(A) are subject to being treated as zero.

LDA

LDA is INTEGER
The leading dimension of the array A. LDA >= 2.

B

B is REAL array, dimension (LDB, 2)
On entry, the 2 x 2 upper triangular matrix B. It is
assumed that the one-norm of B is less than 1/SAFMIN. The
diagonals should be at least sqrt(SAFMIN) times the largest
element of B (in absolute value); if a diagonal is smaller
than that, then +/- sqrt(SAFMIN) will be used instead of

that diagonal.

LDB

LDB is INTEGER
The leading dimension of the array B. LDB >= 2.

SAFMIN

SAFMIN is REAL
The smallest positive number s.t. 1/SAFMIN does not
overflow. (This should always be SLAMCH('S') -- it is an
argument in order to avoid having to call SLAMCH frequently.)

SCALE1

SCALE1 is REAL
A scaling factor used to avoid over-/underflow in the
eigenvalue equation which defines the first eigenvalue. If
the eigenvalues are complex, then the eigenvalues are
(WR1 +/- WI i) / SCALE1 (which may lie outside the
exponent range of the machine), SCALE1=SCALE2, and SCALE1
will always be positive. If the eigenvalues are real, then
the first (real) eigenvalue is WR1 / SCALE1 , but this may
overflow or underflow, and in fact, SCALE1 may be zero or
less than the underflow threshold if the exact eigenvalue
is sufficiently large.

SCALE2

SCALE2 is REAL
A scaling factor used to avoid over-/underflow in the
eigenvalue equation which defines the second eigenvalue. If
the eigenvalues are complex, then SCALE2=SCALE1. If the
eigenvalues are real, then the second (real) eigenvalue is
WR2 / SCALE2 , but this may overflow or underflow, and in
fact, SCALE2 may be zero or less than the underflow
threshold if the exact eigenvalue is sufficiently large.

WR1

WR1 is REAL
If the eigenvalue is real, then WR1 is SCALE1 times the
eigenvalue closest to the (2,2) element of A B**(-1). If the
eigenvalue is complex, then WR1=WR2 is SCALE1 times the real
part of the eigenvalues.

WR2

WR2 is REAL
If the eigenvalue is real, then WR2 is SCALE2 times the
other eigenvalue. If the eigenvalue is complex, then
WR1=WR2 is SCALE1 times the real part of the eigenvalues.

WI

WI is REAL
If the eigenvalue is real, then WI is zero. If the
eigenvalue is complex, then WI is SCALE1 times the imaginary
part of the eigenvalues. WI will always be non-negative.

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

June 2016

subroutine slags2 (logical UPPER, real A1, real A2, real A3, real B1, real B2, real B3, real CSU, real SNU, real CSV, real SNV, real CSQ, real SNQ)

SLAGS2 computes 2-by-2 orthogonal matrices U, V, and Q, and applies them to matrices A and B such that the rows of the transformed A and B are parallel.

Purpose:

```
SLAGS2 computes 2-by-2 orthogonal matrices U, V and Q, such
that if ( UPPER ) then

     U**T *A*Q = U**T *( A1 A2 )*Q = ( x  0  )
                       ( 0  A3 )     ( x  x  )
and
     V**T*B*Q = V**T *( B1 B2 )*Q = ( x  0  )
                      ( 0  B3 )     ( x  x  )

or if ( .NOT.UPPER ) then

     U**T *A*Q = U**T *( A1 0  )*Q = ( x  x  )
                       ( A2 A3 )     ( 0  x  )
and
     V**T*B*Q = V**T*( B1 0  )*Q = ( x  x  )
                     ( B2 B3 )     ( 0  x  )

The rows of the transformed A and B are parallel, where

  U = (  CSU  SNU ), V = (  CSV SNV ), Q = (  CSQ   SNQ )
      ( -SNU  CSU )      ( -SNV CSV )      ( -SNQ   CSQ )

Z**T denotes the transpose of Z.
```

Parameters

UPPER

UPPER is LOGICAL
= .TRUE.: the input matrices A and B are upper triangular.
= .FALSE.: the input matrices A and B are lower triangular.

A1

A1 is REAL

A2

A2 is REAL

A3

A3 is REAL
On entry, A1, A2 and A3 are elements of the input 2-by-2
upper (lower) triangular matrix A.

B1

B1 is REAL

B2

B2 is REAL

B3

B3 is REAL
On entry, B1, B2 and B3 are elements of the input 2-by-2
upper (lower) triangular matrix B.

CSU

CSU is REAL

SNU

SNU is REAL
The desired orthogonal matrix U.

CSV

CSV is REAL

SNV

SNV is REAL
The desired orthogonal matrix V.

CSQ

CSQ is REAL

SNQ

SNQ is REAL
The desired orthogonal matrix Q.

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

subroutine slagtm (character TRANS, integer N, integer NRHS, real ALPHA, real, dimension(*) DL, real, dimension(*) D, real, dimension(*) DU, real, dimension(ldx, *) X, integer LDX, real BETA, real, dimension(ldb, *) B, integer LDB)

SLAGTM performs a matrix-matrix product of the form $C = \alpha AB+\beta C$, where A is a tridiagonal matrix, B and C are rectangular matrices, and α and β are scalars, which may be 0, 1, or -1.

Purpose:

SLAGTM performs a matrix-vector product of the form

B := alpha * A * X + beta * B

where A is a tridiagonal matrix of order N, B and X are N by NRHS
matrices, and alpha and beta are real scalars, each of which may be
0., 1., or -1.

Parameters

TRANS

TRANS is CHARACTER*1
Specifies the operation applied to A.
= 'N': No transpose, B := alpha * A * X + beta * B
= 'T': Transpose, B := alpha * A'* X + beta * B
= 'C': Conjugate transpose = Transpose

N

N is INTEGER
The order of the matrix A. N >= 0.

NRHS

NRHS is INTEGER
The number of right hand sides, i.e., the number of columns

of the matrices X and B.

ALPHA

ALPHA is REAL
The scalar alpha. ALPHA must be 0., 1., or -1.; otherwise,
it is assumed to be 0.

DL

DL is REAL array, dimension (N-1)
The (n-1) sub-diagonal elements of T.

D

D is REAL array, dimension (N)
The diagonal elements of T.

DU

DU is REAL array, dimension (N-1)
The (n-1) super-diagonal elements of T.

X

X is REAL array, dimension (LDX,NRHS)
The N by NRHS matrix X.

LDX

LDX is INTEGER
The leading dimension of the array X. LDX >= max(N,1).

BETA

BETA is REAL
The scalar beta. BETA must be 0., 1., or -1.; otherwise,
it is assumed to be 1.

B

B is REAL array, dimension (LDB,NRHS)
On entry, the N by NRHS matrix B.
On exit, B is overwritten by the matrix expression
B := alpha * A * X + beta * B.

LDB

LDB is INTEGER
The leading dimension of the array B. LDB >= max(N,1).

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

subroutine slagv2 (real, dimension(lda, *) A, integer LDA, real, dimension(ldb, *) B, integer LDB, real, dimension(2) ALPHAR, real, dimension(2) ALPHAI, real, dimension(2) BETA, real CSL, real SNL, real CSR, real SNR)

SLAGV2 computes the Generalized Schur factorization of a real 2-by-2 matrix pencil (A,B) where B is upper triangular.

Purpose:

SLAGV2 computes the Generalized Schur factorization of a real 2-by-2
matrix pencil (A,B) where B is upper triangular. This routine

computes orthogonal (rotation) matrices given by CSL, SNL and CSR,
SNR such that

1) if the pencil (A,B) has two real eigenvalues (include 0/0 or 1/0
types), then

```
[ a11 a12 ] := [  CSL  SNL ] [ a11 a12 ] [  CSR -SNR ]
[  0  a22 ]    [ -SNL  CSL ] [ a21 a22 ] [  SNR  CSR ]

[ b11 b12 ] := [  CSL  SNL ] [ b11 b12 ] [  CSR -SNR ]
[  0  b22 ]    [ -SNL  CSL ] [  0  b22 ] [  SNR  CSR ],
```

2) if the pencil (A,B) has a pair of complex conjugate eigenvalues,
then

```
[ a11 a12 ] := [  CSL  SNL ] [ a11 a12 ] [  CSR -SNR ]
[ a21 a22 ]    [ -SNL  CSL ] [ a21 a22 ] [  SNR  CSR ]

[ b11  0  ] := [  CSL  SNL ] [ b11 b12 ] [  CSR -SNR ]
[  0  b22 ]    [ -SNL  CSL ] [  0  b22 ] [  SNR  CSR ]
```

where b11 >= b22 > 0.

Parameters

A

A is REAL array, dimension (LDA, 2)
On entry, the 2 x 2 matrix A.
On exit, A is overwritten by the ‘‘A-part’’ of the
generalized Schur form.

LDA

LDA is INTEGER
THe leading dimension of the array A. LDA >= 2.

B

B is REAL array, dimension (LDB, 2)
On entry, the upper triangular 2 x 2 matrix B.
On exit, B is overwritten by the ‘‘B-part’’ of the
generalized Schur form.

LDB

LDB is INTEGER
THe leading dimension of the array B. LDB >= 2.

ALPHAR

ALPHAR is REAL array, dimension (2)

ALPHAI

ALPHAI is REAL array, dimension (2)

BETA

BETA is REAL array, dimension (2)
(ALPHAR(k)+i*ALPHAI(k))/BETA(k) are the eigenvalues of the
pencil (A,B), k=1,2, i = sqrt(-1). Note that BETA(k) may
be zero.

CSL

CSL is REAL
The cosine of the left rotation matrix.

SNL

SNL is REAL
The sine of the left rotation matrix.

CSR

CSR is REAL
The cosine of the right rotation matrix.

SNR

SNR is REAL
The sine of the right rotation matrix.

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

Contributors:

Mark Fahey, Department of Mathematics, Univ. of Kentucky, USA

subroutine slahqr (logical WANTT, logical WANTZ, integer N, integer ILO, integer IHI, real, dimension(ldh, *) H, integer LDH, real, dimension(*) WR, real, dimension(*) WI, integer ILOZ, integer IHIZ, real, dimension(ldz, *) Z, integer LDZ, integer INFO)

SLAHQR computes the eigenvalues and Schur factorization of an upper Hessenberg matrix, using the double-shift/single-shift QR algorithm.

Purpose:

SLAHQR is an auxiliary routine called by SHSEQR to update the
eigenvalues and Schur decomposition already computed by SHSEQR, by
dealing with the Hessenberg submatrix in rows and columns ILO to
IHI.

Parameters

WANTT

WANTT is LOGICAL
= .TRUE. : the full Schur form T is required;
= .FALSE.: only eigenvalues are required.

WANTZ

WANTZ is LOGICAL
= .TRUE. : the matrix of Schur vectors Z is required;
= .FALSE.: Schur vectors are not required.

N

N is INTEGER
The order of the matrix H. N >= 0.

ILO

ILO is INTEGER

IHI

IHI is INTEGER
It is assumed that H is already upper quasi-triangular in
rows and columns IHI+1:N, and that H(ILO,ILO-1) = 0 (unless
ILO = 1). SLAHQR works primarily with the Hessenberg
submatrix in rows and columns ILO to IHI, but applies
transformations to all of H if WANTT is .TRUE..

1 <= ILO <= max(1,IHI); IHI <= N.

H

H is REAL array, dimension (LDH,N)
On entry, the upper Hessenberg matrix H.
On exit, if INFO is zero and if WANTT is .TRUE., H is upper quasi-triangular in rows and columns ILO:IHI, with any 2-by-2 diagonal blocks in standard form. If INFO is zero and WANTT is .FALSE., the contents of H are unspecified on exit. The output state of H if INFO is nonzero is given below under the description of INFO.

LDH

LDH is INTEGER
The leading dimension of the array H. LDH >= max(1,N).

WR

WR is REAL array, dimension (N)

WI

WI is REAL array, dimension (N)
The real and imaginary parts, respectively, of the computed eigenvalues ILO to IHI are stored in the corresponding elements of WR and WI. If two eigenvalues are computed as a complex conjugate pair, they are stored in consecutive elements of WR and WI, say the i-th and (i+1)th, with WI(i) > 0 and WI(i+1) < 0. If WANTT is .TRUE., the eigenvalues are stored in the same order as on the diagonal of the Schur form returned in H, with WR(i) = H(i,i), and, if H(i:i+1,i:i+1) is a 2-by-2 diagonal block, WI(i) = sqrt(H(i+1,i)*H(i,i+1)) and WI(i+1) = -WI(i).

ILOZ

ILOZ is INTEGER

IHIZ

IHIZ is INTEGER
Specify the rows of Z to which transformations must be applied if WANTZ is .TRUE..
1 <= ILOZ <= ILO; IHI <= IHIZ <= N.

Z

Z is REAL array, dimension (LDZ,N)
If WANTZ is .TRUE., on entry Z must contain the current matrix Z of transformations accumulated by SHSEQR, and on exit Z has been updated; transformations are applied only to the submatrix Z(ILOZ:IHIZ,ILO:IHI).
If WANTZ is .FALSE., Z is not referenced.

LDZ

LDZ is INTEGER
The leading dimension of the array Z. LDZ >= max(1,N).

INFO

INFO is INTEGER
= 0: successful exit
> 0: If INFO = i, SLAHQR failed to compute all the eigenvalues ILO to IHI in a total of 30 iterations per eigenvalue; elements i+1:ihi of WR and WI contain those eigenvalues which have been

successfully computed.

If INFO > 0 and WANTT is .FALSE., then on exit,
the remaining unconverged eigenvalues are the
eigenvalues of the upper Hessenberg matrix rows
and columns ILO through INFO of the final, output
value of H.

If INFO > 0 and WANTT is .TRUE., then on exit
(*) (initial value of H)*U = U*(final value of H)
where U is an orthogonal matrix. The final
value of H is upper Hessenberg and triangular in
rows and columns INFO+1 through IHI.

If INFO > 0 and WANTZ is .TRUE., then on exit
(final value of Z) = (initial value of Z)*U
where U is the orthogonal matrix in (*)
(regardless of the value of WANTT.)

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

Further Details:

02-96 Based on modifications by
David Day, Sandia National Laboratory, USA

12-04 Further modifications by
Ralph Byers, University of Kansas, USA
This is a modified version of SLAHQR from LAPACK version 3.0.
It is (1) more robust against overflow and underflow and
(2) adopts the more conservative Ahues & Tisseur stopping
criterion (LAWN 122, 1997).

subroutine slahr2 (integer N, integer K, integer NB, real, dimension(lda, *) A, integer LDA, real, dimension(nb) TAU, real, dimension(ldt, nb) T, integer LDT, real, dimension(ldy, nb) Y, integer LDY)

SLAHR2 reduces the specified number of first columns of a general rectangular matrix A so that elements below the specified subdiagonal are zero, and returns auxiliary matrices which are needed to apply the transformation to the unreduced part of A.

Purpose:

SLAHR2 reduces the first NB columns of A real general n-BY-(n-k+1)
matrix A so that elements below the k-th subdiagonal are zero. The
reduction is performed by an orthogonal similarity transformation
Q**T * A * Q. The routine returns the matrices V and T which determine
Q as a block reflector I - V*T*V**T, and also the matrix Y = A * V * T.

This is an auxiliary routine called by SGEHRD.

Parameters

N

N is INTEGER

The order of the matrix A.

K

K is INTEGER
The offset for the reduction. Elements below the k-th
subdiagonal in the first NB columns are reduced to zero.
K < N.

NB

NB is INTEGER
The number of columns to be reduced.

A

A is REAL array, dimension (LDA,N-K+1)
On entry, the n-by-(n-k+1) general matrix A.
On exit, the elements on and above the k-th subdiagonal in
the first NB columns are overwritten with the corresponding
elements of the reduced matrix; the elements below the k-th
subdiagonal, with the array TAU, represent the matrix Q as a
product of elementary reflectors. The other columns of A are
unchanged. See Further Details.

LDA

LDA is INTEGER
The leading dimension of the array A. LDA >= max(1,N).

TAU

TAU is REAL array, dimension (NB)
The scalar factors of the elementary reflectors. See Further
Details.

T

T is REAL array, dimension (LDT,NB)
The upper triangular matrix T.

LDT

LDT is INTEGER
The leading dimension of the array T. LDT >= NB.

Y

Y is REAL array, dimension (LDY,NB)
The n-by-nb matrix Y.

LDY

LDY is INTEGER
The leading dimension of the array Y. LDY >= N.

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

Further Details:

The matrix Q is represented as a product of nb elementary reflectors

Q = H(1) H(2) . . . H(nb).

```
Each H(i) has the form

   H(i) = I - tau * v * v**T

where tau is a real scalar, and v is a real vector with
v(1:i+k-1) = 0, v(i+k) = 1; v(i+k+1:n) is stored on exit in
A(i+k+1:n,i), and tau in TAU(i).

The elements of the vectors v together form the (n-k+1)-by-nb matrix
V which is needed, with T and Y, to apply the transformation to the
unreduced part of the matrix, using an update of the form:
A := (I - V*T*V**T) * (A - Y*V**T).

The contents of A on exit are illustrated by the following example
with n = 7, k = 3 and nb = 2:

   ( a   a   a   a   a )
   ( a   a   a   a   a )
   ( a   a   a   a   a )
   ( h   h   a   a   a )
   ( v1  h   a   a   a )
   ( v1  v2  a   a   a )
   ( v1  v2  a   a   a )

where a denotes an element of the original matrix A, h denotes a
modified element of the upper Hessenberg matrix H, and vi denotes an
element of the vector defining H(i).

This subroutine is a slight modification of LAPACK-3.0's DLAHRD
incorporating improvements proposed by Quintana-Orti and Van de
Gejin. Note that the entries of A(1:K,2:NB) differ from those
returned by the original LAPACK-3.0's DLAHRD routine. (This
subroutine is not backward compatible with LAPACK-3.0's DLAHRD.)
```

References:

Gregorio Quintana-Orti and Robert van de Geijn, 'Improving the performance of reduction to Hessenberg form,' ACM Transactions on Mathematical Software, 32(2):180-194, June 2006.

subroutine slaic1 (integer JOB, integer J, real, dimension(j) X, real SEST, real, dimension(j) W, real GAMMA, real SESTPR, real S, real C)

SLAIC1 applies one step of incremental condition estimation.

