NAG Library Routine Document

F08RNF (ZUNCSD)

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details.

1 Purpose

F08RNF (ZUNCSD) computes the CS decomposition of a complex m by m unitary matrix X, partitioned into a 2 by 2 array of submatrices.

2 Specification

```
SUBROUTINE F08RNF (JOBU1, JOBU2, JOBV1T, JOBV2T, TRANS, SIGNS, M, P, Q,     &
                   X11, LDX11, X12, LDX12, X21, LDX21, X22, LDX22,          &
                   THETA, U1, LDU1, U2, LDU2, V1T, LDV1T, V2T, LDV2T,       &
                   WORK, LWORK, RWORK, LRWORK, IWORK, INFO)

INTEGER                 M, P, Q, LDX11, LDX12, LDX21, LDX22, LDU1, LDU2,    &
                        LDV1T, LDV2T, LWORK, LRWORK,                        &
                        IWORK(M-min(P,M-P,Q,M-Q)), INFO
REAL (KIND=nag_wp)      THETA(min(P,M-P,Q,M-Q)), RWORK(max(1,LRWORK))
COMPLEX (KIND=nag_wp) X11(LDX11,*), X12(LDX12,*), X21(LDX21,*),             &
                        X22(LDX22,*), U1(LDU1,*), U2(LDU2,*),               &
                        V1T(LDV1T,*), V2T(LDV2T,*), WORK(max(1,LWORK))
CHARACTER(1)            JOBU1, JOBU2, JOBV1T, JOBV2T, TRANS, SIGNS
```

The routine may be called by its LAPACK name ***zuncsd***.

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 Arguments

1: JOBU1 – CHARACTER(1) *Input*

On entry:

if JOBU1 = 'Y', U_1 is computed;

otherwise, U_1 is not computed.

2: JOBU2 – CHARACTER(1) *Input*

On entry:

if JOBU2 = 'Y', U_2 is computed;

otherwise, U_2 is not computed.

3: JOBV1T – CHARACTER(1) *Input*

On entry:

if JOBV1T = 'Y', V_1^{T} is computed;

otherwise, V_1^{T} is not computed.

4: JOBV2T – CHARACTER(1) *Input*

On entry:

if JOBV2T = 'Y', V_2^{T} is computed;

otherwise, V_2^{T} is not computed.

5: TRANS – CHARACTER(1) *Input*

On entry:

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) *Input*

On entry:

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

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

7: M – INTEGER *Input*

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

Constraint: $\mathrm{M} \geq 0$.

8: P – INTEGER *Input*

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

Constraint: $0 \leq \mathrm{P} \leq \mathrm{M}$.

9: Q – INTEGER *Input*

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

Constraint: $0 \leq \mathrm{Q} \leq \mathrm{M}$.

10: X11(LDX11, $*$) – COMPLEX (KIND=nag_wp) array *Input/Output*

Note: the second dimension of the array X11 must be at least $\max(1, \mathrm{P})$ if TRANS = 'T', and at least Q otherwise.

On entry: the upper left partition of the unitary matrix X whose CSD is desired.

On exit: contains details of the unitary matrix used in a simultaneous bidiagonalization process.

11: LDX11 – INTEGER *Input*

On entry: the first dimension of the array X11 as declared in the (sub)program from which F08RNF (ZUNCSD) is called.

Constraints:

if TRANS = 'T', $\mathrm{LDX11} \geq \max(1, \mathrm{Q})$;
otherwise $\mathrm{LDX11} \geq \max(1, \mathrm{P})$.

12: X12(LDX12, $*$) – COMPLEX (KIND=nag_wp) array *Input/Output*

Note: the second dimension of the array X12 must be at least $\max(1, \mathrm{P})$ if TRANS = 'T', and at least $\mathrm{M} - \mathrm{Q}$ otherwise.

On entry: the upper right partition of the unitary matrix X whose CSD is desired.

On exit: contains details of the unitary matrix used in a simultaneous bidiagonalization process.

13: LDX12 – INTEGER *Input*

On entry: the first dimension of the array X12 as declared in the (sub)program from which F08RNF (ZUNCSD) is called.

Constraints:

if TRANS = 'T', LDX12 $\geq \max(1, \mathrm{M} - \mathrm{Q})$;
otherwise LDX12 $\geq \max(1, \mathrm{P})$.

