NAG Toolbox

nag_lapack_zuncsd (f08rn)

1 Purpose

nag_lapack_zuncsd (f08rn) computes the CS decomposition of a complex m by m unitary matrix X, partitioned into a 2 by 2 array of submatrices.

2 Syntax

```
[x11, x12, x21, x22, theta, u1, u2, v1t, v2t, info] = nag_lapack_zuncsd(m, p, q, x11, x12, x21, x22, 'jobu1', jobu1, 'jobu2', jobu2, 'jobv1t', jobv1t, 'jobv2t', jobv2t, 'trans', trans, 'signs', signs)
[x11, x12, x21, x22, theta, u1, u2, v1t, v2t, info] = f08rn(m, p, q, x11, x12, x21, x22, 'jobu1', jobu1, 'jobu2', jobu2, 'jobv1t', jobv1t, 'jobv2t', jobv2t, 'trans', trans, 'signs', signs)
```

3 Description

The m by m unitary matrix X is partitioned as

$$X = \begin{pmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{pmatrix}$$

where X_{11} is a p by q submatrix and the dimensions of the other submatrices X_{12}, X_{21} and X_{22} are such that X remains m by m.

The CS decomposition of X is $X = U\Sigma_p V^{\mathrm{T}}$ where U, V and Σ_p are m by m matrices, such that

$$U = \begin{pmatrix} U_1 & \mathbf{0} \\ \mathbf{0} & U_2 \end{pmatrix}$$

is a unitary matrix containing the p by p unitary matrix U_1 and the $(m-p)$ by $(m-p)$ unitary matrix U_2;

$$V = \begin{pmatrix} V_1 & \mathbf{0} \\ \mathbf{0} & V_2 \end{pmatrix}$$

is a unitary matrix containing the q by q unitary matrix V_1 and the $(m-q)$ by $(m-q)$ unitary matrix V_2; and

$$\Sigma_p = \left(\begin{array}{ccc|ccc} I_{11} & & \mathbf{0} & & \mathbf{0} & \mathbf{0} \\ & C & \mathbf{0} & \mathbf{0} & -S & \\ \mathbf{0} & \mathbf{0} & & \mathbf{0} & & -I_{12} \\ \hline & \mathbf{0} & \mathbf{0} & I_{22} & & \mathbf{0} \\ \mathbf{0} & S & & & C & \mathbf{0} \\ \mathbf{0} & & I_{21} & \mathbf{0} & \mathbf{0} & \end{array}\right)$$

contains the r by r non-negative diagonal submatrices C and S satisfying $C^2 + S^2 = I$, where $r = \min(p, m-p, q, m-q)$ and the top left partition is p by q.

The identity matrix I_{11} is of order $\min(p,q) - r$ and vanishes if $\min(p,q) = r$.

The identity matrix I_{12} is of order $\min(p, m-q) - r$ and vanishes if $\min(p, m-q) = r$.

The identity matrix I_{21} is of order $\min(m-p, q) - r$ and vanishes if $\min(m-p, q) = r$.

The identity matrix I_{22} is of order $\min(m-p, m-q) - r$ and vanishes if $\min(m-p, m-q) = r$.

In each of the four cases $r = p, q, m-p, m-q$ at least two of the identity matrices vanish.

The indicated zeros represent augmentations by additional rows or columns (but not both) to the square diagonal matrices formed by I_{ij} and C or S.

Σ_p does not need to be stored in full; it is sufficient to return only the values θ_i for $i = 1, 2, \ldots, r$ where $C_{ii} = \cos(\theta_i)$ and $S_{ii} = \sin(\theta_i)$.

The algorithm used to perform the complete CS decomposition is described fully in Sutton (2009) including discussions of the stability and accuracy of the algorithm.

4 References

Anderson E, Bai Z, Bischof C, Blackford S, Demmel J, Dongarra J J, Du Croz J J, Greenbaum A, Hammarling S, McKenney A and Sorensen D (1999) *LAPACK Users' Guide* (3rd Edition) SIAM, Philadelphia http://www.netlib.org/lapack/lug

Golub G H and Van Loan C F (2012) *Matrix Computations* (4th Edition) Johns Hopkins University Press, Baltimore

Sutton B D (2009) Computing the complete CS decomposition *Numerical Algorithms (Volume 50)* **1017–1398** Springer US 33–65 http://dx.doi.org/10.1007/s11075-008-9215-6

5 Parameters

5.1 Compulsory Input Parameters

1: **m** – INTEGER

m, the number of rows and columns in the unitary matrix X.

Constraint: $\mathbf{m} \geq 0$.