Purpose:

```
SLAIC1 applies one step of incremental condition estimation in
its simplest version:

Let x, twonorm(x) = 1, be an approximate singular vector of an j-by-j
lower triangular matrix L, such that
        twonorm(L*x) = sest
Then SLAIC1 computes sestpr, s, c such that
the vector
               [ s*x ]
        xhat = [  c  ]
is an approximate singular vector of
               [ L      0  ]
        Lhat = [ w**T gamma ]
in the sense that
        twonorm(Lhat*xhat) = sestpr.
```

Depending on JOB, an estimate for the largest or smallest singular value is computed.

Note that [s c]**T and sestpr**2 is an eigenpair of the system

```
diag(sest*sest, 0) + [alpha  gamma] * [ alpha ]
                                      [ gamma ]
```

where alpha = x**T*w.

Parameters

JOB

JOB is INTEGER
= 1: an estimate for the largest singular value is computed.
= 2: an estimate for the smallest singular value is computed.

J

J is INTEGER
Length of X and W

X

X is REAL array, dimension (J)
The j-vector x.

SEST

SEST is REAL
Estimated singular value of j by j matrix L

W

W is REAL array, dimension (J)
The j-vector w.

GAMMA

GAMMA is REAL
The diagonal element gamma.

SESTPR

SESTPR is REAL
Estimated singular value of (j+1) by (j+1) matrix Lhat.

S

S is REAL
Sine needed in forming xhat.

C

C is REAL
Cosine needed in forming xhat.

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

subroutine slaln2 (logical LTRANS, integer NA, integer NW, real SMIN, real CA, real, dimension(lda, *) A, integer LDA, real D1, real D2, real, dimension(ldb, *) B, integer LDB, real WR, real WI, real, dimension(ldx, *) X, integer LDX, real SCALE, real XNORM, integer INFO)
SLALN2 solves a 1-by-1 or 2-by-2 linear system of equations of the specified form.

Purpose:

SLALN2 solves a system of the form (ca A - w D) X = s B
or (ca A**T - w D) X = s B with possible scaling ("s") and
perturbation of A. (A**T means A-transpose.)

A is an NA x NA real matrix, ca is a real scalar, D is an NA x NA
real diagonal matrix, w is a real or complex value, and X and B are
NA x 1 matrices -- real if w is real, complex if w is complex. NA
may be 1 or 2.

If w is complex, X and B are represented as NA x 2 matrices,
the first column of each being the real part and the second
being the imaginary part.

"s" is a scaling factor (<= 1), computed by SLALN2, which is
scaled if necessary to assure that norm(ca A - w D)*norm(X) is less
than overflow.

If both singular values of (ca A - w D) are less than SMIN,
SMIN*identity will be used instead of (ca A - w D). If only one
singular value is less than SMIN, one element of (ca A - w D) will be
perturbed enough to make the smallest singular value roughly SMIN.
If both singular values are at least SMIN, (ca A - w D) will not be
perturbed. In any case, the perturbation will be at most some small
multiple of max(SMIN, ulp*norm(ca A - w D)). The singular values
are computed by infinity-norm approximations, and thus will only be
correct to a factor of 2 or so.

Note: all input quantities are assumed to be smaller than overflow
by a reasonable factor. (See BIGNUM.)

Parameters

LTRANS

LTRANS is LOGICAL
=.TRUE.: A-transpose will be used.
=.FALSE.: A will be used (not transposed.)

NA

NA is INTEGER
The size of the matrix A. It may (only) be 1 or 2.

NW

NW is INTEGER
1 if "w" is real, 2 if "w" is complex. It may only be 1
or 2.

SMIN

SMIN is REAL
The desired lower bound on the singular values of A. This
should be a safe distance away from underflow or overflow,
say, between (underflow/machine precision) and (machine
precision * overflow). (See BIGNUM and ULP.)

CA

CA is REAL
The coefficient c, which A is multiplied by.

A

A is REAL array, dimension (LDA,NA)
The NA x NA matrix A.

LDA

LDA is INTEGER
The leading dimension of A. It must be at least NA.

D1

D1 is REAL
The 1,1 element in the diagonal matrix D.

D2

D2 is REAL
The 2,2 element in the diagonal matrix D. Not used if NA=1.

B

B is REAL array, dimension (LDB,NW)
The NA x NW matrix B (right-hand side). If NW=2 ("w" is complex), column 1 contains the real part of B and column 2 contains the imaginary part.

LDB

LDB is INTEGER
The leading dimension of B. It must be at least NA.

WR

WR is REAL
The real part of the scalar "w".

WI

WI is REAL
The imaginary part of the scalar "w". Not used if NW=1.

X

X is REAL array, dimension (LDX,NW)
The NA x NW matrix X (unknowns), as computed by SLALN2. If NW=2 ("w" is complex), on exit, column 1 will contain the real part of X and column 2 will contain the imaginary part.

LDX

LDX is INTEGER
The leading dimension of X. It must be at least NA.

SCALE

SCALE is REAL
The scale factor that B must be multiplied by to insure that overflow does not occur when computing X. Thus, (ca A - w D) X will be SCALE*B, not B (ignoring perturbations of A.) It will be at most 1.

XNORM

XNORM is REAL
The infinity-norm of X, when X is regarded as an NA x NW real matrix.

INFO

INFO is INTEGER
An error flag. It will be set to zero if no error occurs, a negative number if an argument is in error, or a positive number if ca A - w D had to be perturbed.

```
The possible values are:
= 0: No error occurred, and (ca A - w D) did not have to be
      perturbed.
= 1: (ca A - w D) had to be perturbed to make its smallest
     (or only) singular value greater than SMIN.
NOTE: In the interests of speed, this routine does not
      check the inputs for errors.
```

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

real function slangt (character NORM, integer N, real, dimension(*) DL, real, dimension(*) D, real, dimension(*) DU)

SLANGT returns the value of the 1-norm, Frobenius norm, infinity-norm, or the largest absolute value of any element of a general tridiagonal matrix.

Purpose:

```
SLANGT  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 tridiagonal matrix A.
```

Returns

SLANGT

```
SLANGT = ( 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 SLANGT as described
above.
```

N

```
N is INTEGER
The order of the matrix A.  N >= 0.  When N = 0, SLANGT is
set to zero.
```

DL

```
DL is REAL array, dimension (N-1)
The (n-1) sub-diagonal elements of A.
```

D

```
D is REAL array, dimension (N)
```

The diagonal elements of A.

DU

DU is REAL array, dimension (N-1)
The (n-1) super-diagonal elements of A.

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

real function slanhs (character NORM, integer N, real, dimension(lda, *) A, integer LDA, real, dimension(*) WORK)

SLANHS returns the value of the 1-norm, Frobenius norm, infinity-norm, or the largest absolute value of any element of an upper Hessenberg matrix.

Purpose:

```
SLANHS  returns the value of the one norm,  or the Frobenius norm, or
the  infinity norm,  or the  element of  largest absolute value  of a
Hessenberg matrix A.
```

Returns

SLANHS

```
SLANHS = ( 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 SLANHS as described
above.

N

N is INTEGER
The order of the matrix A. N >= 0. When N = 0, SLANHS is
set to zero.

A

A is REAL array, dimension (LDA,N)
The n by n upper Hessenberg matrix A; the part of A below the
first sub-diagonal is not referenced.

LDA

LDA is INTEGER
The leading dimension of the array A. LDA >= max(N,1).

WORK

WORK is REAL array, dimension (MAX(1,LWORK)),
where LWORK >= N when NORM = 'I'; otherwise, WORK is not
referenced.

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

real function slansb (character NORM, character UPLO, integer N, integer K, real, dimension(ldab, *) AB, integer LDAB, real, dimension(*) WORK)

SLANSB returns the value of the 1-norm, or the Frobenius norm, or the infinity norm, or the element of largest absolute value of a symmetric band matrix.

Purpose:

```
SLANSB  returns the value of the one norm,  or the Frobenius norm, or
the  infinity norm,  or the element of  largest absolute value  of an
n by n symmetric band matrix A,  with k super-diagonals.
```

Returns

SLANSB

```
SLANSB = ( 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 SLANSB as described
above.

UPLO

UPLO is CHARACTER*1
Specifies whether the upper or lower triangular part of the
band matrix A is supplied.
= 'U': Upper triangular part is supplied
= 'L': Lower triangular part is supplied

N

N is INTEGER
The order of the matrix A. N >= 0. When N = 0, SLANSB is
set to zero.

K

K is INTEGER

The number of super-diagonals or sub-diagonals of the
band matrix A. K >= 0.

AB

AB is REAL array, dimension (LDAB,N)
The upper or lower triangle of the symmetric band matrix A,
stored in the first K+1 rows of AB. The j-th column of A is
stored in the j-th column of the array AB as follows:
if UPLO = 'U', AB(k+1+i-j,j) = A(i,j) for max(1,j-k)<=i<=j;
if UPLO = 'L', AB(1+i-j,j) = A(i,j) for j<=i<=min(n,j+k).

LDAB

LDAB is INTEGER
The leading dimension of the array AB. LDAB >= K+1.

WORK

WORK is REAL array, dimension (MAX(1,LWORK)),
where LWORK >= N when NORM = 'I' or '1' or 'O'; otherwise,
WORK is not referenced.

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

real function slansp (character NORM, character UPLO, integer N, real, dimension(*) AP, real, dimension(*) WORK)

SLANSP returns the value of the 1-norm, or the Frobenius norm, or the infinity norm, or the element of largest absolute value of a symmetric matrix supplied in packed form.

Purpose:

```
SLANSP  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 matrix A,  supplied in packed form.
```

Returns

SLANSP

```
SLANSP = ( 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 SLANSP as described
above.

UPLO

UPLO is CHARACTER*1
Specifies whether the upper or lower triangular part of the
symmetric matrix A is supplied.
= 'U': Upper triangular part of A is supplied
= 'L': Lower triangular part of A is supplied

N

N is INTEGER
The order of the matrix A. N >= 0. When N = 0, SLANSP is
set to zero.

AP

AP is REAL array, dimension (N*(N+1)/2)
The upper or lower triangle of the symmetric matrix A, packed
columnwise in a linear array. The j-th column of A is stored
in the array AP as follows:
if UPLO = 'U', AP(i + (j-1)*j/2) = A(i,j) for 1<=i<=j;
if UPLO = 'L', AP(i + (j-1)*(2n-j)/2) = A(i,j) for j<=i<=n.

WORK

WORK is REAL array, dimension (MAX(1,LWORK)),
where LWORK >= N when NORM = 'I' or '1' or 'O'; otherwise,
WORK is not referenced.

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

real function slantb (character NORM, character UPLO, character DIAG, integer N, integer K, real, dimension(ldab, *) AB, integer LDAB, real, dimension(*) WORK)

SLANTB returns the value of the 1-norm, or the Frobenius norm, or the infinity norm, or the element of largest absolute value of a triangular band matrix.

Purpose:

```
SLANTB  returns the value of the one norm,  or the Frobenius norm, or
the  infinity norm,  or the element of  largest absolute value  of an
n by n triangular band matrix A,  with ( k + 1 ) diagonals.
```

Returns

SLANTB

```
   SLANTB = ( 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 SLANTB as described
above.

UPLO

UPLO is CHARACTER*1
Specifies whether the matrix A is upper or lower triangular.
= 'U': Upper triangular
= 'L': Lower triangular

DIAG

DIAG is CHARACTER*1
Specifies whether or not the matrix A is unit triangular.
= 'N': Non-unit triangular
= 'U': Unit triangular

N

N is INTEGER
The order of the matrix A. N >= 0. When N = 0, SLANTB is
set to zero.

K

K is INTEGER
The number of super-diagonals of the matrix A if UPLO = 'U',
or the number of sub-diagonals of the matrix A if UPLO = 'L'.
K >= 0.

AB

AB is REAL array, dimension (LDAB,N)
The upper or lower triangular band matrix A, stored in the
first k+1 rows of AB. The j-th column of A is stored
in the j-th column of the array AB as follows:
if UPLO = 'U', AB(k+1+i-j,j) = A(i,j) for max(1,j-k)<=i<=j;
if UPLO = 'L', AB(1+i-j,j) = A(i,j) for j<=i<=min(n,j+k).
Note that when DIAG = 'U', the elements of the array AB
corresponding to the diagonal elements of the matrix A are
not referenced, but are assumed to be one.

LDAB

LDAB is INTEGER
The leading dimension of the array AB. LDAB >= K+1.

WORK

WORK is REAL array, dimension (MAX(1,LWORK)),
where LWORK >= N when NORM = 'I'; otherwise, WORK is not
referenced.

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

real function slantp (character NORM, character UPLO, character DIAG, integer N, real, dimension(*) AP, real, dimension(*) WORK)

SLANTP returns the value of the 1-norm, or the Frobenius norm, or the infinity norm, or the element of largest absolute value of a triangular matrix supplied in packed form.

Purpose:

SLANTP returns the value of the one norm, or the Frobenius norm, or the infinity norm, or the element of largest absolute value of a triangular matrix A, supplied in packed form.

Returns

SLANTP

```
SLANTP = ( 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 SLANTP as described above.

UPLO

UPLO is CHARACTER*1
Specifies whether the matrix A is upper or lower triangular.
= 'U': Upper triangular
= 'L': Lower triangular

DIAG

DIAG is CHARACTER*1
Specifies whether or not the matrix A is unit triangular.
= 'N': Non-unit triangular
= 'U': Unit triangular

N

N is INTEGER
The order of the matrix A. N >= 0. When N = 0, SLANTP is set to zero.

AP

AP is REAL array, dimension (N*(N+1)/2)
The upper or lower triangular matrix A, packed columnwise in a linear array. The j-th column of A is stored in the array AP as follows:
if UPLO = 'U', AP(i + (j-1)*j/2) = A(i,j) for 1<=i<=j;
if UPLO = 'L', AP(i + (j-1)*(2n-j)/2) = A(i,j) for j<=i<=n.
Note that when DIAG = 'U', the elements of the array AP corresponding to the diagonal elements of the matrix A are not referenced, but are assumed to be one.

WORK

WORK is REAL array, dimension (MAX(1,LWORK)),
where LWORK >= N when NORM = 'I'; otherwise, WORK is not
referenced.

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

real function slantr (character NORM, character UPLO, character DIAG, integer M, integer N, real, dimension(lda, *) A, integer LDA, real, dimension(*) WORK)

SLANTR returns the value of the 1-norm, or the Frobenius norm, or the infinity norm, or the element of largest absolute value of a trapezoidal or triangular matrix.

Purpose:

```
SLANTR  returns the value of the one norm,  or the Frobenius norm, or
the  infinity norm,  or the  element of  largest absolute value  of a
trapezoidal or triangular matrix A.
```

Returns

SLANTR

```
SLANTR = ( 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 SLANTR as described
above.

UPLO

UPLO is CHARACTER*1
Specifies whether the matrix A is upper or lower trapezoidal.
= 'U': Upper trapezoidal
= 'L': Lower trapezoidal
Note that A is triangular instead of trapezoidal if M = N.

DIAG

DIAG is CHARACTER*1
Specifies whether or not the matrix A has unit diagonal.
= 'N': Non-unit diagonal
= 'U': Unit diagonal

M

M is INTEGER

The number of rows of the matrix A. M >= 0, and if
UPLO = 'U', M <= N. When M = 0, SLANTR is set to zero.

N

N is INTEGER
The number of columns of the matrix A. N >= 0, and if
UPLO = 'L', N <= M. When N = 0, SLANTR is set to zero.

A

A is REAL array, dimension (LDA,N)
The trapezoidal matrix A (A is triangular if M = N).
If UPLO = 'U', the leading m by n upper trapezoidal part of
the array A contains the upper trapezoidal matrix, and the
strictly lower triangular part of A is not referenced.
If UPLO = 'L', the leading m by n lower trapezoidal part of
the array A contains the lower trapezoidal matrix, and the
strictly upper triangular part of A is not referenced. Note
that when DIAG = 'U', the diagonal elements of A are not
referenced and are assumed to be one.

LDA

LDA is INTEGER
The leading dimension of the array A. LDA >= max(M,1).

WORK

WORK is REAL array, dimension (MAX(1,LWORK)),
where LWORK >= M when NORM = 'I'; otherwise, WORK is not
referenced.

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

subroutine slanv2 (real A, real B, real C, real D, real RT1R, real RT1I, real RT2R, real RT2I, real CS, real SN)

SLANV2 computes the Schur factorization of a real 2-by-2 nonsymmetric matrix in standard form.

Purpose:

SLANV2 computes the Schur factorization of a real 2-by-2 nonsymmetric
matrix in standard form:

```
[ A  B ] = [ CS -SN ] [ AA  BB ] [ CS  SN ]
[ C  D ]   [ SN  CS ] [ CC  DD ] [-SN  CS ]
```

where either
1) CC = 0 so that AA and DD are real eigenvalues of the matrix, or
2) AA = DD and BB*CC < 0, so that AA + or - sqrt(BB*CC) are complex
conjugate eigenvalues.

Parameters

A

A is REAL

B

B is REAL

C

C is REAL

D

D is REAL
On entry, the elements of the input matrix.
On exit, they are overwritten by the elements of the
standardised Schur form.