14: X21(LDX21, $*$) – COMPLEX (KIND=nag_wp) array *Input/Output*

Note: the second dimension of the array X21 must be at least $\max(1, \mathrm{M} - \mathrm{P})$ if TRANS = 'T', and at least Q otherwise.

On entry: the lower left partition of the unitary matrix X whose CSD is desired.

On exit: contains details of the unitary matrix used in a simultaneous bidiagonalization process.

15: LDX21 – INTEGER *Input*

On entry: the first dimension of the array X21 as declared in the (sub)program from which F08RNF (ZUNCSD) is called.

Constraints:

if TRANS = 'T', LDX21 $\geq \max(1, \mathrm{Q})$;
otherwise LDX21 $\geq \max(1, \mathrm{M} - \mathrm{P})$.

16: X22(LDX22, $*$) – COMPLEX (KIND=nag_wp) array *Input/Output*

Note: the second dimension of the array X22 must be at least $\max(1, \mathrm{M} - \mathrm{P})$ if TRANS = 'T', and at least $\mathrm{M} - \mathrm{Q}$ otherwise.

On entry: the lower right partition of the unitary matrix X CSD is desired.

On exit: contains details of the unitary matrix used in a simultaneous bidiagonalization process.

17: LDX22 – INTEGER *Input*

On entry: the first dimension of the array X22 as declared in the (sub)program from which F08RNF (ZUNCSD) is called.

Constraints:

if TRANS = 'T', LDX22 $\geq \max(1, \mathrm{M} - \mathrm{Q})$;
otherwise LDX22 $\geq \max(1, \mathrm{M} - \mathrm{P})$.

18: THETA($\min(\mathrm{P}, \mathrm{M} - \mathrm{P}, \mathrm{Q}, \mathrm{M} - \mathrm{Q})$) – REAL (KIND=nag_wp) array *Output*

On exit: 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

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

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

19: U1(LDU1, $*$) – COMPLEX (KIND=nag_wp) array *Output*

Note: the second dimension of the array U1 must be at least $\max(1, \mathrm{P})$ if JOBU1 = 'Y', and at least 1 otherwise.

On exit: if JOBU1 = 'Y', U1 contains the p by p unitary matrix U_1.

20: LDU1 – INTEGER *Input*

On entry: the first dimension of the array U1 as declared in the (sub)program from which F08RNF (ZUNCSD) is called.

Constraint: if JOBU1 = 'Y', LDU1 $\geq \max(1, \mathrm{P})$.

21: U2(LDU2, $*$) – COMPLEX (KIND=nag_wp) array *Output*

Note: the second dimension of the array U2 must be at least $\max(1, \mathrm{M} - \mathrm{P})$ if JOBU2 = 'Y', and at least 1 otherwise.

On exit: if JOBU2 = 'Y', U2 contains the $m - p$ by $m - p$ unitary matrix U_2.

22: LDU2 – INTEGER *Input*

On entry: the first dimension of the array U2 as declared in the (sub)program from which F08RNF (ZUNCSD) is called.

Constraint: if JOBU2 = 'Y', LDU2 $\geq \max(1, \mathrm{M} - \mathrm{P})$.

23: V1T(LDV1T, $*$) – COMPLEX (KIND=nag_wp) array *Output*

Note: the second dimension of the array V1T must be at least $\max(1, \mathrm{Q})$ if JOBV1T = 'Y', and at least 1 otherwise.

On exit: if JOBV1T = 'Y', V1T contains the q by q unitary matrix $V_1{}^{\mathrm{H}}$.

24: LDV1T – INTEGER *Input*

On entry: the first dimension of the array V1T as declared in the (sub)program from which F08RNF (ZUNCSD) is called.

Constraint: if JOBV1T = 'Y', LDV1T $\geq \max(1, \mathrm{Q})$.

25: V2T(LDV2T, $*$) – COMPLEX (KIND=nag_wp) array *Output*

Note: the second dimension of the array V2T must be at least $\max(1, \mathrm{M} - \mathrm{Q})$ if JOBV2T = 'Y', and at least 1 otherwise.

On exit: if JOBV2T = 'Y', V2T contains the $m - q$ by $m - q$ unitary matrix $V_2{}^{\mathrm{H}}$.

26: LDV2T – INTEGER *Input*

On entry: the first dimension of the array V2T as declared in the (sub)program from which F08RNF (ZUNCSD) is called.