2: **p** – INTEGER

p, the number of rows in X_{11} and X_{12}.

Constraint: $0 \leq \mathbf{p} \leq \mathbf{m}$.

3: **q** – INTEGER

q, the number of columns in X_{11} and X_{21}.

Constraint: $0 \leq \mathbf{q} \leq \mathbf{m}$.

4: **x11**$(ldx11, :)$ – COMPLEX (KIND=nag_wp) array

The first dimension, *ldx11*, of the array **x11** must satisfy

if **trans** = 'T', $ldx11 \geq \max(1, \mathbf{q})$;
otherwise $ldx11 \geq \max(1, \mathbf{p})$.

The second dimension of the array **x11** must be at least $\max(1, \mathbf{p})$ if **trans** = 'T', and at least **q** otherwise.

The upper left partition of the unitary matrix X whose CSD is desired.

5: **x12**$(ldx12, :)$ – COMPLEX (KIND=nag_wp) array

The first dimension, *ldx12*, of the array **x12** must satisfy

if **trans** = 'T', $ldx12 \geq \max(1, \mathbf{m} - \mathbf{q})$;
otherwise $ldx12 \geq \max(1, \mathbf{p})$.

The second dimension of the array **x12** must be at least $\max(1, \mathbf{p})$ if **trans** = 'T', and at least **m** − **q** otherwise.

The upper right partition of the unitary matrix X whose CSD is desired.

6: **x21**$(ldx21,:)$ – COMPLEX (KIND=nag_wp) array

The first dimension, *ldx21*, of the array **x21** must satisfy

if **trans** = 'T', $ldx21 \geq \max(1, \mathbf{q})$;
otherwise $ldx21 \geq \max(1, \mathbf{m} - \mathbf{p})$.

The second dimension of the array **x21** must be at least $\max(1, \mathbf{m} - \mathbf{p})$ if **trans** = 'T', and at least **q** otherwise.

The lower left partition of the unitary matrix X whose CSD is desired.

7: **x22**$(ldx22,:)$ – COMPLEX (KIND=nag_wp) array

The first dimension, *ldx22*, of the array **x22** must satisfy

if **trans** = 'T', $ldx22 \geq \max(1, \mathbf{m} - \mathbf{q})$;
otherwise $ldx22 \geq \max(1, \mathbf{m} - \mathbf{p})$.

The second dimension of the array **x22** must be at least $\max(1, \mathbf{m} - \mathbf{p})$ if **trans** = 'T', and at least $\mathbf{m} - \mathbf{q}$ otherwise.

The lower right partition of the unitary matrix X CSD is desired.

5.2 Optional Input Parameters

1: **jobu1** – CHARACTER(1)

Default: 'Y'

if **jobu1** = 'Y', U_1 is computed;

otherwise, U_1 is not computed.

2: **jobu2** – CHARACTER(1)

Default: 'Y'

if **jobu2** = 'Y', U_2 is computed;

otherwise, U_2 is not computed.

3: **jobv1t** – CHARACTER(1)

Default: 'Y'

if **jobv1t** = 'Y', V_1^{T} is computed;

otherwise, V_1^{T} is not computed.

4: **jobv2t** – CHARACTER(1)

Default: 'Y'

if **jobv2t** = 'Y', V_2^{T} is computed;

otherwise, V_2^{T} is not computed.

5: **trans** – CHARACTER(1)

Default: 'N'

if **trans** = 'T', X, U_1, U_2, V_1^{T} and V_2^{T} are stored in row-major order;

otherwise, X, U_1, U_2, V_1^{T} and V_2^{T} are stored in column-major order.

6: **signs** – CHARACTER(1)

Default: 'D'

if **signs** = 'O', the lower-left block is made nonpositive (the other convention);

otherwise, the upper-right block is made nonpositive (the default convention).

5.3 Output Parameters

1: **x11**($ldx11$, :) – COMPLEX (KIND=nag_wp) array

The first dimension, $ldx11$, of the array **x11** will be

if **trans** = 'T', $ldx11 = \max(1, \mathbf{q})$;
otherwise $ldx11 = \max(1, \mathbf{p})$.

The second dimension of the array **x11** will be $\max(1, \mathbf{p})$ if **trans** = 'T' and **q** otherwise.

Contains details of the unitary matrix used in a simultaneous bidiagonalization process.

2: **x12**($ldx12$, :) – COMPLEX (KIND=nag_wp) array

The first dimension, $ldx12$, of the array **x12** will be

if **trans** = 'T', $ldx12 = \max(1, \mathbf{m} - \mathbf{q})$;
otherwise $ldx12 = \max(1, \mathbf{p})$.