RT1R

RT1R is REAL

RT1I

RT1I is REAL

RT2R

RT2R is REAL

RT2I

RT2I is REAL
The real and imaginary parts of the eigenvalues. If the
eigenvalues are a complex conjugate pair, RT1I > 0.

CS

CS is REAL

SN

SN is REAL
Parameters of the rotation matrix.

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

Further Details:

```
Modified by V. Sima, Research Institute for Informatics, Bucharest,
Romania, to reduce the risk of cancellation errors,
when computing real eigenvalues, and to ensure, if possible, that
abs(RT1R) >= abs(RT2R).
```

subroutine slapll (integer N, real, dimension(*) X, integer INCX, real, dimension(*) Y, integer INCY, real SSMIN)

SLAPLL measures the linear dependence of two vectors.

Purpose:

```
Given two column vectors X and Y, let

                   A = ( X Y ).

The subroutine first computes the QR factorization of A = Q*R,
and then computes the SVD of the 2-by-2 upper triangular matrix R.
The smaller singular value of R is returned in SSMIN, which is used
as the measurement of the linear dependency of the vectors X and Y.
```

Parameters

N

N is INTEGER
The length of the vectors X and Y.

X

X is REAL array,
dimension (1+(N-1)*INCX)
On entry, X contains the N-vector X.
On exit, X is overwritten.

INCX

INCX is INTEGER
The increment between successive elements of X. INCX > 0.

Y

Y is REAL array,
dimension (1+(N-1)*INCY)
On entry, Y contains the N-vector Y.
On exit, Y is overwritten.

INCY

INCY is INTEGER
The increment between successive elements of Y. INCY > 0.

SSMIN

SSMIN is REAL
The smallest singular value of the N-by-2 matrix A = (X Y).

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

subroutine slapmr (logical FORWRD, integer M, integer N, real, dimension(ldx, *) X, integer LDX, integer, dimension(*) K)

SLAPMR rearranges rows of a matrix as specified by a permutation vector.

Purpose:

```
SLAPMR rearranges the rows of the M by N matrix X as specified
by the permutation K(1),K(2),...,K(M) of the integers 1,...,M.
If FORWRD = .TRUE.,  forward permutation:

   X(K(I),*) is moved X(I,*) for I = 1,2,...,M.

If FORWRD = .FALSE., backward permutation:

   X(I,*) is moved to X(K(I),*) for I = 1,2,...,M.
```

Parameters

FORWRD

FORWRD is LOGICAL
= .TRUE., forward permutation
= .FALSE., backward permutation

M

M is INTEGER
The number of rows of the matrix X. M >= 0.

N

N is INTEGER
The number of columns of the matrix X. N >= 0.

X

X is REAL array, dimension (LDX,N)
On entry, the M by N matrix X.
On exit, X contains the permuted matrix X.

LDX

LDX is INTEGER
The leading dimension of the array X, LDX >= MAX(1,M).

K

K is INTEGER array, dimension (M)
On entry, K contains the permutation vector. K is used as
internal workspace, but reset to its original value on
output.

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

subroutine slapmt (logical FORWRD, integer M, integer N, real, dimension(ldx, *) X, integer LDX, integer, dimension(*) K)

SLAPMT performs a forward or backward permutation of the columns of a matrix.

Purpose:

```
SLAPMT rearranges the columns of the M by N matrix X as specified
by the permutation K(1),K(2),...,K(N) of the integers 1,...,N.
If FORWRD = .TRUE.,  forward permutation:

    X(*,K(J)) is moved X(*,J) for J = 1,2,...,N.

If FORWRD = .FALSE., backward permutation:

    X(*,J) is moved to X(*,K(J)) for J = 1,2,...,N.
```

Parameters

FORWRD

FORWRD is LOGICAL
= .TRUE., forward permutation
= .FALSE., backward permutation

M

M is INTEGER
The number of rows of the matrix X. M >= 0.

N

N is INTEGER
The number of columns of the matrix X. N >= 0.

X

X is REAL array, dimension (LDX,N)
On entry, the M by N matrix X.
On exit, X contains the permuted matrix X.

LDX

LDX is INTEGER
The leading dimension of the array X, LDX >= MAX(1,M).

K

K is INTEGER array, dimension (N)
On entry, K contains the permutation vector. K is used as
internal workspace, but reset to its original value on
output.

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

subroutine slaqp2 (integer M, integer N, integer OFFSET, real, dimension(lda, *) A, integer LDA, integer, dimension(*) JPVT, real, dimension(*) TAU, real, dimension(*) VN1, real, dimension(*) VN2, real, dimension(*) WORK)

SLAQP2 computes a QR factorization with column pivoting of the matrix block.

Purpose:

SLAQP2 computes a QR factorization with column pivoting of
the block A(OFFSET+1:M,1:N).
The block A(1:OFFSET,1:N) is accordingly pivoted, but not factorized.

Parameters

M

M is INTEGER
The number of rows of the matrix A. M >= 0.

N

N is INTEGER
The number of columns of the matrix A. N >= 0.

OFFSET

OFFSET is INTEGER
The number of rows of the matrix A that must be pivoted
but no factorized. OFFSET >= 0.

A

A is REAL array, dimension (LDA,N)
On entry, the M-by-N matrix A.
On exit, the upper triangle of block A(OFFSET+1:M,1:N) is
the triangular factor obtained; the elements in block
A(OFFSET+1:M,1:N) below the diagonal, together with the
array TAU, represent the orthogonal matrix Q as a product of
elementary reflectors. Block A(1:OFFSET,1:N) has been
accordingly pivoted, but no factorized.

LDA

LDA is INTEGER
The leading dimension of the array A. LDA >= max(1,M).

JPVT

JPVT is INTEGER array, dimension (N)
On entry, if JPVT(i) .ne. 0, the i-th column of A is permuted
to the front of A*P (a leading column); if JPVT(i) = 0,
the i-th column of A is a free column.
On exit, if JPVT(i) = k, then the i-th column of A*P
was the k-th column of A.

TAU

TAU is REAL array, dimension (min(M,N))
The scalar factors of the elementary reflectors.

VN1

VN1 is REAL array, dimension (N)
The vector with the partial column norms.

VN2

VN2 is REAL array, dimension (N)
The vector with the exact column norms.

WORK

WORK is REAL array, dimension (N)

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

Contributors:

G. Quintana-Orti, Depto. de Informatica, Universidad Jaime I, Spain X. Sun, Computer Science Dept., Duke University, USA
Partial column norm updating strategy modified on April 2011 Z. Drmac and Z. Bujanovic, Dept. of Mathematics, University of Zagreb, Croatia.

References:

LAPACK Working Note 176

subroutine slaqps (integer M, integer N, integer OFFSET, integer NB, integer KB, real, dimension(lda, *) A, integer LDA, integer, dimension(*) JPVT, real, dimension(*) TAU, real, dimension(*) VN1, real, dimension(*) VN2, real, dimension(*) AUXV, real, dimension(ldf, *) F, integer LDF)

SLAQPS computes a step of QR factorization with column pivoting of a real m-by-n matrix A by using BLAS level 3.

Purpose:

```
SLAQPS computes a step of QR factorization with column pivoting
of a real M-by-N matrix A by using Blas-3.  It tries to factorize
NB columns from A starting from the row OFFSET+1, and updates all
of the matrix with Blas-3 xGEMM.

In some cases, due to catastrophic cancellations, it cannot
factorize NB columns.  Hence, the actual number of factorized
columns is returned in KB.
```

Block A(1:OFFSET,1:N) is accordingly pivoted, but not factorized.

Parameters

M

M is INTEGER
The number of rows of the matrix A. M >= 0.

N

N is INTEGER
The number of columns of the matrix A. N >= 0

OFFSET

OFFSET is INTEGER
The number of rows of A that have been factorized in
previous steps.

NB

NB is INTEGER
The number of columns to factorize.

KB

KB is INTEGER
The number of columns actually factorized.

A

A is REAL array, dimension (LDA,N)
On entry, the M-by-N matrix A.
On exit, block A(OFFSET+1:M,1:KB) is the triangular
factor obtained and block A(1:OFFSET,1:N) has been
accordingly pivoted, but no factorized.
The rest of the matrix, block A(OFFSET+1:M,KB+1:N) has
been updated.

LDA

LDA is INTEGER
The leading dimension of the array A. LDA >= max(1,M).

JPVT

JPVT is INTEGER array, dimension (N)
JPVT(I) = K <==> Column K of the full matrix A has been
permuted into position I in AP.

TAU

TAU is REAL array, dimension (KB)
The scalar factors of the elementary reflectors.

VN1

VN1 is REAL array, dimension (N)
The vector with the partial column norms.

VN2

VN2 is REAL array, dimension (N)
The vector with the exact column norms.

AUXV

AUXV is REAL array, dimension (NB)
Auxiliary vector.

F

F is REAL array, dimension (LDF,NB)
Matrix F**T = L*Y**T*A.

LDF

LDF is INTEGER
The leading dimension of the array F. LDF >= max(1,N).

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

Contributors:

G. Quintana-Orti, Depto. de Informatica, Universidad Jaime I, Spain X. Sun, Computer Science Dept., Duke University, USA

Partial column norm updating strategy modified on April 2011 Z. Drmac and Z. Bujanovic, Dept. of Mathematics, University of Zagreb, Croatia.

References:

LAPACK Working Note 176

subroutine slaqr0 (logical WANTT, logical WANTZ, integer N, integer ILO, integer IHI, real, dimension(ldh, *) H, integer LDH, real, dimension(*) WR, real, dimension(*) WI, integer ILOZ, integer IHIZ, real, dimension(ldz, *) Z, integer LDZ, real, dimension(*) WORK, integer LWORK, integer INFO)

SLAQR0 computes the eigenvalues of a Hessenberg matrix, and optionally the matrices from the Schur decomposition.

Purpose:

```
SLAQR0 computes the eigenvalues of a Hessenberg matrix H
and, optionally, the matrices T and Z from the Schur decomposition
H = Z T Z**T, where T is an upper quasi-triangular matrix (the
Schur form), and Z is the orthogonal matrix of Schur vectors.

Optionally Z may be postmultiplied into an input orthogonal
matrix Q so that this routine can give the Schur factorization
of a matrix A which has been reduced to the Hessenberg form H
by the orthogonal matrix Q:  A = Q*H*Q**T = (QZ)*T*(QZ)**T.
```

Parameters

WANTT

WANTT is LOGICAL
= .TRUE. : the full Schur form T is required;
= .FALSE.: only eigenvalues are required.

WANTZ

WANTZ is LOGICAL
= .TRUE. : the matrix of Schur vectors Z is required;
= .FALSE.: Schur vectors are not required.

N

N is INTEGER
The order of the matrix H. N >= 0.

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 and, if ILO > 1,
H(ILO,ILO-1) is zero. ILO and IHI are normally set by a
previous call to SGEBAL, and then passed to SGEHRD when the
matrix output by SGEBAL is reduced to Hessenberg form.
Otherwise, ILO and IHI should be set to 1 and N,
respectively. If N > 0, then 1 <= ILO <= IHI <= N.
If N = 0, then ILO = 1 and IHI = 0.

H

H is REAL array, dimension (LDH,N)
On entry, the upper Hessenberg matrix H.
On exit, if INFO = 0 and WANTT is .TRUE., then H contains
the upper quasi-triangular matrix T from the Schur
decomposition (the Schur form); 2-by-2 diagonal blocks
(corresponding to complex conjugate pairs of eigenvalues)
are returned in standard form, with H(i,i) = H(i+1,i+1)
and H(i+1,i)*H(i,i+1) < 0. If INFO = 0 and WANTT is
.FALSE., then the contents of H are unspecified on exit.
(The output value of H when INFO > 0 is given under the
description of INFO below.)

This subroutine may explicitly set H(i,j) = 0 for i > j and
j = 1, 2, ... ILO-1 or j = IHI+1, IHI+2, ... N.

LDH

LDH is INTEGER
The leading dimension of the array H. LDH >= max(1,N).

WR

WR is REAL array, dimension (IHI)

WI

WI is REAL array, dimension (IHI)
The real and imaginary parts, respectively, of the computed
eigenvalues of H(ILO:IHI,ILO:IHI) are stored in WR(ILO:IHI)
and WI(ILO:IHI). If two eigenvalues are computed as a
complex conjugate pair, they are stored in consecutive
elements of WR and WI, say the i-th and (i+1)th, with
WI(i) > 0 and WI(i+1) < 0. If WANTT is .TRUE., then
the eigenvalues are stored in the same order as on the
diagonal of the Schur form returned in H, with
WR(i) = H(i,i) and, if H(i:i+1,i:i+1) is a 2-by-2 diagonal
block, WI(i) = sqrt(-H(i+1,i)*H(i,i+1)) and
WI(i+1) = -WI(i).

ILOZ

ILOZ is INTEGER

IHIZ

IHIZ is INTEGER
Specify the rows of Z to which transformations must be
applied if WANTZ is .TRUE..
1 <= ILOZ <= ILO; IHI <= IHIZ <= N.

Z

Z is REAL array, dimension (LDZ,IHI)
If WANTZ is .FALSE., then Z is not referenced.
If WANTZ is .TRUE., then Z(ILO:IHI,ILOZ:IHIZ) is
replaced by Z(ILO:IHI,ILOZ:IHIZ)*U where U is the

orthogonal Schur factor of H(ILO:IHI,ILO:IHI).
(The output value of Z when INFO > 0 is given under
the description of INFO below.)

LDZ

LDZ is INTEGER
The leading dimension of the array Z. if WANTZ is .TRUE.
then LDZ >= MAX(1,IHIZ). Otherwise, LDZ >= 1.

WORK

WORK is REAL array, dimension LWORK
On exit, if LWORK = -1, WORK(1) returns an estimate of
the optimal value for LWORK.

LWORK

LWORK is INTEGER
The dimension of the array WORK. LWORK >= max(1,N)
is sufficient, but LWORK typically as large as 6*N may
be required for optimal performance. A workspace query
to determine the optimal workspace size is recommended.

If LWORK = -1, then SLAQR0 does a workspace query.
In this case, SLAQR0 checks the input parameters and
estimates the optimal workspace size for the given
values of N, ILO and IHI. The estimate is returned
in WORK(1). No error message related to LWORK is
issued by XERBLA. Neither H nor Z are accessed.

INFO

INFO is INTEGER
= 0: successful exit
> 0: if INFO = i, SLAQR0 failed to compute all of
the eigenvalues. Elements 1:ilo-1 and i+1:n of WR
and WI contain those eigenvalues which have been
successfully computed. (Failures are rare.)

If INFO > 0 and WANT is .FALSE., then on exit,
the remaining unconverged eigenvalues are the eigen-
values of the upper Hessenberg matrix rows and
columns ILO through INFO of the final, output
value of H.

If INFO > 0 and WANTT is .TRUE., then on exit

(*) (initial value of H)*U = U*(final value of H)

where U is an orthogonal matrix. The final
value of H is upper Hessenberg and quasi-triangular
in rows and columns INFO+1 through IHI.

If INFO > 0 and WANTZ is .TRUE., then on exit

(final value of Z(ILO:IHI,ILOZ:IHIZ)
= (initial value of Z(ILO:IHI,ILOZ:IHIZ)*U

where U is the orthogonal matrix in (*) (regard-
less of the value of WANTT.)

If INFO > 0 and WANTZ is .FALSE., then Z is not accessed.

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

Contributors:

Karen Braman and Ralph Byers, Department of Mathematics, University of Kansas, USA

References:

```
K. Braman, R. Byers and R. Mathias, The Multi-Shift QR
Algorithm Part I: Maintaining Well Focused Shifts, and Level 3
Performance, SIAM Journal of Matrix Analysis, volume 23, pages
929--947, 2002.
```

K. Braman, R. Byers and R. Mathias, The Multi-Shift QR Algorithm Part II: Aggressive Early Deflation, SIAM Journal of Matrix Analysis, volume 23, pages 948--973, 2002.

subroutine slaqr1 (integer N, real, dimension(ldh, *) H, integer LDH, real SR1, real SI1, real SR2, real SI2, real, dimension(*) V)

SLAQR1 sets a scalar multiple of the first column of the product of 2-by-2 or 3-by-3 matrix H and specified shifts.

Purpose:

```
Given a 2-by-2 or 3-by-3 matrix H, SLAQR1 sets v to a
scalar multiple of the first column of the product

(*)  K = (H - (sr1 + i*si1)*I)*(H - (sr2 + i*si2)*I)

scaling to avoid overflows and most underflows. It
is assumed that either

      1) sr1 = sr2 and si1 = -si2
   or
      2) si1 = si2 = 0.

This is useful for starting double implicit shift bulges
in the QR algorithm.
```

Parameters

N

```
N is INTEGER
   Order of the matrix H. N must be either 2 or 3.
```

H

```
H is REAL array, dimension (LDH,N)
   The 2-by-2 or 3-by-3 matrix H in (*).
```

LDH

```
LDH is INTEGER
   The leading dimension of H as declared in
   the calling procedure.  LDH >= N
```

SR1

SR1 is REAL

SI1

SI1 is REAL

SR2

SR2 is REAL

SI2

SI2 is REAL
The shifts in (*).

V

V is REAL array, dimension (N)
A scalar multiple of the first column of the
matrix K in (*).

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

June 2017

Contributors:

Karen Braman and Ralph Byers, Department of Mathematics, University of Kansas, USA

subroutine slaqr2 (logical WANTT, logical WANTZ, integer N, integer KTOP, integer KBOT, integer NW, real, dimension(ldh, *) H, integer LDH, integer ILOZ, integer IHIZ, real, dimension(ldz, *) Z, integer LDZ, integer NS, integer ND, real, dimension(*) SR, real, dimension(*) SI, real, dimension(ldv, *) V, integer LDV, integer NH, real, dimension(ldt, *) T, integer LDT, integer NV, real, dimension(ldwv, *) WV, integer LDWV, real, dimension(*) WORK, integer LWORK)

SLAQR2 performs the orthogonal similarity transformation of a Hessenberg matrix to detect and deflate fully converged eigenvalues from a trailing principal submatrix (aggressive early deflation).