Constraint: if JOBV2T = 'Y', LDV2T $\geq \max(1, \mathrm{M} - \mathrm{Q})$.

27: WORK($\max(1, \mathrm{LWORK})$) – COMPLEX (KIND=nag_wp) array *Workspace*

On exit: if INFO $= 0$, WORK(1) returns the optimal LWORK.

If INFO > 0 on exit, WORK($2 : r$) contains the values PHI (1), ... PHI ($r - 1$) that, together with THETA(1), ... THETA(r), define the matrix in intermediate bidiagonal-block form remaining after nonconvergence. INFO specifies the number of nonzero PHI's.

28: LWORK – INTEGER *Input*

On entry:

If LWORK $= -1$, a workspace query is assumed; the routine only calculates the optimal size of the WORK array, returns this value as the first entry of the WORK array, and no error message related to LWORK is issued.

The minimum workspace required is $\max(1, \mathrm{P}) + \max(1, \mathrm{M} - \mathrm{P}) + \max(1, \mathrm{Q}) + \max(1, \mathrm{M} - \mathrm{Q}) + \max(1, \mathrm{P}, \mathrm{M} - \mathrm{P}, \mathrm{Q}, \mathrm{M} - \mathrm{Q}) + 1$; the optimal amount of workspace depends on internal block sizes and the relative problem dimensions.

Constraint:
LWORK $= -1$ or LWORK $\geq \max(1, \mathrm{P}) + \max(1, \mathrm{M} - \mathrm{P}) + \max(1, \mathrm{Q}) + \max(1, \mathrm{M} - \mathrm{Q}) + \max(1, \mathrm{P}, \mathrm{M} - \mathrm{P}, \mathrm{Q}, \mathrm{M} - \mathrm{Q}) + 1$.

29: RWORK(max(1, LRWORK)) – REAL (KIND=nag_wp) array *Workspace*

30: LRWORK – INTEGER *Input*

On entry:

If LRWORK $= -1$, a workspace query is assumed; the routine only calculates the optimal size of the WORK array, returns this value as the first entry of the WORK array, and no error message related to LRWORK is issued. Otherwise the required workspace is $5 \times \max(1, \mathrm{Q} - 1) + 4 \times \max(1, \mathrm{Q}) + \max(1, 8 \times \mathrm{Q}) + 1$ which equates to 11 for $\mathrm{Q} = 0$, 18 for $\mathrm{Q} = 1$ and $17 \times \mathrm{Q} - 4$ when $\mathrm{Q} > 1$.

Constraint:
LRWORK $= -1$ or LRWORK $\geq 5 \times \max(1, \mathrm{Q} - 1) + 4 \times \max(1, \mathrm{Q}) + \max(1, 8 \times \mathrm{Q}) + 1$.

31: IWORK($\mathrm{M} - \min(\mathrm{P}, \mathrm{M} - \mathrm{P}, \mathrm{Q}, \mathrm{M} - \mathrm{Q})$) – INTEGER array *Workspace*

32: INFO – INTEGER *Output*

On exit: INFO $= 0$ unless the routine 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 $\min(\mathrm{P}, \mathrm{M} - \mathrm{P}, \mathrm{Q}, \mathrm{M} - \mathrm{Q})$ 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 Parallelism and Performance

F08RNF (ZUNCSD) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

F08RNF (ZUNCSD) makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 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 routine is F08RAF (DORCSD).

10 Example

This example finds the full CS decomposition of a unitary 6 by 6 matrix X (see Section 10.2) partitioned so that the top left block is 2 by 4.

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.