The second dimension of the array **x12** will be $\max(1, \mathbf{p})$ if **trans** = 'T' and $\mathbf{m} - \mathbf{q}$ otherwise.

Contains details of the unitary matrix used in a simultaneous bidiagonalization process.

3: **x21**($ldx21$, :) – COMPLEX (KIND=nag_wp) array

The first dimension, $ldx21$, of the array **x21** will be

if **trans** = 'T', $ldx21 = \max(1, \mathbf{q})$;
otherwise $ldx21 = \max(1, \mathbf{m} - \mathbf{p})$.

The second dimension of the array **x21** will be $\max(1, \mathbf{m} - \mathbf{p})$ if **trans** = 'T' and **q** otherwise.

Contains details of the unitary matrix used in a simultaneous bidiagonalization process.

4: **x22**($ldx22$, :) – COMPLEX (KIND=nag_wp) array

The first dimension, $ldx22$, of the array **x22** will be

if **trans** = 'T', $ldx22 = \max(1, \mathbf{m} - \mathbf{q})$;
otherwise $ldx22 = \max(1, \mathbf{m} - \mathbf{p})$.

The second dimension of the array **x22** will be $\max(1, \mathbf{m} - \mathbf{p})$ if **trans** = 'T' and $\mathbf{m} - \mathbf{q}$ otherwise.

Contains details of the unitary matrix used in a simultaneous bidiagonalization process.

5: **theta**($\min(\mathbf{p}, \mathbf{m} - \mathbf{p}, \mathbf{q}, \mathbf{m} - \mathbf{q})$) – REAL (KIND=nag_wp) array

The values θ_i for $i = 1, 2, \ldots, r$ where $r = \min(p, m - p, q, m - q)$. The diagonal submatrices C and S of Σ_p are constructed from these values as

$C = \mathrm{diag}(\cos(\mathbf{theta}(1)), \ldots, \cos(\mathbf{theta}(r)))$ and

$S = \mathrm{diag}(\sin(\mathbf{theta}(1)), \ldots, \sin(\mathbf{theta}(r)))$.

6: **u1**($ldu1$, :) – COMPLEX (KIND=nag_wp) array

The first dimension of the array **u1** will be if **jobu1** = 'Y', $ldu1 = \max(1, \mathbf{p})$.

The second dimension of the array **u1** will be $\max(1, \mathbf{p})$ if **jobu1** = 'Y' and 1 otherwise.

If **jobu1** = 'Y', **u1** contains the p by p unitary matrix U_1.

7: **u2**$(ldu2,:)$ – COMPLEX (KIND=nag_wp) array

The first dimension of the array **u2** will be if **jobu2** = 'Y', $ldu2 = \max(1, \mathbf{m} - \mathbf{p})$.

The second dimension of the array **u2** will be $\max(1, \mathbf{m} - \mathbf{p})$ if **jobu2** = 'Y' and 1 otherwise.

If **jobu2** = 'Y', **u2** contains the $m - p$ by $m - p$ unitary matrix U_2.

8: **v1t**$(ldv1t,:)$ – COMPLEX (KIND=nag_wp) array

The first dimension of the array **v1t** will be if **jobv1t** = 'Y', $ldv1t = \max(1, \mathbf{q})$.

The second dimension of the array **v1t** will be $\max(1, \mathbf{q})$ if **jobv1t** = 'Y' and 1 otherwise.

If **jobv1t** = 'Y', **v1t** contains the q by q unitary matrix ${V_1}^{\mathrm{H}}$.

9: **v2t**$(ldv2t,:)$ – COMPLEX (KIND=nag_wp) array

The first dimension of the array **v2t** will be if **jobv2t** = 'Y', $ldv2t = \max(1, \mathbf{m} - \mathbf{q})$.

The second dimension of the array **v2t** will be $\max(1, \mathbf{m} - \mathbf{q})$ if **jobv2t** = 'Y' and 1 otherwise.

If **jobv2t** = 'Y', **v2t** contains the $m - q$ by $m - q$ unitary matrix ${V_2}^{\mathrm{H}}$.

10: **info** – INTEGER

info = 0 unless the function detects an error (see Section 6).

6 Error Indicators and Warnings

info < 0

If **info** $= -i$, argument i had an illegal value. An explanatory message is output, and execution of the program is terminated.

info > 0

The Jacobi-type procedure failed to converge during an internal reduction to bidiagonal-block form. The process requires convergence to $\min(\mathbf{p}, \mathbf{m} - \mathbf{p}, \mathbf{q}, \mathbf{m} - \mathbf{q})$ values, the value of **info** gives the number of converged values.