Purpose:

```
SLAQR2 is identical to SLAQR3 except that it avoids
recursion by calling SLAHQR instead of SLAQR4.

Aggressive early deflation:

This subroutine accepts as input an upper Hessenberg matrix
H and performs an orthogonal similarity transformation
designed to detect and deflate fully converged eigenvalues from
a trailing principal submatrix.  On output H has been over-
written by a new Hessenberg matrix that is a perturbation of
an orthogonal similarity transformation of H.  It is to be
hoped that the final version of H has many zero subdiagonal
entries.
```

Parameters

WANTT

WANTT is LOGICAL
If .TRUE., then the Hessenberg matrix H is fully updated
so that the quasi-triangular Schur factor may be
computed (in cooperation with the calling subroutine).
If .FALSE., then only enough of H is updated to preserve
the eigenvalues.

WANTZ

WANTZ is LOGICAL
If .TRUE., then the orthogonal matrix Z is updated so
so that the orthogonal Schur factor may be computed
(in cooperation with the calling subroutine).
If .FALSE., then Z is not referenced.

N

N is INTEGER
The order of the matrix H and (if WANTZ is .TRUE.) the
order of the orthogonal matrix Z.

KTOP

KTOP is INTEGER
It is assumed that either KTOP = 1 or H(KTOP,KTOP-1)=0.
KBOT and KTOP together determine an isolated block
along the diagonal of the Hessenberg matrix.

KBOT

KBOT is INTEGER
It is assumed without a check that either
KBOT = N or H(KBOT+1,KBOT)=0. KBOT and KTOP together
determine an isolated block along the diagonal of the
Hessenberg matrix.

NW

NW is INTEGER
Deflation window size. 1 <= NW <= (KBOT-KTOP+1).

H

H is REAL array, dimension (LDH,N)
On input the initial N-by-N section of H stores the
Hessenberg matrix undergoing aggressive early deflation.
On output H has been transformed by an orthogonal
similarity transformation, perturbed, and the returned
to Hessenberg form that (it is to be hoped) has some
zero subdiagonal entries.

LDH

LDH is INTEGER
Leading dimension of H just as declared in the calling
subroutine. N <= LDH

ILOZ

ILOZ is INTEGER

IHIZ

IHIZ is INTEGER
Specify the rows of Z to which transformations must be
applied if WANTZ is .TRUE.. 1 <= ILOZ <= IHIZ <= N.

Z

Z is REAL array, dimension (LDZ,N)
IF WANTZ is .TRUE., then on output, the orthogonal
similarity transformation mentioned above has been
accumulated into Z(ILOZ:IHIZ,ILOZ:IHIZ) from the right.
If WANTZ is .FALSE., then Z is unreferenced.

LDZ

LDZ is INTEGER

The leading dimension of Z just as declared in the calling subroutine. 1 <= LDZ.

NS

NS is INTEGER
The number of unconverged (ie approximate) eigenvalues returned in SR and SI that may be used as shifts by the calling subroutine.

ND

ND is INTEGER
The number of converged eigenvalues uncovered by this subroutine.

SR

SR is REAL array, dimension (KBOT)

SI

SI is REAL array, dimension (KBOT)
On output, the real and imaginary parts of approximate eigenvalues that may be used for shifts are stored in SR(KBOT-ND-NS+1) through SR(KBOT-ND) and SI(KBOT-ND-NS+1) through SI(KBOT-ND), respectively. The real and imaginary parts of converged eigenvalues are stored in SR(KBOT-ND+1) through SR(KBOT) and SI(KBOT-ND+1) through SI(KBOT), respectively.

V

V is REAL array, dimension (LDV,NW)
An NW-by-NW work array.

LDV

LDV is INTEGER
The leading dimension of V just as declared in the calling subroutine. NW <= LDV

NH

NH is INTEGER
The number of columns of T. NH >= NW.

T

T is REAL array, dimension (LDT,NW)

LDT

LDT is INTEGER
The leading dimension of T just as declared in the calling subroutine. NW <= LDT

NV

NV is INTEGER
The number of rows of work array WV available for workspace. NV >= NW.

WV

WV is REAL array, dimension (LDWV,NW)

LDWV

LDWV is INTEGER
The leading dimension of W just as declared in the calling subroutine. NW <= LDV

WORK

WORK is REAL array, dimension (LWORK)
On exit, WORK(1) is set to an estimate of the optimal value
of LWORK for the given values of N, NW, KTOP and KBOT.

LWORK

LWORK is INTEGER
The dimension of the work array WORK. LWORK = 2*NW
suffices, but greater efficiency may result from larger
values of LWORK.

If LWORK = -1, then a workspace query is assumed; SLAQR2
only estimates the optimal workspace size for the given
values of N, NW, KTOP and KBOT. The estimate is returned
in WORK(1). No error message related to LWORK is issued
by XERBLA. Neither H nor Z are accessed.

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

June 2017

Contributors:

Karen Braman and Ralph Byers, Department of Mathematics, University of Kansas, USA

subroutine slaqr3 (logical WANTT, logical WANTZ, integer N, integer KTOP, integer KBOT, integer NW, real, dimension(ldh, *) H, integer LDH, integer ILOZ, integer IHIZ, real, dimension(ldz, *) Z, integer LDZ, integer NS, integer ND, real, dimension(*) SR, real, dimension(*) SI, real, dimension(ldv, *) V, integer LDV, integer NH, real, dimension(ldt, *) T, integer LDT, integer NV, real, dimension(ldwv, *) WV, integer LDWV, real, dimension(*) WORK, integer LWORK)

SLAQR3 performs the orthogonal similarity transformation of a Hessenberg matrix to detect and deflate fully converged eigenvalues from a trailing principal submatrix (aggressive early deflation).

Purpose:

Aggressive early deflation:

SLAQR3 accepts as input an upper Hessenberg matrix
H and performs an orthogonal similarity transformation
designed to detect and deflate fully converged eigenvalues from
a trailing principal submatrix. On output H has been over-
written by a new Hessenberg matrix that is a perturbation of
an orthogonal similarity transformation of H. It is to be
hoped that the final version of H has many zero subdiagonal
entries.

Parameters

WANTT

WANTT is LOGICAL
If .TRUE., then the Hessenberg matrix H is fully updated
so that the quasi-triangular Schur factor may be
computed (in cooperation with the calling subroutine).
If .FALSE., then only enough of H is updated to preserve
the eigenvalues.

WANTZ

WANTZ is LOGICAL
If .TRUE., then the orthogonal matrix Z is updated so

so that the orthogonal Schur factor may be computed
(in cooperation with the calling subroutine).
If .FALSE., then Z is not referenced.

N

N is INTEGER
The order of the matrix H and (if WANTZ is .TRUE.) the
order of the orthogonal matrix Z.

KTOP

KTOP is INTEGER
It is assumed that either KTOP = 1 or H(KTOP,KTOP-1)=0.
KBOT and KTOP together determine an isolated block
along the diagonal of the Hessenberg matrix.

KBOT

KBOT is INTEGER
It is assumed without a check that either
KBOT = N or H(KBOT+1,KBOT)=0. KBOT and KTOP together
determine an isolated block along the diagonal of the
Hessenberg matrix.

NW

NW is INTEGER
Deflation window size. 1 <= NW <= (KBOT-KTOP+1).

H

H is REAL array, dimension (LDH,N)
On input the initial N-by-N section of H stores the
Hessenberg matrix undergoing aggressive early deflation.
On output H has been transformed by an orthogonal
similarity transformation, perturbed, and the returned
to Hessenberg form that (it is to be hoped) has some
zero subdiagonal entries.

LDH

LDH is INTEGER
Leading dimension of H just as declared in the calling
subroutine. N <= LDH

ILOZ

ILOZ is INTEGER

IHIZ

IHIZ is INTEGER
Specify the rows of Z to which transformations must be
applied if WANTZ is .TRUE.. 1 <= ILOZ <= IHIZ <= N.

Z

Z is REAL array, dimension (LDZ,N)
IF WANTZ is .TRUE., then on output, the orthogonal
similarity transformation mentioned above has been
accumulated into Z(ILOZ:IHIZ,ILOZ:IHIZ) from the right.
If WANTZ is .FALSE., then Z is unreferenced.

LDZ

LDZ is INTEGER
The leading dimension of Z just as declared in the
calling subroutine. 1 <= LDZ.

NS

NS is INTEGER
The number of unconverged (ie approximate) eigenvalues
returned in SR and SI that may be used as shifts by the
calling subroutine.

ND

ND is INTEGER
The number of converged eigenvalues uncovered by this
subroutine.

SR

SR is REAL array, dimension (KBOT)

SI

SI is REAL array, dimension (KBOT)
On output, the real and imaginary parts of approximate
eigenvalues that may be used for shifts are stored in
SR(KBOT-ND-NS+1) through SR(KBOT-ND) and
SI(KBOT-ND-NS+1) through SI(KBOT-ND), respectively.
The real and imaginary parts of converged eigenvalues
are stored in SR(KBOT-ND+1) through SR(KBOT) and
SI(KBOT-ND+1) through SI(KBOT), respectively.

V

V is REAL array, dimension (LDV,NW)
An NW-by-NW work array.

LDV

LDV is INTEGER
The leading dimension of V just as declared in the
calling subroutine. NW <= LDV

NH

NH is INTEGER
The number of columns of T. NH >= NW.

T

T is REAL array, dimension (LDT,NW)

LDT

LDT is INTEGER
The leading dimension of T just as declared in the
calling subroutine. NW <= LDT

NV

NV is INTEGER
The number of rows of work array WV available for
workspace. NV >= NW.

WV

WV is REAL array, dimension (LDWV,NW)

LDWV

LDWV is INTEGER
The leading dimension of W just as declared in the
calling subroutine. NW <= LDV

WORK

WORK is REAL array, dimension (LWORK)
On exit, WORK(1) is set to an estimate of the optimal value
of LWORK for the given values of N, NW, KTOP and KBOT.

LWORK

LWORK is INTEGER
The dimension of the work array WORK. LWORK = 2*NW
suffices, but greater efficiency may result from larger
values of LWORK.

If LWORK = -1, then a workspace query is assumed; SLAQR3
only estimates the optimal workspace size for the given
values of N, NW, KTOP and KBOT. The estimate is returned
in WORK(1). No error message related to LWORK is issued
by XERBLA. Neither H nor Z are accessed.

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

June 2016

Contributors:

Karen Braman and Ralph Byers, Department of Mathematics, University of Kansas, USA

subroutine slaqr4 (logical WANTT, logical WANTZ, integer N, integer ILO, integer IHI, real, dimension(ldh, *) H, integer LDH, real, dimension(*) WR, real, dimension(*) WI, integer ILOZ, integer IHIZ, real, dimension(ldz, *) Z, integer LDZ, real, dimension(*) WORK, integer LWORK, integer INFO)

SLAQR4 computes the eigenvalues of a Hessenberg matrix, and optionally the matrices from the Schur decomposition.

Purpose:

SLAQR4 implements one level of recursion for SLAQR0.
It is a complete implementation of the small bulge multi-shift
QR algorithm. It may be called by SLAQR0 and, for large enough
deflation window size, it may be called by SLAQR3. This
subroutine is identical to SLAQR0 except that it calls SLAQR2
instead of SLAQR3.

SLAQR4 computes the eigenvalues of a Hessenberg matrix H
and, optionally, the matrices T and Z from the Schur decomposition
H = Z T Z**T, where T is an upper quasi-triangular matrix (the
Schur form), and Z is the orthogonal matrix of Schur vectors.

Optionally Z may be postmultiplied into an input orthogonal
matrix Q so that this routine can give the Schur factorization
of a matrix A which has been reduced to the Hessenberg form H
by the orthogonal matrix Q: A = Q*H*Q**T = (QZ)*T*(QZ)**T.

Parameters

WANTT

WANTT is LOGICAL
= .TRUE. : the full Schur form T is required;
= .FALSE.: only eigenvalues are required.

WANTZ

WANTZ is LOGICAL
= .TRUE. : the matrix of Schur vectors Z is required;
= .FALSE.: Schur vectors are not required.

N

N is INTEGER
The order of the matrix H. N >= 0.

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 and, if ILO > 1, H(ILO,ILO-1) is zero. ILO and IHI are normally set by a previous call to SGEBAL, and then passed to SGEHRD when the matrix output by SGEBAL is reduced to Hessenberg form. Otherwise, ILO and IHI should be set to 1 and N, respectively. If N > 0, then 1 <= ILO <= IHI <= N. If N = 0, then ILO = 1 and IHI = 0.

H

H is REAL array, dimension (LDH,N)
On entry, the upper Hessenberg matrix H.
On exit, if INFO = 0 and WANTT is .TRUE., then H contains the upper quasi-triangular matrix T from the Schur decomposition (the Schur form); 2-by-2 diagonal blocks (corresponding to complex conjugate pairs of eigenvalues) are returned in standard form, with H(i,i) = H(i+1,i+1) and H(i+1,i)*H(i,i+1) < 0. If INFO = 0 and WANTT is .FALSE., then the contents of H are unspecified on exit. (The output value of H when INFO > 0 is given under the description of INFO below.)

This subroutine may explicitly set H(i,j) = 0 for i > j and j = 1, 2, ... ILO-1 or j = IHI+1, IHI+2, ... N.

LDH

LDH is INTEGER
The leading dimension of the array H. LDH >= max(1,N).

WR

WR is REAL array, dimension (IHI)

WI

WI is REAL array, dimension (IHI)
The real and imaginary parts, respectively, of the computed eigenvalues of H(ILO:IHI,ILO:IHI) are stored in WR(ILO:IHI) and WI(ILO:IHI). If two eigenvalues are computed as a complex conjugate pair, they are stored in consecutive elements of WR and WI, say the i-th and (i+1)th, with WI(i) > 0 and WI(i+1) < 0. If WANTT is .TRUE., then the eigenvalues are stored in the same order as on the diagonal of the Schur form returned in H, with WR(i) = H(i,i) and, if H(i:i+1,i:i+1) is a 2-by-2 diagonal block, WI(i) = sqrt(-H(i+1,i)*H(i,i+1)) and WI(i+1) = -WI(i).

ILOZ

ILOZ is INTEGER

IHIZ

IHIZ is INTEGER

Specify the rows of Z to which transformations must be
applied if WANTZ is .TRUE..
1 <= ILOZ <= ILO; IHI <= IHIZ <= N.

Z

Z is REAL array, dimension (LDZ,IHI)
If WANTZ is .FALSE., then Z is not referenced.
If WANTZ is .TRUE., then Z(ILO:IHI,ILOZ:IHIZ) is
replaced by Z(ILO:IHI,ILOZ:IHIZ)*U where U is the
orthogonal Schur factor of H(ILO:IHI,ILO:IHI).
(The output value of Z when INFO > 0 is given under
the description of INFO below.)

LDZ

LDZ is INTEGER
The leading dimension of the array Z. if WANTZ is .TRUE.
then LDZ >= MAX(1,IHIZ). Otherwise, LDZ >= 1.

WORK

WORK is REAL array, dimension LWORK
On exit, if LWORK = -1, WORK(1) returns an estimate of
the optimal value for LWORK.

LWORK

LWORK is INTEGER
The dimension of the array WORK. LWORK >= max(1,N)
is sufficient, but LWORK typically as large as 6*N may
be required for optimal performance. A workspace query
to determine the optimal workspace size is recommended.

If LWORK = -1, then SLAQR4 does a workspace query.
In this case, SLAQR4 checks the input parameters and
estimates the optimal workspace size for the given
values of N, ILO and IHI. The estimate is returned
in WORK(1). No error message related to LWORK is
issued by XERBLA. Neither H nor Z are accessed.

INFO

INFO is INTEGER
batim
INFO is INTEGER
= 0: successful exit
> 0: if INFO = i, SLAQR4 failed to compute all of
the eigenvalues. Elements 1:ilo-1 and i+1:n of WR
and WI contain those eigenvalues which have been
successfully computed. (Failures are rare.)

If INFO > 0 and WANT is .FALSE., then on exit,
the remaining unconverged eigenvalues are the eigen-
values of the upper Hessenberg matrix rows and
columns ILO through INFO of the final, output
value of H.

If INFO > 0 and WANTT is .TRUE., then on exit

(*) (initial value of H)*U = U*(final value of H)

where U is a orthogonal matrix. The final
value of H is upper Hessenberg and triangular in
rows and columns INFO+1 through IHI.

```
If INFO > 0 and WANTZ is .TRUE., then on exit

  (final value of Z(ILO:IHI,ILOZ:IHIZ)
   =  (initial value of Z(ILO:IHI,ILOZ:IHIZ)*U

where U is the orthogonal matrix in (*) (regard-
less of the value of WANTT.)

If INFO > 0 and WANTZ is .FALSE., then Z is not
accessed.
```

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

Contributors:

Karen Braman and Ralph Byers, Department of Mathematics, University of Kansas, USA

References:

```
K. Braman, R. Byers and R. Mathias, The Multi-Shift QR
Algorithm Part I: Maintaining Well Focused Shifts, and Level 3
Performance, SIAM Journal of Matrix Analysis, volume 23, pages
929--947, 2002.
```

K. Braman, R. Byers and R. Mathias, The Multi-Shift QR Algorithm Part II: Aggressive Early Deflation, SIAM Journal of Matrix Analysis, volume 23, pages 948--973, 2002.

subroutine slaqr5 (logical WANTT, logical WANTZ, integer KACC22, integer N, integer KTOP, integer KBOT, integer NSHFTS, real, dimension(*) SR, real, dimension(*) SI, real, dimension(ldh, *) H, integer LDH, integer ILOZ, integer IHIZ, real, dimension(ldz, *) Z, integer LDZ, real, dimension(ldv, *) V, integer LDV, real, dimension(ldu, *) U, integer LDU, integer NV, real, dimension(ldwv, *) WV, integer LDWV, integer NH, real, dimension(ldwh, *) WH, integer LDWH)

SLAQR5 performs a single small-bulge multi-shift QR sweep.