10.1 Program Text

```
    Program f08rnfe

!     F08RNF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: nag_wp, x04caf, x04dbf, zgemm, zuncsd
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Complex (Kind=nag_wp), Parameter :: one = (1.0_nag_wp,0.0_nag_wp)
      Complex (Kind=nag_wp), Parameter :: zero = (0.0_nag_wp,0.0_nag_wp)
      Integer, Parameter               :: nin = 5, nout = 6, recombine = 1,    &
                                          reprint = 1
!     .. Local Scalars ..
      Integer                          :: i, ifail, info, info2, j, ldu, ldu1, &
                                          ldu2, ldv, ldv1t, ldv2t, ldx, ldx11, &
                                          ldx12, ldx21, ldx22, lrwork, lwork,  &
                                          m, n11, n12, n21, n22, p, q, r
!     .. Local Arrays ..
      Complex (Kind=nag_wp), Allocatable :: u(:,:), u1(:,:), u2(:,:), v(:,:),  &
                                          v1t(:,:), v2t(:,:), w(:,:), work(:), &
                                          x(:,:), x11(:,:), x12(:,:),          &
                                          x21(:,:), x22(:,:)
      Complex (Kind=nag_wp)            :: wdum(1)
      Real (Kind=nag_wp)               :: rwdum(1)
      Real (Kind=nag_wp), Allocatable  :: rwork(:), theta(:)
      Integer, Allocatable             :: iwork(:)
      Character (1)                    :: clabs(1), rlabs(1)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: cmplx, cos, min, nint, real, sin
!     .. Executable Statements ..
      Write (nout,*) 'F08RNF Example Program Results'
      Write (nout,*)
      Flush (nout)
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) m, p, q

      r = min(min(p,q),min(m-p,m-q))

      ldx = m
      ldx11 = p
      ldx12 = p
      ldx21 = m - p
      ldx22 = m - p
      ldu = m
      ldu1 = p
      ldu2 = m - p
      ldv = m
      ldv1t = q
      ldv2t = m - q
      Allocate (x(ldx,m),u(ldu,m),v(ldv,m),theta(r),iwork(m),w(ldx,m))
      Allocate (x11(ldx11,q),x12(ldx12,m-q),x21(ldx21,q),x22(ldx22,m-q))
      Allocate (u1(ldu1,p),u2(ldu2,m-p),v1t(ldv1t,q),v2t(ldv2t,m-q))

!     Read (by column) and print unitary X from data file
!      (as, say, generated by a generalized singular value decomposition).
```

```
      Do i = 1, m
        Read (nin,*) x(1:m,i)
      End Do

!     Print general complex matrix using matrix printing routine x04dbf.
!     ifail: behaviour on error exit
!             =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
      ifail = 0
      Call x04dbf('General','N',m,m,x,ldx,'Bracketed','F7.4',                 &
        '    Unitary matrix X','Integer',rlabs,'Integer',clabs,80,0,ifail)
      Write (nout,*)

!     Compute the complete CS factorization of X:
!        X11 is stored in X(1:p,   1:q), X12 is stored in X(1:p,   q+1:m)
!        X21 is stored in X(p+1:m, 1:q), X22 is stored in X(p+1:m, q+1:m)
!        U1  is stored in U(1:p,   1:p), U2  is stored in U(p+1:m, p+1:m)
!        V1  is stored in V(1:q,   1:q), V2  is stored in V(q+1:m, q+1:m)
      x11(1:p,1:q) = x(1:p,1:q)
      x12(1:p,1:m-q) = x(1:p,q+1:m)
      x21(1:m-p,1:q) = x(p+1:m,1:q)
      x22(1:m-p,1:m-q) = x(p+1:m,q+1:m)

!     Workspace query first to get length of work array for complete CS
!     factorization routine zuncsd/f08rnf.
      lwork = -1
      lrwork = -1
      Call zuncsd('Yes U1','Yes U2','Yes V1T','Yes V2T','Column','Default',m,  &
        p,q,x,ldx,x(1,q+1),ldx,x(p+1,1),ldx,x(p+1,q+1),ldx,theta,u,ldu,        &
        u(p+1,p+1),ldu,v,ldv,v(q+1,q+1),ldv,wdum,lwork,rwdum,lrwork,iwork,     &
        info)
      lwork = nint(real(wdum(1)))
      lrwork = nint(rwdum(1))
      Allocate (work(lwork),rwork(lrwork))
!     Initialize all of  u, v to zero.
      u(1:m,1:m) = zero
      v(1:m,1:m) = zero

!     This is how you might pass partitions as sub-matrices
      Call zuncsd('Yes U1','Yes U2','Yes V1T','Yes V2T','Column','Default',m,  &
        p,q,x,ldx,x(1,q+1),ldx,x(p+1,1),ldx,x(p+1,q+1),ldx,theta,u,ldu,        &
        u(p+1,p+1),ldu,v,ldv,v(q+1,q+1),ldv,work,lwork,rwork,lrwork,iwork,     &
        info)
      If (info/=0) Then
        Write (nout,99999) 'Failure in ZUNCSD/F08RNF. info =', info
        Go To 100
      End If

!     Print Theta using real matrix printing routine x04caf
!     Note: U1, U2, V1T, V2T not printed since these may differ by a sign
!     change in columns of U1, U2 and corresponding rows of V1T, V2T.
      Write (nout,99998) 'Theta Component of CS factorization of X:'
      ifail = 0
      Call x04caf('G','N',r,1,theta,r,'    Theta',ifail)
      Write (nout,*)

!     And this is how you might pass partitions as separate matrices.
      Call zuncsd('Yes U1','Yes U2','Yes V1T','Yes V2T','Column','Default',m,  &
        p,q,x11,ldx11,x12,ldx12,x21,ldx21,x22,ldx22,theta,u1,ldu1,u2,ldu2,v1t, &
        ldv1t,v2t,ldv2t,work,lwork,rwork,lrwork,iwork,info2)
      If (info/=0) Then
        Write (nout,99999) 'Failure in ZUNCSD/F08RNF. info =', info
        Go To 100
      End If

!     Reprint Theta using matrix printing routine x04caf.
      If (reprint/=0) Then
        Write (nout,99998) 'Components of CS factorization of X:'
        ifail = 0
        Call x04caf('G','N',r,1,theta,r,'    Theta',ifail)
        Write (nout,*)
```

```
      End If

      If (recombine/=0) Then
!       Recombining should return the original matrix
!       Assemble Sigma_p into X
        x(1:m,1:m) = zero
        n11 = min(p,q) - r
        n12 = min(p,m-q) - r
        n21 = min(m-p,q) - r
        n22 = min(m-p,m-q) - r
!       Top Half
        Do j = 1, n11
          x(j,j) = one
        End Do
        Do j = 1, r
          x(j+n11,j+n11) = cmplx(cos(theta(j)),0.0_nag_wp,kind=nag_wp)
          x(j+n11,j+n11+r+n21+n22) = cmplx(-sin(theta(j)),0.0_nag_wp,          &
            kind=nag_wp)
        End Do
        Do j = 1, n12
          x(j+n11+r,j+n11+r+n21+n22+r) = -one
        End Do
!       Bottom half
        Do j = 1, n22
          x(p+j,q+j) = one
        End Do
        Do j = 1, r
          x(p+n22+j,j+n11) = cmplx(sin(theta(j)),0.0_nag_wp,kind=nag_wp)
          x(p+n22+j,j+r+n21+n22) = cmplx(cos(theta(j)),0.0_nag_wp,kind=nag_wp)
        End Do
        Do j = 1, n21
          x(p+n22+r+j,n11+r+j) = one
        End Do
!       multiply U * Sigma_p into w
        Call zgemm('n','n',m,m,m,one,u,ldu,x,ldx,zero,w,ldx)
!       form U * Sigma_p * V^T into u
        Call zgemm('n','n',m,m,m,one,w,ldx,v,ldv,zero,u,ldu)

!       Print recombined matrix using complex matrix printing routine x04dbf.
        Write (nout,*)
        ifail = 0
        Call x04dbf('General','N',m,m,u,ldu,'Bracketed','F7.4',                &
          '    Recombined matrix X = U * Sigma_p * V^H','Integer',rlabs,       &
          'Integer',clabs,80,0,ifail)
      End If
100   Continue

99999 Format (1X,A,I4)
99998 Format (/,1X,A,/)
    End Program f08rnfe
```