7 Accuracy

The computed CS decomposition is nearly the exact CS decomposition for the nearby matrix $(X + E)$, where

$$\|E\|_2 = O(\epsilon),$$

and ϵ is the ***machine precision***.

8 Further Comments

The total number of floating-point operations required to perform the full CS decomposition is approximately $2m^3$.

The real analogue of this function is nag_lapack_dorcsd (f08ra).

9 Example

This example finds the full CS decomposition of a unitary 6 by 6 matrix X partitioned in 3 by 3 blocks.

The decomposition is performed both on submatrices of the unitary matrix X and on separated partition matrices. Code is also provided to perform a recombining check if required.

9.1 Program Text

```
     function f08rn_example

fprintf('f08rn example results\n\n');

% Find the full CS decomposition of X:
m = nag_int(6);
p = nag_int(2);
q = nag_int(4);

xr = [-1.3038E-02 -1.4039E-01  2.5177E-01 -5.0956E-02 -4.5947E-02 -5.2773E-02;
       4.2764E-01  8.6298E-02 -3.2188E-01  1.1979E-01 -8.0311E-02 -3.8117E-02;
      -3.2595E-01  3.8163E-01  1.3231E-01 -5.0671E-01  5.9714E-02 -1.3850E-01;
       1.5906E-01 -2.8207E-01  2.1598E-01 -4.0163E-01 -4.6443E-02 -3.7354E-01;
      -1.7210E-01 -5.0942E-01  3.6488E-02  1.9271E-01  5.7843E-01 -1.8815E-02;
      -2.6336E-01 -1.0861E-01  1.0906E-01 -8.8159E-02  1.5763E-02  6.5007E-01];
xi = [-3.2595E-01 -2.6167E-01 -7.9789E-01 -2.1750E-01  1.4052E-04 -2.2492E-01;
      -6.2582E-01 -3.8174E-02  1.6112E-01  1.6319E-01 -4.3605E-01 -2.1907E-01;
       1.6428E-01 -1.8219E-01 -1.4565E-02  1.8615E-01 -5.8974E-01 -9.0941E-02;
      -5.2151E-03  1.9732E-01  1.8813E-01  2.6787E-01  3.0864E-01 -5.5148E-01;
      -1.3038E-02 -5.0319E-01  2.0316E-01  1.5574E-01 -1.2439E-01 -5.5686E-02;
      -2.4772E-01  2.8474E-01 -1.2712E-01  5.6169E-01  4.7130E-02  4.9173E-03];
x = xr + i*xi;

% Partition matrix
x11 = x(1:p,  1:q);   x12 = x(1:p,  q+1:m);
x21 = x(p+1:m,1:q);   x22 = x(p+1:m,q+1:m);

[x11, x12, x21, x22, theta, u1, u2, v1t, v2t, info] = ...
  f08rn(...
        m, p, q, x11, x12, x21, x22);

disp('Components of CS factorization of X:');
disp('    Theta');
disp(theta);
disp('    U1');
disp(u1);
disp('    U2');
disp(u2);
disp('    V1');
disp(v1t');
disp('    V2');
disp(v2t');
```

9.2 Program Results

```
     f08rn example results

Components of CS factorization of X:
    Theta
    0.1973
    0.5386

    U1
  -0.2084 - 0.9384i   0.1565 + 0.2269i
  -0.0490 + 0.2713i   0.1972 + 0.9408i

    U2
   0.1599 - 0.2574i  -0.0334 - 0.6270i   0.0573 + 0.6677i   0.2378 + 0.0912i
   0.3473 + 0.6047i  -0.3155 - 0.0122i   0.0703 - 0.1039i   0.4497 + 0.4427i
   0.3373 - 0.0221i   0.5536 - 0.1556i   0.6869 - 0.2398i  -0.1173 + 0.1096i
  -0.5483 + 0.0839i   0.4188 + 0.0043i   0.0576 - 0.0505i   0.7043 - 0.1224i

    V1
   0.1202 + 0.0302i  -0.6762 + 0.6685i  -0.0150 - 0.1654i  -0.1297 + 0.1903i
   0.2654 + 0.1007i  -0.1168 + 0.1140i  -0.3900 + 0.7376i  -0.0721 - 0.4376i
   0.7707 - 0.4915i  -0.0624 - 0.1777i  -0.0175 - 0.0930i   0.3209 + 0.1306i
```

```
   0.2581 + 0.0438i   0.1396 + 0.1200i   0.1007 - 0.5071i  -0.3019 - 0.7342i

    V2
   0.5354 + 0.0000i   0.8446 + 0.0000i
  -0.8393 + 0.0941i   0.5321 - 0.0596i
```