Purpose:

```
SLAQR5, called by SLAQR0, performs a
single small-bulge multi-shift QR sweep.
```

Parameters

WANTT

```
WANTT is LOGICAL
  WANTT = .true. if the quasi-triangular Schur factor
  is being computed.  WANTT is set to .false. otherwise.
```

WANTZ

```
WANTZ is LOGICAL
  WANTZ = .true. if the orthogonal Schur factor is being
  computed.  WANTZ is set to .false. otherwise.
```

KACC22

```
KACC22 is INTEGER with value 0, 1, or 2.
  Specifies the computation mode of far-from-diagonal
```

orthogonal updates.
= 0: SLAQR5 does not accumulate reflections and does not use matrix-matrix multiply to update far-from-diagonal matrix entries.
= 1: SLAQR5 accumulates reflections and uses matrix-matrix multiply to update the far-from-diagonal matrix entries.
= 2: SLAQR5 accumulates reflections, uses matrix-matrix multiply to update the far-from-diagonal matrix entries, and takes advantage of 2-by-2 block structure during matrix multiplies.

N

N is INTEGER
N is the order of the Hessenberg matrix H upon which this subroutine operates.

KTOP

KTOP is INTEGER

KBOT

KBOT is INTEGER
These are the first and last rows and columns of an isolated diagonal block upon which the QR sweep is to be applied. It is assumed without a check that
either KTOP = 1 or H(KTOP,KTOP-1) = 0
and
either KBOT = N or H(KBOT+1,KBOT) = 0.

NSHFTS

NSHFTS is INTEGER
NSHFTS gives the number of simultaneous shifts. NSHFTS must be positive and even.

SR

SR is REAL array, dimension (NSHFTS)

SI

SI is REAL array, dimension (NSHFTS)
SR contains the real parts and SI contains the imaginary parts of the NSHFTS shifts of origin that define the multi-shift QR sweep. On output SR and SI may be reordered.

H

H is REAL array, dimension (LDH,N)
On input H contains a Hessenberg matrix. On output a multi-shift QR sweep with shifts SR(J)+i*SI(J) is applied to the isolated diagonal block in rows and columns KTOP through KBOT.

LDH

LDH is INTEGER
LDH is the leading dimension of H just as declared in the calling procedure. LDH >= MAX(1,N).

ILOZ

ILOZ is INTEGER

IHIZ

IHIZ is INTEGER
Specify the rows of Z to which transformations must be

applied if WANTZ is .TRUE.. 1 <= ILOZ <= IHIZ <= N

Z

Z is REAL array, dimension (LDZ,IHIZ)
If WANTZ = .TRUE., then the QR Sweep orthogonal
similarity transformation is accumulated into
Z(ILOZ:IHIZ,ILOZ:IHIZ) from the right.
If WANTZ = .FALSE., then Z is unreferenced.

LDZ

LDZ is INTEGER
LDA is the leading dimension of Z just as declared in
the calling procedure. LDZ >= N.

V

V is REAL array, dimension (LDV,NSHFTS/2)

LDV

LDV is INTEGER
LDV is the leading dimension of V as declared in the
calling procedure. LDV >= 3.

U

U is REAL array, dimension (LDU,3*NSHFTS-3)

LDU

LDU is INTEGER
LDU is the leading dimension of U just as declared in the
in the calling subroutine. LDU >= 3*NSHFTS-3.

NV

NV is INTEGER
NV is the number of rows in WV agailable for workspace.
NV >= 1.

WV

WV is REAL array, dimension (LDWV,3*NSHFTS-3)

LDWV

LDWV is INTEGER
LDWV is the leading dimension of WV as declared in the
in the calling subroutine. LDWV >= NV.

NH

NH is INTEGER
NH is the number of columns in array WH available for
workspace. NH >= 1.

WH

WH is REAL array, dimension (LDWH,NH)

LDWH

LDWH is INTEGER
Leading dimension of WH just as declared in the
calling procedure. LDWH >= 3*NSHFTS-3.

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

June 2016

Contributors:

Karen Braman and Ralph Byers, Department of Mathematics, University of Kansas, USA

References:

K. Braman, R. Byers and R. Mathias, The Multi-Shift QR Algorithm Part I: Maintaining Well Focused Shifts, and Level 3 Performance, SIAM Journal of Matrix Analysis, volume 23, pages 929--947, 2002.

subroutine slaqsb (character UPLO, integer N, integer KD, real, dimension(ldab, *) AB, integer LDAB, real, dimension(*) S, real SCOND, real AMAX, character EQUED)

SLAQSB scales a symmetric/Hermitian band matrix, using scaling factors computed by spbequ.

Purpose:

```
SLAQSB equilibrates a symmetric band matrix A using the scaling
factors in the vector S.
```

Parameters

UPLO

```
UPLO is CHARACTER*1
Specifies whether the upper or lower triangular part of the
symmetric matrix A is stored.
= 'U':  Upper triangular
= 'L':  Lower triangular
```

N

```
N is INTEGER
The order of the matrix A.  N >= 0.
```

KD

```
KD is INTEGER
The number of super-diagonals of the matrix A if UPLO = 'U',
or the number of sub-diagonals if UPLO = 'L'.  KD >= 0.
```

AB

```
AB is REAL array, dimension (LDAB,N)
On entry, the upper or lower triangle of the symmetric band
matrix A, stored in the first KD+1 rows of the array.  The
j-th column of A is stored in the j-th column of the array AB
as follows:
if UPLO = 'U', AB(kd+1+i-j,j) = A(i,j) for max(1,j-kd)<=i<=j;
if UPLO = 'L', AB(1+i-j,j)    = A(i,j) for j<=i<=min(n,j+kd).

On exit, if INFO = 0, the triangular factor U or L from the
Cholesky factorization A = U**T*U or A = L*L**T of the band
matrix A, in the same storage format as A.
```

LDAB

```
LDAB is INTEGER
The leading dimension of the array AB.  LDAB >= KD+1.
```

S

```
S is REAL array, dimension (N)
The scale factors for A.
```

SCOND

```
SCOND is REAL
Ratio of the smallest S(i) to the largest S(i).
```

AMAX

AMAX is REAL
Absolute value of largest matrix entry.

EQUED

EQUED is CHARACTER*1
Specifies whether or not equilibration was done.
= 'N': No equilibration.
= 'Y': Equilibration was done, i.e., A has been replaced by
diag(S) * A * diag(S).

Internal Parameters:

THRESH is a threshold value used to decide if scaling should be done
based on the ratio of the scaling factors. If SCOND < THRESH,
scaling is done.

LARGE and SMALL are threshold values used to decide if scaling should
be done based on the absolute size of the largest matrix element.
If AMAX > LARGE or AMAX < SMALL, scaling is done.

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

subroutine slaqsp (character UPLO, integer N, real, dimension(*) AP, real, dimension(*) S, real SCOND, real AMAX, character EQUED)

SLAQSP scales a symmetric/Hermitian matrix in packed storage, using scaling factors computed by sppequ.

Purpose:

SLAQSP equilibrates a symmetric matrix A using the scaling factors
in the vector S.

Parameters

UPLO

UPLO is CHARACTER*1
Specifies whether the upper or lower triangular part of the
symmetric matrix A is stored.
= 'U': Upper triangular
= 'L': Lower triangular

N

N is INTEGER
The order of the matrix A. N >= 0.

AP

AP is REAL array, dimension (N*(N+1)/2)
On entry, the upper or lower triangle of the symmetric matrix
A, packed columnwise in a linear array. The j-th column of A
is stored in the array AP as follows:
if UPLO = 'U', AP(i + (j-1)*j/2) = A(i,j) for 1<=i<=j;
if UPLO = 'L', AP(i + (j-1)*(2n-j)/2) = A(i,j) for j<=i<=n.

On exit, the equilibrated matrix: diag(S) * A * diag(S), in
the same storage format as A.

S

S is REAL array, dimension (N)
The scale factors for A.

SCOND

SCOND is REAL
Ratio of the smallest S(i) to the largest S(i).

AMAX

AMAX is REAL
Absolute value of largest matrix entry.

EQUED

EQUED is CHARACTER*1
Specifies whether or not equilibration was done.
= 'N': No equilibration.
= 'Y': Equilibration was done, i.e., A has been replaced by
diag(S) * A * diag(S).

Internal Parameters:

THRESH is a threshold value used to decide if scaling should be done
based on the ratio of the scaling factors. If SCOND < THRESH,
scaling is done.

LARGE and SMALL are threshold values used to decide if scaling should
be done based on the absolute size of the largest matrix element.
If AMAX > LARGE or AMAX < SMALL, scaling is done.

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

subroutine slaqtr (logical LTRAN, logical LREAL, integer N, real, dimension(ldt, *) T, integer LDT, real, dimension(*) B, real W, real SCALE, real, dimension(*) X, real, dimension(*) WORK, integer INFO)

SLAQTR solves a real quasi-triangular system of equations, or a complex quasi-triangular system of special form, in real arithmetic.

Purpose:

SLAQTR solves the real quasi-triangular system

op(T)*p = scale*c, if LREAL = .TRUE.

or the complex quasi-triangular systems

op(T + iB)*(p+iq) = scale*(c+id), if LREAL = .FALSE.

in real arithmetic, where T is upper quasi-triangular.
If LREAL = .FALSE., then the first diagonal block of T must be
1 by 1, B is the specially structured matrix

```
B = [ b(1) b(2) ... b(n) ]
    [       w            ]
    [           w        ]
    [              .     ]
    [                 w  ]
```

op(A) = A or A**T, A**T denotes the transpose of
matrix A.

```
On input, X = [ c ].  On output, X = [ p ].
              [ d ]                  [ q ]
```

This subroutine is designed for the condition number estimation
in routine STRSNA.

Parameters

LTRAN

LTRAN is LOGICAL
On entry, LTRAN specifies the option of conjugate transpose:
= .FALSE., op(T+i*B) = T+i*B,
= .TRUE., op(T+i*B) = (T+i*B)**T.

LREAL

LREAL is LOGICAL
On entry, LREAL specifies the input matrix structure:
= .FALSE., the input is complex
= .TRUE., the input is real

N

N is INTEGER
On entry, N specifies the order of T+i*B. N >= 0.

T

T is REAL array, dimension (LDT,N)
On entry, T contains a matrix in Schur canonical form.
If LREAL = .FALSE., then the first diagonal block of T must
be 1 by 1.

LDT

LDT is INTEGER
The leading dimension of the matrix T. LDT >= max(1,N).

B

B is REAL array, dimension (N)
On entry, B contains the elements to form the matrix
B as described above.
If LREAL = .TRUE., B is not referenced.

W

W is REAL
On entry, W is the diagonal element of the matrix B.
If LREAL = .TRUE., W is not referenced.

SCALE

SCALE is REAL
On exit, SCALE is the scale factor.

X

X is REAL array, dimension (2*N)
On entry, X contains the right hand side of the system.

On exit, X is overwritten by the solution.

WORK

WORK is REAL array, dimension (N)

INFO

INFO is INTEGER
On exit, INFO is set to
  0: successful exit.
    1: the some diagonal 1 by 1 block has been perturbed by
      a small number SMIN to keep nonsingularity.
    2: the some diagonal 2 by 2 block has been perturbed by
      a small number in SLALN2 to keep nonsingularity.
NOTE: In the interests of speed, this routine does not
    check the inputs for errors.

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

subroutine slar1v (integer N, integer B1, integer BN, real LAMBDA, real, dimension(*) D, real, dimension(*) L, real, dimension(*) LD, real, dimension(*) LLD, real PIVMIN, real GAPTOL, real, dimension(*) Z, logical WANTNC, integer NEGCNT, real ZTZ, real MINGMA, integer R, integer, dimension(*) ISUPPZ, real NRMINV, real RESID, real RQCORR, real, dimension(*) WORK)

SLAR1V computes the (scaled) r-th column of the inverse of the submatrix in rows b1 through bn of the tridiagonal matrix LDLT - λI.

Purpose:

```
SLAR1V computes the (scaled) r-th column of the inverse of
the sumbmatrix in rows B1 through BN of the tridiagonal matrix
L D L**T - sigma I. When sigma is close to an eigenvalue, the
computed vector is an accurate eigenvector. Usually, r corresponds
to the index where the eigenvector is largest in magnitude.
The following steps accomplish this computation :
(a) Stationary qd transform,  L D L**T - sigma I = L(+) D(+) L(+)**T,
(b) Progressive qd transform, L D L**T - sigma I = U(-) D(-) U(-)**T,
(c) Computation of the diagonal elements of the inverse of
   L D L**T - sigma I by combining the above transforms, and choosing
   r as the index where the diagonal of the inverse is (one of the)
   largest in magnitude.
(d) Computation of the (scaled) r-th column of the inverse using the
   twisted factorization obtained by combining the top part of the
   the stationary and the bottom part of the progressive transform.
```

Parameters

N

N is INTEGER
 The order of the matrix L D L**T.

B1

B1 is INTEGER
 First index of the submatrix of L D L**T.

BN

BN is INTEGER
Last index of the submatrix of L D L**T.

LAMBDA

LAMBDA is REAL
The shift. In order to compute an accurate eigenvector,
LAMBDA should be a good approximation to an eigenvalue
of L D L**T.

L

L is REAL array, dimension (N-1)
The (n-1) subdiagonal elements of the unit bidiagonal matrix
L, in elements 1 to N-1.

D

D is REAL array, dimension (N)
The n diagonal elements of the diagonal matrix D.

LD

LD is REAL array, dimension (N-1)
The n-1 elements L(i)*D(i).

LLD

LLD is REAL array, dimension (N-1)
The n-1 elements L(i)*L(i)*D(i).

PIVMIN

PIVMIN is REAL
The minimum pivot in the Sturm sequence.

GAPTOL

GAPTOL is REAL
Tolerance that indicates when eigenvector entries are negligible
w.r.t. their contribution to the residual.

Z

Z is REAL array, dimension (N)
On input, all entries of Z must be set to 0.
On output, Z contains the (scaled) r-th column of the
inverse. The scaling is such that Z(R) equals 1.

WANTNC

WANTNC is LOGICAL
Specifies whether NEGCNT has to be computed.

NEGCNT

NEGCNT is INTEGER
If WANTNC is .TRUE. then NEGCNT = the number of pivots < pivmin
in the matrix factorization L D L**T, and NEGCNT = -1 otherwise.

ZTZ

ZTZ is REAL
The square of the 2-norm of Z.

MINGMA

MINGMA is REAL
The reciprocal of the largest (in magnitude) diagonal
element of the inverse of L D L**T - sigma I.

R

R is INTEGER

The twist index for the twisted factorization used to
compute Z.
On input, 0 <= R <= N. If R is input as 0, R is set to
the index where (L D L**T - sigma I)^{-1} is largest
in magnitude. If 1 <= R <= N, R is unchanged.
On output, R contains the twist index used to compute Z.
Ideally, R designates the position of the maximum entry in the
eigenvector.

ISUPPZ

ISUPPZ is INTEGER array, dimension (2)
The support of the vector in Z, i.e., the vector Z is
nonzero only in elements ISUPPZ(1) through ISUPPZ(2).

NRMINV

NRMINV is REAL
NRMINV = 1/SQRT(ZTZ)

RESID

RESID is REAL
The residual of the FP vector.
RESID = ABS(MINGMA)/SQRT(ZTZ)

RQCORR

RQCORR is REAL
The Rayleigh Quotient correction to LAMBDA.
RQCORR = MINGMA*TMP

WORK

WORK is REAL array, dimension (4*N)

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 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 slar2v (integer N, real, dimension(*) X, real, dimension(*) Y, real, dimension(*) Z, integer INCX, real, dimension(*) C, real, dimension(*) S, integer INCC)

SLAR2V applies a vector of plane rotations with real cosines and real sines from both sides to a sequence of 2-by-2 symmetric/Hermitian matrices.

Purpose:

```
SLAR2V applies a vector of real plane rotations from both sides to
a sequence of 2-by-2 real symmetric matrices, defined by the elements
of the vectors x, y and z. For i = 1,2,...,n

   ( x(i)  z(i) ) := (  c(i)  s(i) ) ( x(i)  z(i) ) ( c(i) -s(i) )
   ( z(i)  y(i) )    ( -s(i)  c(i) ) ( z(i)  y(i) ) ( s(i)  c(i) )
```

Parameters

N

N is INTEGER
The number of plane rotations to be applied.

X

X is REAL array,
dimension (1+(N-1)*INCX)
The vector x.

Y

Y is REAL array,
dimension (1+(N-1)*INCX)
The vector y.

Z

Z is REAL array,
dimension (1+(N-1)*INCX)
The vector z.

INCX

INCX is INTEGER
The increment between elements of X, Y and Z. INCX > 0.

C

C is REAL array, dimension (1+(N-1)*INCC)
The cosines of the plane rotations.

S

S is REAL array, dimension (1+(N-1)*INCC)
The sines of the plane rotations.

INCC

INCC is INTEGER
The increment between elements of C and S. INCC > 0.

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

subroutine slarf (character SIDE, integer M, integer N, real, dimension(*) V, integer INCV, real TAU, real, dimension(ldc, *) C, integer LDC, real, dimension(*) WORK)

SLARF applies an elementary reflector to a general rectangular matrix.