10.2 Program Data

```
F08RNF Example Program Data

   6        2       4        : m    p    q

 ( -1.3038E-02, -3.2597E-01)
 (  4.2764E-01, -6.2582E-01)
 ( -3.2597E-01,  1.6428E-01)
 (  1.5906E-01, -5.2151E-03)
 ( -1.7210E-01, -1.3027E-02)
 ( -2.6336E-01, -2.4772E-01) : column 1 of unitary matrix X

 ( -1.4039E-01, -2.6167E-01)
 (  8.6298E-02, -3.8174E-02)
 (  3.8163E-01, -1.8219E-01)
 ( -2.8207E-01,  1.9734E-01)
 ( -5.0942E-01, -5.0319E-01)
 ( -1.0861E-01,  2.8474E-01) : column 2 of unitary matrix X
```

```
(  2.5177E-01, -7.9789E-01)
( -3.2188E-01,  1.6112E-01)
(  1.3231E-01, -1.4563E-02)
(  2.1598E-01,  1.8813E-01)
(  3.6488E-02,  2.0319E-01)
(  1.0906E-01, -1.2712E-01) : column 3 of unitary matrix X

( -5.0956E-02, -2.1750E-01)
(  1.1979E-01,  1.6319E-01)
( -5.0671E-01,  1.8612E-01)
( -4.0163E-01,  2.6787E-01)
(  1.9271E-01,  1.5574E-01)
( -8.8179E-02,  5.6169E-01) : column 4 of unitary matrix X

( -4.5947E-02,  1.4052E-04)
( -8.0311E-02, -4.3611E-01)
(  5.9714E-02, -5.8970E-01)
( -4.6443E-02,  3.0864E-01)
(  5.7843E-01, -1.2439E-01)
(  1.5763E-02,  4.7130E-02) : column 5 of unitary matrix X

( -5.2773E-02, -2.2492E-01)
( -3.8117E-02, -2.1907E-01)
( -1.3850E-01, -9.0941E-02)
( -3.7354E-01, -5.5148E-01)
( -1.8818E-02, -5.5686E-02)
(  6.5007E-01,  4.9173E-03) : column 6 of unitary matrix X
```