Purpose:

```
SLARF applies a real elementary reflector H to a real m by n matrix
C, from either the left or the right. H is represented in the form

   H = I - tau * v * v**T

where tau is a real scalar and v is a real vector.

If tau = 0, then H is taken to be the unit matrix.
```

Parameters

SIDE

```
SIDE is CHARACTER*1
= 'L': form  H * C
= 'R': form  C * H
```

M

```
M is INTEGER
The number of rows of the matrix C.
```

N

```
N is INTEGER
The number of columns of the matrix C.
```

V

```
V is REAL array, dimension
           (1 + (M-1)*abs(INCV)) if SIDE = 'L'
        or (1 + (N-1)*abs(INCV)) if SIDE = 'R'
The vector v in the representation of H. V is not used if
TAU = 0.
```

INCV

```
INCV is INTEGER
The increment between elements of v. INCV <> 0.
```

TAU

```
TAU is REAL
The value tau in the representation of H.
```

C

```
C is REAL array, dimension (LDC,N)
On entry, the m by n matrix C.
On exit, C is overwritten by the matrix H * C if SIDE = 'L',
or C * H if SIDE = 'R'.
```

LDC

```
LDC is INTEGER
The leading dimension of the array C. LDC >= max(1,M).
```

WORK

```
WORK is REAL array, dimension
              (N) if SIDE = 'L'
           or (M) if SIDE = 'R'
```

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

subroutine slarfb (character SIDE, character TRANS, character DIRECT, character STOREV, integer M, integer N, integer K, real, dimension(ldv, *) V, integer LDV, real, dimension(ldt, *) T, integer LDT, real, dimension(ldc, *) C, integer LDC, real, dimension(ldwork, *) WORK, integer LDWORK)

SLARFB applies a block reflector or its transpose to a general rectangular matrix.

Purpose:

```
SLARFB applies a real block reflector H or its transpose H**T to a
real m by n matrix C, from either the left or the right.
```

Parameters

SIDE

SIDE is CHARACTER*1
= 'L': apply H or H**T from the Left
= 'R': apply H or H**T from the Right

TRANS

TRANS is CHARACTER*1
= 'N': apply H (No transpose)
= 'T': apply H**T (Transpose)

DIRECT

DIRECT is CHARACTER*1
Indicates how H is formed from a product of elementary
reflectors
= 'F': H = H(1) H(2) . . . H(k) (Forward)
= 'B': H = H(k) . . . H(2) H(1) (Backward)

STOREV

STOREV is CHARACTER*1
Indicates how the vectors which define the elementary
reflectors are stored:
= 'C': Columnwise
= 'R': Rowwise

M

M is INTEGER
The number of rows of the matrix C.

N

N is INTEGER
The number of columns of the matrix C.

K

K is INTEGER
The order of the matrix T (= the number of elementary
reflectors whose product defines the block reflector).
If SIDE = 'L', M >= K >= 0;
if SIDE = 'R', N >= K >= 0.

V

V is REAL array, dimension
(LDV,K) if STOREV = 'C'
(LDV,M) if STOREV = 'R' and SIDE = 'L'
(LDV,N) if STOREV = 'R' and SIDE = 'R'
The matrix V. See Further Details.

LDV

LDV is INTEGER
The leading dimension of the array V.
If STOREV = 'C' and SIDE = 'L', LDV >= max(1,M);
if STOREV = 'C' and SIDE = 'R', LDV >= max(1,N);
if STOREV = 'R', LDV >= K.

T

T is REAL array, dimension (LDT,K)
The triangular k by k matrix T in the representation of the
block reflector.

LDT

LDT is INTEGER
The leading dimension of the array T. LDT >= K.

C

C is REAL array, dimension (LDC,N)
On entry, the m by n matrix C.
On exit, C is overwritten by H*C or H**T*C or C*H or C*H**T.

LDC

LDC is INTEGER
The leading dimension of the array C. LDC >= max(1,M).

WORK

WORK is REAL array, dimension (LDWORK,K)

LDWORK

LDWORK is INTEGER
The leading dimension of the array WORK.
If SIDE = 'L', LDWORK >= max(1,N);
if SIDE = 'R', LDWORK >= max(1,M).

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

June 2013

Further Details:

```
The shape of the matrix V and the storage of the vectors which define
the H(i) is best illustrated by the following example with n = 5 and
k = 3. The elements equal to 1 are not stored; the corresponding
array elements are modified but restored on exit. The rest of the
array is not used.

DIRECT = 'F' and STOREV = 'C':         DIRECT = 'F' and STOREV = 'R':

             V = (  1       )                 V = (  1 v1 v1 v1 v1 )
                 ( v1  1    )                     (     1 v2 v2 v2 )
                 ( v1 v2  1 )                     (        1 v3 v3 )
                 ( v1 v2 v3 )
                 ( v1 v2 v3 )

DIRECT = 'B' and STOREV = 'C':         DIRECT = 'B' and STOREV = 'R':

             V = ( v1 v2 v3 )                 V = ( v1 v1  1       )
                 ( v1 v2 v3 )                     ( v2 v2 v2  1    )
                 (  1 v2 v3 )                     ( v3 v3 v3 v3  1 )
                 (     1 v3 )
                 (        1 )
```

subroutine slarfg (integer N, real ALPHA, real, dimension(*) X, integer INCX, real TAU)

SLARFG generates an elementary reflector (Householder matrix).

Purpose:

SLARFG generates a real elementary reflector H of order n, such that

```
H * ( alpha ) = ( beta ),   H**T * H = I.
    (   x   )   (   0  )
```

where alpha and beta are scalars, and x is an (n-1)-element real vector. H is represented in the form

```
H = I - tau * ( 1 ) * ( 1 v**T ) ,
              ( v )
```

where tau is a real scalar and v is a real (n-1)-element vector.

If the elements of x are all zero, then tau = 0 and H is taken to be the unit matrix.

Otherwise 1 <= tau <= 2.

Parameters

N

N is INTEGER
The order of the elementary reflector.

ALPHA

ALPHA is REAL
On entry, the value alpha.
On exit, it is overwritten with the value beta.

X

X is REAL array, dimension
(1+(N-2)*abs(INCX))
On entry, the vector x.
On exit, it is overwritten with the vector v.

INCX

INCX is INTEGER
The increment between elements of X. INCX > 0.

TAU

TAU is REAL
The value tau.

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

November 2017

subroutine slarfgp (integer N, real ALPHA, real, dimension(*) X, integer INCX, real TAU)

SLARFGP generates an elementary reflector (Householder matrix) with non-negative beta.

Purpose:

SLARFGP generates a real elementary reflector H of order n, such that

```
    H * ( alpha ) = ( beta ),   H**T * H = I.
        (   x   )   (   0  )

where alpha and beta are scalars, beta is non-negative, and x is
an (n-1)-element real vector.  H is represented in the form

    H = I - tau * ( 1 ) * ( 1 v**T ) ,
                  ( v )

where tau is a real scalar and v is a real (n-1)-element
vector.

If the elements of x are all zero, then tau = 0 and H is taken to be
the unit matrix.
```

Parameters

N

N is INTEGER
The order of the elementary reflector.

ALPHA

ALPHA is REAL
On entry, the value alpha.
On exit, it is overwritten with the value beta.

X

X is REAL array, dimension
(1+(N-2)*abs(INCX))
On entry, the vector x.
On exit, it is overwritten with the vector v.

INCX

INCX is INTEGER
The increment between elements of X. INCX > 0.

TAU

TAU is REAL
The value tau.

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

November 2017

subroutine slarft (character DIRECT, character STOREV, integer N, integer K, real, dimension(ldv, *) V, integer LDV, real, dimension(*) TAU, real, dimension(ldt, *) T, integer LDT)

SLARFT forms the triangular factor T of a block reflector H = I - vtvH

Purpose:

```
SLARFT forms the triangular factor T of a real block reflector H
of order n, which is defined as a product of k elementary reflectors.

If DIRECT = 'F', H = H(1) H(2) . . . H(k) and T is upper triangular;

If DIRECT = 'B', H = H(k) . . . H(2) H(1) and T is lower triangular.
```

If STOREV = 'C', the vector which defines the elementary reflector
H(i) is stored in the i-th column of the array V, and

H = I - V * T * V**T

If STOREV = 'R', the vector which defines the elementary reflector
H(i) is stored in the i-th row of the array V, and

H = I - V**T * T * V

Parameters

DIRECT

DIRECT is CHARACTER*1
Specifies the order in which the elementary reflectors are
multiplied to form the block reflector:
= 'F': H = H(1) H(2) . . . H(k) (Forward)
= 'B': H = H(k) . . . H(2) H(1) (Backward)

STOREV

STOREV is CHARACTER*1
Specifies how the vectors which define the elementary
reflectors are stored (see also Further Details):
= 'C': columnwise
= 'R': rowwise

N

N is INTEGER
The order of the block reflector H. N >= 0.

K

K is INTEGER
The order of the triangular factor T (= the number of
elementary reflectors). K >= 1.

V

V is REAL array, dimension
(LDV,K) if STOREV = 'C'
(LDV,N) if STOREV = 'R'
The matrix V. See further details.

LDV

LDV is INTEGER
The leading dimension of the array V.
If STOREV = 'C', LDV >= max(1,N); if STOREV = 'R', LDV >= K.

TAU

TAU is REAL array, dimension (K)
TAU(i) must contain the scalar factor of the elementary
reflector H(i).

T

T is REAL array, dimension (LDT,K)
The k by k triangular factor T of the block reflector.
If DIRECT = 'F', T is upper triangular; if DIRECT = 'B', T is
lower triangular. The rest of the array is not used.

LDT

LDT is INTEGER
The leading dimension of the array T. LDT >= K.

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

Further Details:

```
The shape of the matrix V and the storage of the vectors which define
the H(i) is best illustrated by the following example with n = 5 and
k = 3. The elements equal to 1 are not stored.

DIRECT = 'F' and STOREV = 'C':         DIRECT = 'F' and STOREV = 'R':

         V = (  1       )                 V = (  1 v1 v1 v1 v1 )
             ( v1  1    )                     (     1 v2 v2 v2 )
             ( v1 v2  1 )                     (        1 v3 v3 )
             ( v1 v2 v3 )
             ( v1 v2 v3 )

DIRECT = 'B' and STOREV = 'C':         DIRECT = 'B' and STOREV = 'R':

         V = ( v1 v2 v3 )                 V = ( v1 v1  1       )
             ( v1 v2 v3 )                     ( v2 v2 v2  1    )
             (  1 v2 v3 )                     ( v3 v3 v3 v3  1 )
             (     1 v3 )
             (        1 )
```

subroutine slarfx (character SIDE, integer M, integer N, real, dimension(*) V, real TAU, real, dimension(ldc, *) C, integer LDC, real, dimension(*) WORK)

SLARFX applies an elementary reflector to a general rectangular matrix, with loop unrolling when the reflector has order ≤ 10.

Purpose:

```
SLARFX applies a real elementary reflector H to a real m by n
matrix C, from either the left or the right. H is represented in the
form

     H = I - tau * v * v**T

where tau is a real scalar and v is a real vector.

If tau = 0, then H is taken to be the unit matrix

This version uses inline code if H has order < 11.
```

Parameters

SIDE

```
SIDE is CHARACTER*1
= 'L': form  H * C
= 'R': form  C * H
```

M

```
M is INTEGER
The number of rows of the matrix C.
```

N

N is INTEGER
The number of columns of the matrix C.

V

V is REAL array, dimension (M) if SIDE = 'L'
or (N) if SIDE = 'R'
The vector v in the representation of H.

TAU

TAU is REAL
The value tau in the representation of H.

C

C is REAL array, dimension (LDC,N)
On entry, the m by n matrix C.
On exit, C is overwritten by the matrix H * C if SIDE = 'L',
or C * H if SIDE = 'R'.

LDC

LDC is INTEGER
The leading dimension of the array C. LDC >= (1,M).

WORK

WORK is REAL array, dimension
(N) if SIDE = 'L'
or (M) if SIDE = 'R'
WORK is not referenced if H has order < 11.

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

subroutine slarfy (character UPLO, integer N, real, dimension(*) V, integer INCV, real TAU, real, dimension(ldc, *) C, integer LDC, real, dimension(*) WORK)
SLARFY

Purpose:

SLARFY applies an elementary reflector, or Householder matrix, H,
to an n x n symmetric matrix C, from both the left and the right.

H is represented in the form

H = I - tau * v * v'

where tau is a scalar and v is a vector.

If tau is zero, then H is taken to be the unit matrix.

Parameters

UPLO

UPLO is CHARACTER*1
Specifies whether the upper or lower triangular part of the
symmetric matrix C is stored.
= 'U': Upper triangle
= 'L': Lower triangle

N

N is INTEGER
The number of rows and columns of the matrix C. N >= 0.

V

V is REAL array, dimension
(1 + (N-1)*abs(INCV))
The vector v as described above.

INCV

INCV is INTEGER
The increment between successive elements of v. INCV must
not be zero.

TAU

TAU is REAL
The value tau as described above.

C

C is REAL array, dimension (LDC, N)
On entry, the matrix C.
On exit, C is overwritten by H * C * H'.

LDC

LDC is INTEGER
The leading dimension of the array C. LDC >= max(1, N).

WORK

WORK is REAL array, dimension (N)

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

subroutine slargv (integer N, real, dimension(*) X, integer INCX, real, dimension(*) Y, integer INCY, real, dimension(*) C, integer INCC)

SLARGV generates a vector of plane rotations with real cosines and real sines.

Purpose:

SLARGV generates a vector of real plane rotations, determined by
elements of the real vectors x and y. For i = 1,2,...,n

```
  (  c(i)  s(i) ) ( x(i) ) = ( a(i) )
  ( -s(i)  c(i) ) ( y(i) ) = (   0  )
```

Parameters

N

N is INTEGER
The number of plane rotations to be generated.

X

X is REAL array,
dimension (1+(N-1)*INCX)
On entry, the vector x.
On exit, x(i) is overwritten by a(i), for i = 1,...,n.

INCX

INCX is INTEGER
The increment between elements of X. INCX > 0.

Y

Y is REAL array,
dimension (1+(N-1)*INCY)
On entry, the vector y.
On exit, the sines of the plane rotations.

INCY

INCY is INTEGER
The increment between elements of Y. INCY > 0.

C

C is REAL array, dimension (1+(N-1)*INCC)
The cosines of the plane rotations.

INCC

INCC is INTEGER
The increment between elements of C. INCC > 0.

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

subroutine slarrv (integer N, real VL, real VU, real, dimension(*) D, real, dimension(*) L, real PIVMIN, integer, dimension(*) ISPLIT, integer M, integer DOL, integer DOU, real MINRGP, real RTOL1, real RTOL2, real, dimension(*) W, real, dimension(*) WERR, real, dimension(*) WGAP, integer, dimension(*) IBLOCK, integer, dimension(*) INDEXW, real, dimension(*) GERS, real, dimension(ldz, *) Z, integer LDZ, integer, dimension(*) ISUPPZ, real, dimension(*) WORK, integer, dimension(*) IWORK, integer INFO)

SLARRV computes the eigenvectors of the tridiagonal matrix T = L D LT given L, D and the eigenvalues of L D LT.

Purpose:

SLARRV computes the eigenvectors of the tridiagonal matrix
T = L D L**T given L, D and APPROXIMATIONS to the eigenvalues of L D L**T.
The input eigenvalues should have been computed by SLARRE.

Parameters

N

N is INTEGER
The order of the matrix. N >= 0.

VL

VL is REAL
Lower bound of the interval that contains the desired
eigenvalues. VL < VU. Needed to compute gaps on the left or right
end of the extremal eigenvalues in the desired RANGE.

VU

VU is REAL
Upper bound of the interval that contains the desired

eigenvalues. VL < VU.
Note: VU is currently not used by this implementation of SLARRV, VU is passed to SLARRV because it could be used compute gaps on the right end of the extremal eigenvalues. However, with not much initial accuracy in LAMBDA and VU, the formula can lead to an overestimation of the right gap and thus to inadequately early RQI 'convergence'. This is currently prevented this by forcing a small right gap. And so it turns out that VU is currently not used by this implementation of SLARRV.

D

D is REAL array, dimension (N)
On entry, the N diagonal elements of the diagonal matrix D.
On exit, D may be overwritten.

L

L is REAL array, dimension (N)
On entry, the (N-1) subdiagonal elements of the unit bidiagonal matrix L are in elements 1 to N-1 of L (if the matrix is not split.) At the end of each block is stored the corresponding shift as given by SLARRE.
On exit, L is overwritten.

PIVMIN

PIVMIN is REAL
The minimum pivot allowed in the Sturm sequence.

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.

M

M is INTEGER
The total number of input eigenvalues. 0 <= M <= N.

DOL

DOL is INTEGER

DOU

DOU is INTEGER
If the user wants to compute only selected eigenvectors from all the eigenvalues supplied, he can specify an index range DOL:DOU.
Or else the setting DOL=1, DOU=M should be applied.
Note that DOL and DOU refer to the order in which the eigenvalues are stored in W.
If the user wants to compute only selected eigenpairs, then the columns DOL-1 to DOU+1 of the eigenvector space Z contain the computed eigenvectors. All other columns of Z are set to zero.

MINRGP

MINRGP is REAL

RTOL1

RTOL1 is REAL

RTOL2

RTOL2 is REAL
Parameters for bisection.
An interval [LEFT,RIGHT] has converged if

RIGHT-LEFT < MAX(RTOL1*GAP, RTOL2*MAX(|LEFT|,|RIGHT|))

W

W is REAL array, dimension (N)
The first M elements of W contain the APPROXIMATE eigenvalues for which eigenvectors are to be computed. The eigenvalues should be grouped by split-off block and ordered from smallest to largest within the block (The output array W from SLARRE is expected here). Furthermore, they are with respect to the shift of the corresponding root representation for their block. On exit, W holds the eigenvalues of the UNshifted matrix.

WERR

WERR is REAL array, dimension (N)
The first M elements contain the semiwidth of the uncertainty interval of the corresponding eigenvalue in W

WGAP

WGAP is REAL array, dimension (N)
The separation from the right neighbor eigenvalue in W.

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 the second block.

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)). The Gerschgorin intervals should be computed from the original UNshifted matrix.

Z

Z is REAL array, dimension (LDZ, max(1,M))
If INFO = 0, the first M columns of Z contain the orthonormal eigenvectors of the matrix T corresponding to the input eigenvalues, with the i-th column of Z holding the eigenvector associated with W(i).
Note: the user must ensure that at least max(1,M) columns are supplied in the array Z.

LDZ

LDZ is INTEGER
The leading dimension of the array Z. LDZ >= 1, and if JOBZ = 'V', LDZ >= max(1,N).

ISUPPZ

ISUPPZ is INTEGER array, dimension (2*max(1,M))
The support of the eigenvectors in Z, i.e., the indices indicating the nonzero elements in Z. The I-th eigenvector is nonzero only in elements ISUPPZ(2*I-1) through

ISUPPZ(2*I).

WORK

WORK is REAL array, dimension (12*N)

IWORK

IWORK is INTEGER array, dimension (7*N)

INFO

```
INFO is INTEGER
= 0:  successful exit

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

=-1:  Problem in SLARRB when refining a child's eigenvalues.
=-2:  Problem in SLARRF when computing the RRR of a child.
      When a child is inside a tight cluster, it can be difficult
      to find an RRR. A partial remedy from the user's point of
      view is to make the parameter MINRGP smaller and recompile.
      However, as the orthogonality of the computed vectors is
      proportional to 1/MINRGP, the user should be aware that
      he might be trading in precision when he decreases MINRGP.
=-3:  Problem in SLARRB when refining a single eigenvalue
      after the Rayleigh correction was rejected.
= 5:  The Rayleigh Quotient Iteration failed to converge to
      full accuracy in MAXITR steps.
```

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 slartv (integer N, real, dimension(*) X, integer INCX, real, dimension(*) Y, integer INCY, real, dimension(*) C, real, dimension(*) S, integer INCC)

SLARTV applies a vector of plane rotations with real cosines and real sines to the elements of a pair of vectors.

Purpose:

```
SLARTV applies a vector of real plane rotations to elements of the
real vectors x and y. For i = 1,2,...,n

   ( x(i) ) := (  c(i)  s(i) ) ( x(i) )
   ( y(i) )    ( -s(i)  c(i) ) ( y(i) )
```

Parameters

N

N is INTEGER
The number of plane rotations to be applied.

X

X is REAL array,
dimension (1+(N-1)*INCX)
The vector x.

INCX

INCX is INTEGER
The increment between elements of X. INCX > 0.

Y

Y is REAL array,
dimension (1+(N-1)*INCY)
The vector y.

INCY

INCY is INTEGER
The increment between elements of Y. INCY > 0.

C

C is REAL array, dimension (1+(N-1)*INCC)
The cosines of the plane rotations.

S

S is REAL array, dimension (1+(N-1)*INCC)
The sines of the plane rotations.

INCC

INCC is INTEGER
The increment between elements of C and S. INCC > 0.

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

subroutine slaswp (integer N, real, dimension(lda, *) A, integer LDA, integer K1, integer K2, integer, dimension(*) IPIV, integer INCX)

SLASWP performs a series of row interchanges on a general rectangular matrix.

Purpose:

SLASWP performs a series of row interchanges on the matrix A.
One row interchange is initiated for each of rows K1 through K2 of A.

Parameters

N

N is INTEGER
The number of columns of the matrix A.

A

A is REAL array, dimension (LDA,N)
On entry, the matrix of column dimension N to which the row
interchanges will be applied.
On exit, the permuted matrix.

LDA

LDA is INTEGER
The leading dimension of the array A.

K1

K1 is INTEGER
The first element of IPIV for which a row interchange will
be done.

K2

K2 is INTEGER
(K2-K1+1) is the number of elements of IPIV for which a row
interchange will be done.

IPIV

IPIV is INTEGER array, dimension (K1+(K2-K1)*abs(INCX))
The vector of pivot indices. Only the elements in positions
K1 through K1+(K2-K1)*abs(INCX) of IPIV are accessed.
IPIV(K1+(K-K1)*abs(INCX)) = L implies rows K and L are to be
interchanged.

INCX

INCX is INTEGER
The increment between successive values of IPIV. If INCX
is negative, the pivots are applied in reverse order.

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

June 2017

Further Details:

Modified by
R. C. Whaley, Computer Science Dept., Univ. of Tenn., Knoxville, USA

subroutine slatbs (character UPLO, character TRANS, character DIAG, character NORMIN, integer N, integer KD, real, dimension(ldab, *) AB, integer LDAB, real, dimension(*) X, real SCALE, real, dimension(*) CNORM, integer INFO)

SLATBS solves a triangular banded system of equations.

Purpose:

SLATBS solves one of the triangular systems

A *x = s*b or A**T*x = s*b

with scaling to prevent overflow, where A is an upper or lower
triangular band matrix. Here A**T denotes the transpose of A, x and b
are n-element vectors, and s is a scaling factor, usually less than
or equal to 1, chosen so that the components of x will be less than
the overflow threshold. If the unscaled problem will not cause
overflow, the Level 2 BLAS routine STBSV is called. If the matrix A
is singular (A(j,j) = 0 for some j), then s is set to 0 and a
non-trivial solution to A*x = 0 is returned.

Parameters

UPLO

UPLO is CHARACTER*1
Specifies whether the matrix A is upper or lower triangular.
= 'U': Upper triangular
= 'L': Lower triangular

TRANS

TRANS is CHARACTER*1
Specifies the operation applied to A.
= 'N': Solve A * x = s*b (No transpose)
= 'T': Solve A**T* x = s*b (Transpose)
= 'C': Solve A**T* x = s*b (Conjugate transpose = Transpose)

DIAG

DIAG is CHARACTER*1
Specifies whether or not the matrix A is unit triangular.
= 'N': Non-unit triangular
= 'U': Unit triangular

NORMIN

NORMIN is CHARACTER*1
Specifies whether CNORM has been set or not.
= 'Y': CNORM contains the column norms on entry
= 'N': CNORM is not set on entry. On exit, the norms will
be computed and stored in CNORM.

N

N is INTEGER
The order of the matrix A. N >= 0.

KD

KD is INTEGER
The number of subdiagonals or superdiagonals in the
triangular matrix A. KD >= 0.

AB

AB is REAL array, dimension (LDAB,N)
The upper or lower triangular band matrix A, stored in the
first KD+1 rows of the array. The j-th column of A is stored
in the j-th column of the array AB as follows:
if UPLO = 'U', AB(kd+1+i-j,j) = A(i,j) for max(1,j-kd)<=i<=j;
if UPLO = 'L', AB(1+i-j,j) = A(i,j) for j<=i<=min(n,j+kd).

LDAB

LDAB is INTEGER
The leading dimension of the array AB. LDAB >= KD+1.

X

X is REAL array, dimension (N)
On entry, the right hand side b of the triangular system.
On exit, X is overwritten by the solution vector x.

SCALE

SCALE is REAL
The scaling factor s for the triangular system
A * x = s*b or A**T* x = s*b.
If SCALE = 0, the matrix A is singular or badly scaled, and
the vector x is an exact or approximate solution to A*x = 0.

CNORM

CNORM is REAL array, dimension (N)

If NORMIN = 'Y', CNORM is an input argument and CNORM(j) contains the norm of the off-diagonal part of the j-th column of A. If TRANS = 'N', CNORM(j) must be greater than or equal to the infinity-norm, and if TRANS = 'T' or 'C', CNORM(j) must be greater than or equal to the 1-norm.

If NORMIN = 'N', CNORM is an output argument and CNORM(j) returns the 1-norm of the offdiagonal part of the j-th column of A.

INFO

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

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

Further Details:

```
A rough bound on x is computed; if that is less than overflow, STBSV
is called, otherwise, specific code is used which checks for possible
overflow or divide-by-zero at every operation.

A columnwise scheme is used for solving A*x = b.  The basic algorithm
if A is lower triangular is

     x[1:n] := b[1:n]
     for j = 1, ..., n
          x(j) := x(j) / A(j,j)
          x[j+1:n] := x[j+1:n] - x(j) * A[j+1:n,j]
     end

Define bounds on the components of x after j iterations of the loop:
   M(j) = bound on x[1:j]
   G(j) = bound on x[j+1:n]
Initially, let M(0) = 0 and G(0) = max{x(i), i=1,...,n}.

Then for iteration j+1 we have
   M(j+1) <= G(j) / | A(j+1,j+1) |
   G(j+1) <= G(j) + M(j+1) * | A[j+2:n,j+1] |
        <= G(j) ( 1 + CNORM(j+1) / | A(j+1,j+1) | )

where CNORM(j+1) is greater than or equal to the infinity-norm of
column j+1 of A, not counting the diagonal.  Hence

   G(j) <= G(0) product ( 1 + CNORM(i) / | A(i,i) | )
              1<=i<=j
and

   |x(j)| <= ( G(0) / |A(j,j)| ) product ( 1 + CNORM(i) / |A(i,i)| )
```

1<=i< j

Since |x(j)| <= M(j), we use the Level 2 BLAS routine STBSV if the
reciprocal of the largest M(j), j=1,..,n, is larger than
max(underflow, 1/overflow).

The bound on x(j) is also used to determine when a step in the
columnwise method can be performed without fear of overflow. If
the computed bound is greater than a large constant, x is scaled to
prevent overflow, but if the bound overflows, x is set to 0, x(j) to
1, and scale to 0, and a non-trivial solution to A*x = 0 is found.

Similarly, a row-wise scheme is used to solve A**T*x = b. The basic
algorithm for A upper triangular is

```
   for j = 1, ..., n
        x(j) := ( b(j) - A[1:j-1,j]**T * x[1:j-1] ) / A(j,j)
   end
```

We simultaneously compute two bounds
   G(j) = bound on (b(i) - A[1:i-1,i]**T * x[1:i-1]), 1<=i<=j
   M(j) = bound on x(i), 1<=i<=j

The initial values are G(0) = 0, M(0) = max{b(i), i=1,..,n}, and we
add the constraint G(j) >= G(j-1) and M(j) >= M(j-1) for j >= 1.
Then the bound on x(j) is

```
   M(j) <= M(j-1) * ( 1 + CNORM(j) ) / | A(j,j) |

        <= M(0) * product ( ( 1 + CNORM(i) ) / |A(i,i)| )
                  1<=i<=j
```

and we can safely call STBSV if 1/M(n) and 1/G(n) are both greater
than max(underflow, 1/overflow).

subroutine slatdf (integer IJOB, integer N, real, dimension(ldz, *) Z, integer LDZ, real, dimension(*) RHS, real RDSUM, real RDSCAL, integer, dimension(*) IPIV, integer, dimension(*) JPIV)

SLATDF uses the LU factorization of the n-by-n matrix computed by sgetc2 and computes a contribution to the reciprocal Dif-estimate.

Purpose:

SLATDF uses the LU factorization of the n-by-n matrix Z computed by
SGETC2 and computes a contribution to the reciprocal Dif-estimate
by solving Z * x = b for x, and choosing the r.h.s. b such that
the norm of x is as large as possible. On entry RHS = b holds the
contribution from earlier solved sub-systems, and on return RHS = x.

The factorization of Z returned by SGETC2 has the form Z = P*L*U*Q,
where P and Q are permutation matrices. L is lower triangular with
unit diagonal elements and U is upper triangular.

Parameters

IJOB

IJOB is INTEGER
IJOB = 2: First compute an approximative null-vector e
   of Z using SGECON, e is normalized and solve for
   Zx = +-e - f with the sign giving the greater value
   of 2-norm(x). About 5 times as expensive as Default.
IJOB .ne. 2: Local look ahead strategy where all entries of

the r.h.s. b is chosen as either +1 or -1 (Default).

N

N is INTEGER
The number of columns of the matrix Z.

Z

Z is REAL array, dimension (LDZ, N)
On entry, the LU part of the factorization of the n-by-n
matrix Z computed by SGETC2: Z = P * L * U * Q

LDZ

LDZ is INTEGER
The leading dimension of the array Z. LDA >= max(1, N).

RHS

RHS is REAL array, dimension N.
On entry, RHS contains contributions from other subsystems.
On exit, RHS contains the solution of the subsystem with
entries according to the value of IJOB (see above).

RDSUM

RDSUM is REAL
On entry, the sum of squares of computed contributions to
the Dif-estimate under computation by STGSYL, where the
scaling factor RDSCAL (see below) has been factored out.
On exit, the corresponding sum of squares updated with the
contributions from the current sub-system.
If TRANS = 'T' RDSUM is not touched.
NOTE: RDSUM only makes sense when STGSY2 is called by STGSYL.

RDSCAL

RDSCAL is REAL
On entry, scaling factor used to prevent overflow in RDSUM.
On exit, RDSCAL is updated w.r.t. the current contributions
in RDSUM.
If TRANS = 'T', RDSCAL is not touched.
NOTE: RDSCAL only makes sense when STGSY2 is called by
STGSYL.

IPIV

IPIV is INTEGER array, dimension (N).
The pivot indices; for 1 <= i <= N, row i of the
matrix has been interchanged with row IPIV(i).

JPIV

JPIV is INTEGER array, dimension (N).
The pivot indices; for 1 <= j <= N, column j of the
matrix has been interchanged with column JPIV(j).

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

June 2016

Further Details:

This routine is a further developed implementation of algorithm BSOLVE in [1] using complete pivoting in the LU factorization.

Contributors:

Bo Kagstrom and Peter Poromaa, Department of Computing Science, Umea University, S-901 87 Umea, Sweden.

References:

[1] Bo Kagstrom and Lars Westin,
Generalized Schur Methods with Condition Estimators for
Solving the Generalized Sylvester Equation, IEEE Transactions
on Automatic Control, Vol. 34, No. 7, July 1989, pp 745-751.

[2] Peter Poromaa,
On Efficient and Robust Estimators for the Separation
between two Regular Matrix Pairs with Applications in
Condition Estimation. Report IMINF-95.05, Departement of
Computing Science, Umea University, S-901 87 Umea, Sweden, 1995.

subroutine slatps (character UPLO, character TRANS, character DIAG, character NORMIN, integer N, real, dimension(*) AP, real, dimension(*) X, real SCALE, real, dimension(*) CNORM, integer INFO)

SLATPS solves a triangular system of equations with the matrix held in packed storage.

Purpose:

SLATPS solves one of the triangular systems

A *x = s*b or A**T*x = s*b

with scaling to prevent overflow, where A is an upper or lower
triangular matrix stored in packed form. Here A**T denotes the
transpose of A, x and b are n-element vectors, and s is a scaling
factor, usually less than or equal to 1, chosen so that the
components of x will be less than the overflow threshold. If the
unscaled problem will not cause overflow, the Level 2 BLAS routine
STPSV is called. If the matrix A is singular (A(j,j) = 0 for some j),
then s is set to 0 and a non-trivial solution to A*x = 0 is returned.

Parameters

UPLO

UPLO is CHARACTER*1
Specifies whether the matrix A is upper or lower triangular.
= 'U': Upper triangular
= 'L': Lower triangular

TRANS

TRANS is CHARACTER*1
Specifies the operation applied to A.
= 'N': Solve A * x = s*b (No transpose)
= 'T': Solve A**T* x = s*b (Transpose)
= 'C': Solve A**T* x = s*b (Conjugate transpose = Transpose)

DIAG

DIAG is CHARACTER*1
Specifies whether or not the matrix A is unit triangular.
= 'N': Non-unit triangular
= 'U': Unit triangular

NORMIN

NORMIN is CHARACTER*1

Specifies whether CNORM has been set or not.
= 'Y': CNORM contains the column norms on entry
= 'N': CNORM is not set on entry. On exit, the norms will
be computed and stored in CNORM.

N

N is INTEGER
The order of the matrix A. N >= 0.

AP

AP is REAL array, dimension (N*(N+1)/2)
The upper or lower triangular matrix A, packed columnwise in
a linear array. The j-th column of A is stored in the array
AP as follows:
if UPLO = 'U', AP(i + (j-1)*j/2) = A(i,j) for 1<=i<=j;
if UPLO = 'L', AP(i + (j-1)*(2n-j)/2) = A(i,j) for j<=i<=n.

X

X is REAL array, dimension (N)
On entry, the right hand side b of the triangular system.
On exit, X is overwritten by the solution vector x.

SCALE

SCALE is REAL
The scaling factor s for the triangular system
A * x = s*b or A**T* x = s*b.
If SCALE = 0, the matrix A is singular or badly scaled, and
the vector x is an exact or approximate solution to A*x = 0.

CNORM

CNORM is REAL array, dimension (N)

If NORMIN = 'Y', CNORM is an input argument and CNORM(j)
contains the norm of the off-diagonal part of the j-th column
of A. If TRANS = 'N', CNORM(j) must be greater than or equal
to the infinity-norm, and if TRANS = 'T' or 'C', CNORM(j)
must be greater than or equal to the 1-norm.

If NORMIN = 'N', CNORM is an output argument and CNORM(j)
returns the 1-norm of the offdiagonal part of the j-th column
of A.

INFO

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

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

Further Details:

A rough bound on x is computed; if that is less than overflow, STPSV
is called, otherwise, specific code is used which checks for possible

overflow or divide-by-zero at every operation.