10.3 Program Results

```
F08RNF Example Program Results

    Unitary matrix X
                   1                  2                  3                  4
1  (-0.0130,-0.3260) (-0.1404,-0.2617) ( 0.2518,-0.7979) (-0.0510,-0.2175)
2  ( 0.4276,-0.6258) ( 0.0863,-0.0382) (-0.3219, 0.1611) ( 0.1198, 0.1632)
3  (-0.3260, 0.1643) ( 0.3816,-0.1822) ( 0.1323,-0.0146) (-0.5067, 0.1861)
4  ( 0.1591,-0.0052) (-0.2821, 0.1973) ( 0.2160, 0.1881) (-0.4016, 0.2679)
5  (-0.1721,-0.0130) (-0.5094,-0.5032) ( 0.0365, 0.2032) ( 0.1927, 0.1557)
6  (-0.2634,-0.2477) (-0.1086, 0.2847) ( 0.1091,-0.1271) (-0.0882, 0.5617)

                   5                  6
1  (-0.0459, 0.0001) (-0.0528,-0.2249)
2  (-0.0803,-0.4361) (-0.0381,-0.2191)
3  ( 0.0597,-0.5897) (-0.1385,-0.0909)
4  (-0.0464, 0.3086) (-0.3735,-0.5515)
5  ( 0.5784,-0.1244) (-0.0188,-0.0557)
6  ( 0.0158, 0.0471) ( 0.6501, 0.0049)


Theta Component of CS factorization of X:

    Theta
         1
1   0.1973
2   0.5387


Components of CS factorization of X:

    Theta
         1
1   0.1973
2   0.5387


    Recombined matrix X = U * Sigma_p * V^H
                   1                  2                  3                  4
1  (-0.0130,-0.3260) (-0.1404,-0.2617) ( 0.2518,-0.7979) (-0.0510,-0.2175)
```

```
 2  ( 0.4276,-0.6258) ( 0.0863,-0.0382) (-0.3219, 0.1611) ( 0.1198, 0.1632)
 3  (-0.3260, 0.1643) ( 0.3816,-0.1822) ( 0.1323,-0.0146) (-0.5067, 0.1861)
 4  ( 0.1591,-0.0052) (-0.2821, 0.1973) ( 0.2160, 0.1881) (-0.4016, 0.2679)
 5  (-0.1721,-0.0131) (-0.5094,-0.5032) ( 0.0365, 0.2032) ( 0.1927, 0.1557)
 6  (-0.2634,-0.2477) (-0.1086, 0.2847) ( 0.1091,-0.1271) (-0.0882, 0.5617)

                   5                  6
 1  (-0.0459, 0.0001) (-0.0528,-0.2249)
 2  (-0.0803,-0.4361) (-0.0381,-0.2191)
 3  ( 0.0597,-0.5897) (-0.1385,-0.0909)
 4  (-0.0464, 0.3086) (-0.3735,-0.5515)
 5  ( 0.5784,-0.1244) (-0.0188,-0.0557)
 6  ( 0.0158, 0.0471) ( 0.6501, 0.0049)
```