A columnwise scheme is used for solving A*x = b. The basic algorithm if A is lower triangular is

```
x[1:n] := b[1:n]
for j = 1, ..., n
    x(j) := x(j) / A(j,j)
    x[j+1:n] := x[j+1:n] - x(j) * A[j+1:n,j]
end
```

Define bounds on the components of x after j iterations of the loop:
  M(j) = bound on x[1:j]
  G(j) = bound on x[j+1:n]
Initially, let M(0) = 0 and G(0) = max{x(i), i=1,...,n}.

Then for iteration j+1 we have

```
M(j+1) <= G(j) / | A(j+1,j+1) |
G(j+1) <= G(j) + M(j+1) * | A[j+2:n,j+1] |
     <= G(j) ( 1 + CNORM(j+1) / | A(j+1,j+1) | )
```

where CNORM(j+1) is greater than or equal to the infinity-norm of column j+1 of A, not counting the diagonal. Hence

```
G(j) <= G(0) product ( 1 + CNORM(i) / | A(i,i) | )
          1<=i<=j
```

and

```
|x(j)| <= ( G(0) / |A(j,j)| ) product ( 1 + CNORM(i) / |A(i,i)| )
                          1<=i< j
```

Since |x(j)| <= M(j), we use the Level 2 BLAS routine STPSV if the reciprocal of the largest M(j), j=1,..,n, is larger than max(underflow, 1/overflow).

The bound on x(j) is also used to determine when a step in the columnwise method can be performed without fear of overflow. If the computed bound is greater than a large constant, x is scaled to prevent overflow, but if the bound overflows, x is set to 0, x(j) to 1, and scale to 0, and a non-trivial solution to A*x = 0 is found.

Similarly, a row-wise scheme is used to solve A**T*x = b. The basic algorithm for A upper triangular is

```
for j = 1, ..., n
    x(j) := ( b(j) - A[1:j-1,j]**T * x[1:j-1] ) / A(j,j)
end
```

We simultaneously compute two bounds
  G(j) = bound on (b(i) - A[1:i-1,i]**T * x[1:i-1]), 1<=i<=j
  M(j) = bound on x(i), 1<=i<=j

The initial values are G(0) = 0, M(0) = max{b(i), i=1,..,n}, and we add the constraint G(j) >= G(j-1) and M(j) >= M(j-1) for j >= 1. Then the bound on x(j) is

```
M(j) <= M(j-1) * ( 1 + CNORM(j) ) / | A(j,j) |

    <= M(0) * product ( ( 1 + CNORM(i) ) / |A(i,i)| )
```

```
            1<=i<=j

 and we can safely call STPSV if 1/M(n) and 1/G(n) are both greater
 than max(underflow, 1/overflow).
```

subroutine slatrs (character UPLO, character TRANS, character DIAG, character NORMIN, integer N, real, dimension(lda, *) A, integer LDA, real, dimension(*) X, real SCALE, real, dimension(*) CNORM, integer INFO)

SLATRS solves a triangular system of equations with the scale factor set to prevent overflow.

Purpose:

```
SLATRS solves one of the triangular systems

   A *x = s*b  or  A**T*x = s*b

with scaling to prevent overflow.  Here A is an upper or lower
triangular matrix, A**T denotes the transpose of A, x and b are
n-element vectors, and s is a scaling factor, usually less than
or equal to 1, chosen so that the components of x will be less than
the overflow threshold.  If the unscaled problem will not cause
overflow, the Level 2 BLAS routine STRSV is called.  If the matrix A
is singular (A(j,j) = 0 for some j), then s is set to 0 and a
non-trivial solution to A*x = 0 is returned.
```

Parameters

UPLO

```
UPLO is CHARACTER*1
Specifies whether the matrix A is upper or lower triangular.
= 'U':  Upper triangular
= 'L':  Lower triangular
```

TRANS

```
TRANS is CHARACTER*1
Specifies the operation applied to A.
= 'N':  Solve A * x = s*b  (No transpose)
= 'T':  Solve A**T* x = s*b  (Transpose)
= 'C':  Solve A**T* x = s*b  (Conjugate transpose = Transpose)
```

DIAG

```
DIAG is CHARACTER*1
Specifies whether or not the matrix A is unit triangular.
= 'N':  Non-unit triangular
= 'U':  Unit triangular
```

NORMIN

```
NORMIN is CHARACTER*1
Specifies whether CNORM has been set or not.
= 'Y':  CNORM contains the column norms on entry
= 'N':  CNORM is not set on entry.  On exit, the norms will
        be computed and stored in CNORM.
```

N

```
N is INTEGER
The order of the matrix A.  N >= 0.
```

A

```
A is REAL array, dimension (LDA,N)
The triangular matrix A.  If UPLO = 'U', the leading n by n
upper triangular part of the array A contains the upper
triangular matrix, and the strictly lower triangular part of
```

A is not referenced. If UPLO = 'L', the leading n by n lower triangular part of the array A contains the lower triangular matrix, and the strictly upper triangular part of A is not referenced. If DIAG = 'U', the diagonal elements of A are also not referenced and are assumed to be 1.

LDA

LDA is INTEGER
The leading dimension of the array A. LDA >= max (1,N).

X

X is REAL array, dimension (N)
On entry, the right hand side b of the triangular system.
On exit, X is overwritten by the solution vector x.

SCALE

SCALE is REAL
The scaling factor s for the triangular system
A * x = s*b or A**T* x = s*b.
If SCALE = 0, the matrix A is singular or badly scaled, and the vector x is an exact or approximate solution to A*x = 0.

CNORM

CNORM is REAL array, dimension (N)

If NORMIN = 'Y', CNORM is an input argument and CNORM(j) contains the norm of the off-diagonal part of the j-th column of A. If TRANS = 'N', CNORM(j) must be greater than or equal to the infinity-norm, and if TRANS = 'T' or 'C', CNORM(j) must be greater than or equal to the 1-norm.

If NORMIN = 'N', CNORM is an output argument and CNORM(j) returns the 1-norm of the offdiagonal part of the j-th column of A.

INFO

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

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

Further Details:

A rough bound on x is computed; if that is less than overflow, STRSV is called, otherwise, specific code is used which checks for possible overflow or divide-by-zero at every operation.

A columnwise scheme is used for solving A*x = b. The basic algorithm if A is lower triangular is

```
x[1:n] := b[1:n]
for j = 1, ..., n
```

```
        x(j) := x(j) / A(j,j)
        x[j+1:n] := x[j+1:n] - x(j) * A[j+1:n,j]
    end
```

Define bounds on the components of x after j iterations of the loop:

```
   M(j) = bound on x[1:j]
   G(j) = bound on x[j+1:n]
```

Initially, let M(0) = 0 and G(0) = max{x(i), i=1,...,n}.

Then for iteration j+1 we have

```
   M(j+1) <= G(j) / | A(j+1,j+1) |
   G(j+1) <= G(j) + M(j+1) * | A[j+2:n,j+1] |
          <= G(j) ( 1 + CNORM(j+1) / | A(j+1,j+1) | )
```

where CNORM(j+1) is greater than or equal to the infinity-norm of column j+1 of A, not counting the diagonal. Hence

```
   G(j) <= G(0) product ( 1 + CNORM(i) / | A(i,i) | )
                1<=i<=j
```

and

```
   |x(j)| <= ( G(0) / |A(j,j)| ) product ( 1 + CNORM(i) / |A(i,i)| )
                                 1<=i< j
```

Since |x(j)| <= M(j), we use the Level 2 BLAS routine STRSV if the reciprocal of the largest M(j), j=1,..,n, is larger than max(underflow, 1/overflow).

The bound on x(j) is also used to determine when a step in the columnwise method can be performed without fear of overflow. If the computed bound is greater than a large constant, x is scaled to prevent overflow, but if the bound overflows, x is set to 0, x(j) to 1, and scale to 0, and a non-trivial solution to A*x = 0 is found.

Similarly, a row-wise scheme is used to solve A**T*x = b. The basic algorithm for A upper triangular is

```
    for j = 1, ..., n
        x(j) := ( b(j) - A[1:j-1,j]**T * x[1:j-1] ) / A(j,j)
    end
```

We simultaneously compute two bounds

```
    G(j) = bound on ( b(i) - A[1:i-1,i]**T * x[1:i-1] ), 1<=i<=j
    M(j) = bound on x(i), 1<=i<=j
```

The initial values are G(0) = 0, M(0) = max{b(i), i=1,..,n}, and we add the constraint G(j) >= G(j-1) and M(j) >= M(j-1) for j >= 1. Then the bound on x(j) is

```
    M(j) <= M(j-1) * ( 1 + CNORM(j) ) / | A(j,j) |

         <= M(0) * product ( ( 1 + CNORM(i) ) / |A(i,i)| )
                   1<=i<=j
```

and we can safely call STRSV if 1/M(n) and 1/G(n) are both greater than max(underflow, 1/overflow).

subroutine slauu2 (character UPLO, integer N, real, dimension(lda, *) A, integer LDA, integer INFO)

SLAUU2 computes the product UUH or LHL, where U and L are upper or lower triangular matrices (unblocked algorithm).

Purpose:

SLAUU2 computes the product U * U**T or L**T * L, where the triangular
factor U or L is stored in the upper or lower triangular part of
the array A.

If UPLO = 'U' or 'u' then the upper triangle of the result is stored,
overwriting the factor U in A.
If UPLO = 'L' or 'l' then the lower triangle of the result is stored,
overwriting the factor L in A.

This is the unblocked form of the algorithm, calling Level 2 BLAS.

Parameters

UPLO

UPLO is CHARACTER*1
Specifies whether the triangular factor stored in the array A
is upper or lower triangular:
= 'U': Upper triangular
= 'L': Lower triangular

N

N is INTEGER
The order of the triangular factor U or L. N >= 0.

A

A is REAL array, dimension (LDA,N)
On entry, the triangular factor U or L.
On exit, if UPLO = 'U', the upper triangle of A is
overwritten with the upper triangle of the product U * U**T;
if UPLO = 'L', the lower triangle of A is overwritten with
the lower triangle of the product L**T * L.

LDA

LDA is INTEGER
The leading dimension of the array A. LDA >= max(1,N).

INFO

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

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

subroutine slauum (character UPLO, integer N, real, dimension(lda, *) A, integer LDA, integer INFO)

SLAUUM computes the product UUH or LHL, where U and L are upper or lower triangular matrices (blocked algorithm).

Purpose:

SLAUUM computes the product U * U**T or L**T * L, where the triangular
factor U or L is stored in the upper or lower triangular part of
the array A.

If UPLO = 'U' or 'u' then the upper triangle of the result is stored,
overwriting the factor U in A.
If UPLO = 'L' or 'l' then the lower triangle of the result is stored,
overwriting the factor L in A.

This is the blocked form of the algorithm, calling Level 3 BLAS.

Parameters

UPLO

UPLO is CHARACTER*1
Specifies whether the triangular factor stored in the array A
is upper or lower triangular:
= 'U': Upper triangular
= 'L': Lower triangular

N

N is INTEGER
The order of the triangular factor U or L. N >= 0.

A

A is REAL array, dimension (LDA,N)
On entry, the triangular factor U or L.
On exit, if UPLO = 'U', the upper triangle of A is
overwritten with the upper triangle of the product U * U**T;
if UPLO = 'L', the lower triangle of A is overwritten with
the lower triangle of the product L**T * L.

LDA

LDA is INTEGER
The leading dimension of the array A. LDA >= max(1,N).

INFO

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

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

subroutine srscl (integer N, real SA, real, dimension(*) SX, integer INCX)

SRSCL multiplies a vector by the reciprocal of a real scalar.

Purpose:

SRSCL multiplies an n-element real vector x by the real scalar 1/a.
This is done without overflow or underflow as long as
the final result x/a does not overflow or underflow.

Parameters

N

N is INTEGER
The number of components of the vector x.

SA

SA is REAL
The scalar a which is used to divide each component of x.
SA must be >= 0, or the subroutine will divide by zero.

SX

SX is REAL array, dimension
(1+(N-1)*abs(INCX))
The n-element vector x.

INCX

INCX is INTEGER
The increment between successive values of the vector SX.
> 0: SX(1) = X(1) and SX(1+(i-1)*INCX) = x(i), 1< i<= n

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

subroutine stprfb (character SIDE, character TRANS, character DIRECT, character STOREV, integer M, integer N, integer K, integer L, real, dimension(ldv, *) V, integer LDV, real, dimension(ldt, *) T, integer LDT, real, dimension(lda, *) A, integer LDA, real, dimension(ldb, *) B, integer LDB, real, dimension(ldwork, *) WORK, integer LDWORK)

STPRFB applies a real or complex 'triangular-pentagonal' blocked reflector to a real or complex matrix, which is composed of two blocks.

Purpose:

STPRFB applies a real "triangular-pentagonal" block reflector H or its
conjugate transpose H^H to a real matrix C, which is composed of two
blocks A and B, either from the left or right.

Parameters

SIDE

SIDE is CHARACTER*1
= 'L': apply H or H^H from the Left
= 'R': apply H or H^H from the Right

TRANS

TRANS is CHARACTER*1
= 'N': apply H (No transpose)
= 'C': apply H^H (Conjugate transpose)

DIRECT

DIRECT is CHARACTER*1
Indicates how H is formed from a product of elementary
reflectors
= 'F': H = H(1) H(2) . . . H(k) (Forward)
= 'B': H = H(k) . . . H(2) H(1) (Backward)

STOREV

STOREV is CHARACTER*1
Indicates how the vectors which define the elementary
reflectors are stored:
= 'C': Columns
= 'R': Rows

M

M is INTEGER
The number of rows of the matrix B.
M >= 0.

N

N is INTEGER
The number of columns of the matrix B.
N >= 0.

K

K is INTEGER
The order of the matrix T, i.e. the number of elementary
reflectors whose product defines the block reflector.
K >= 0.

L

L is INTEGER
The order of the trapezoidal part of V.
K >= L >= 0. See Further Details.

V

V is REAL array, dimension
(LDV,K) if STOREV = 'C'
(LDV,M) if STOREV = 'R' and SIDE = 'L'
(LDV,N) if STOREV = 'R' and SIDE = 'R'
The pentagonal matrix V, which contains the elementary reflectors
H(1), H(2), ..., H(K). See Further Details.

LDV

LDV is INTEGER
The leading dimension of the array V.
If STOREV = 'C' and SIDE = 'L', LDV >= max(1,M);
if STOREV = 'C' and SIDE = 'R', LDV >= max(1,N);
if STOREV = 'R', LDV >= K.

T

T is REAL array, dimension (LDT,K)
The triangular K-by-K matrix T in the representation of the
block reflector.

LDT

LDT is INTEGER
The leading dimension of the array T.
LDT >= K.

A

A is REAL array, dimension
(LDA,N) if SIDE = 'L' or (LDA,K) if SIDE = 'R'
On entry, the K-by-N or M-by-K matrix A.
On exit, A is overwritten by the corresponding block of
H*C or H^H*C or C*H or C*H^H. See Further Details.

LDA

LDA is INTEGER

The leading dimension of the array A.
If SIDE = 'L', LDA >= max(1,K);
If SIDE = 'R', LDA >= max(1,M).

B

B is REAL array, dimension (LDB,N)
On entry, the M-by-N matrix B.
On exit, B is overwritten by the corresponding block of
H*C or H^H*C or C*H or C*H^H. See Further Details.

LDB

LDB is INTEGER
The leading dimension of the array B.
LDB >= max(1,M).

WORK

WORK is REAL array, dimension
(LDWORK,N) if SIDE = 'L',
(LDWORK,K) if SIDE = 'R'.

LDWORK

LDWORK is INTEGER
The leading dimension of the array WORK.
If SIDE = 'L', LDWORK >= K;
if SIDE = 'R', LDWORK >= M.

Author

Univ. of Tennessee

Univ. of California Berkeley

Univ. of Colorado Denver

NAG Ltd.

Date

December 2016

Further Details:

```
The matrix C is a composite matrix formed from blocks A and B.
The block B is of size M-by-N; if SIDE = 'R', A is of size M-by-K,
and if SIDE = 'L', A is of size K-by-N.

If SIDE = 'R' and DIRECT = 'F', C = [A B].

If SIDE = 'L' and DIRECT = 'F', C = [A]
                                    [B].

If SIDE = 'R' and DIRECT = 'B', C = [B A].

If SIDE = 'L' and DIRECT = 'B', C = [B]
                                    [A].

The pentagonal matrix V is composed of a rectangular block V1 and a
trapezoidal block V2.  The size of the trapezoidal block is determined by
the parameter L, where 0<=L<=K.  If L=K, the V2 block of V is triangular;
if L=0, there is no trapezoidal block, thus V = V1 is rectangular.

If DIRECT = 'F' and STOREV = 'C':  V = [V1]
                                       [V2]
  - V2 is upper trapezoidal (first L rows of K-by-K upper triangular)
```

If DIRECT = 'F' and STOREV = 'R': V = [V1 V2]

- V2 is lower trapezoidal (first L columns of K-by-K lower triangular)

If DIRECT = 'B' and STOREV = 'C': V = [V2]
[V1]
- V2 is lower trapezoidal (last L rows of K-by-K lower triangular)

If DIRECT = 'B' and STOREV = 'R': V = [V2 V1]

- V2 is upper trapezoidal (last L columns of K-by-K upper triangular)

If STOREV = 'C' and SIDE = 'L', V is M-by-K with V2 L-by-K.

If STOREV = 'C' and SIDE = 'R', V is N-by-K with V2 L-by-K.

If STOREV = 'R' and SIDE = 'L', V is K-by-M with V2 K-by-L.

If STOREV = 'R' and SIDE = 'R', V is K-by-N with V2 K-by-L.

Author

Generated automatically by Doxygen for LAPACK from the source code.