NAG Library Routine Document

G03BCF

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

G03BCF computes Procrustes rotations in which an orthogonal rotation is found so that a transformed matrix best matches a target matrix.

2 Specification

```
SUBROUTINE G03BCF (STAND, PSCALE, N, M, X, LDX, Y, LDY, YHAT, R, LDR,      &
                   ALPHA, RSS, RES, WK, IFAIL)

INTEGER            N, M, LDX, LDY, LDR, IFAIL
REAL (KIND=nag_wp) X(LDX,M), Y(LDY,M), YHAT(LDY,M), R(LDR,M), ALPHA,       &
                   RSS, RES(N), WK(M*M+7*M)
CHARACTER(1)       STAND, PSCALE
```

3 Description

Let X and Y be n by m matrices. They can be considered as representing sets of n points in an m-dimensional space. The X matrix may be a matrix of loadings from say factor or canonical variate analysis, and the Y matrix may be a postulated pattern matrix or the loadings from a different sample. The problem is to relate the two sets of points without disturbing the relationships between the points in each set. This can be achieved by translating, rotating and scaling the sets of points. The Y matrix is considered as the target matrix and the X matrix is rotated to match that matrix.

First the two sets of points are translated so that their centroids are at the origin to give X_c and Y_c, i.e., the matrices will have zero column means. Then the rotation of the translated X_c matrix which minimizes the sum of squared distances between corresponding points in the two sets is found. This is computed from the singular value decomposition of the matrix:

$$X_c^{\mathrm{T}} Y_c = UDV^{\mathrm{T}},$$

where U and V are orthogonal matrices and D is a diagonal matrix. The matrix of rotations, R, is computed as:

$$R = UV^{\mathrm{T}}.$$

After rotation, a scaling or dilation factor, α, may be estimated by least squares. Thus, the final set of points that best match Y_c is given by:

$$\hat{Y_c} = \alpha X_c R.$$

Before rotation, both sets of points may be normalized to have unit sums of squares or the X matrix may be normalized to have the same sum of squares as the Y matrix. After rotation, the results may be translated to the original Y centroid.

The ith residual, r_i, is given by the distance between the point given in the ith row of Y and the point given in the ith row of $\hat{Y}$. The residual sum of squares is also computed.

4 References

Krzanowski W J (1990) *Principles of Multivariate Analysis* Oxford University Press

Lawley D N and Maxwell A E (1971) *Factor Analysis as a Statistical Method* (2nd Edition) Butterworths

5 Arguments

1: STAND – CHARACTER(1) *Input*

On entry: indicates if translation/normalization is required.

STAND = 'N'
There is no translation or normalization.

STAND = 'Z'
There is translation to the origin (i.e., to zero).

STAND = 'C'
There is translation to origin and then to the Y centroid after rotation.

STAND = 'U'
There is unit normalization.

STAND = 'S'
There is translation and normalization (i.e., there is standardization).

STAND = 'M'
There is translation and normalization to Y scale, then translation to the Y centroid after rotation (i.e., they are matched).

Constraint: STAND = 'N', 'Z', 'C', 'U', 'S' or 'M'.

2: PSCALE – CHARACTER(1) *Input*

On entry: indicates if least squares scaling is to be applied after rotation.

PSCALE = 'S'
Scaling is applied.

PSCALE = 'U'
No scaling is applied.

Constraint: PSCALE = 'S' or 'U'.

3: N – INTEGER *Input*

On entry: n, the number of points.

Constraint: N $\geq$ M.

4: M – INTEGER *Input*

On entry: m, the number of dimensions.

Constraint: M ≥ 1.

5: X(LDX, M) – REAL (KIND=nag_wp) array *Input/Output*

On entry: X, the matrix to be rotated.

On exit: if STAND = 'N', X will be unchanged.

If STAND = 'Z', 'C', 'S' or 'M', X will be translated to have zero column means.

If STAND = 'U' or 'S', X will be scaled to have unit sum of squares.

If STAND = 'M', X will be scaled to have the same sum of squares as Y.

6: LDX – INTEGER *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which G03BCF is called.

Constraint: LDX $\geq$ N.

7: Y(LDY, M) – REAL (KIND=nag_wp) array *Input/Output*

On entry: the target matrix, Y.

On exit: if STAND = 'N', Y will be unchanged.

If STAND = 'Z' or 'S', Y will be translated to have zero column means.

If STAND = 'U' or 'S', Y will be scaled to have unit sum of squares.

If STAND = 'C' or 'M', Y will be translated and then after rotation translated back. The output Y should be the same as the input Y except for rounding errors.

8: LDY – INTEGER *Input*

On entry: the first dimension of the arrays Y and YHAT as declared in the (sub)program from which G03BCF is called.

Constraint: LDY $\geq$ N.

9: YHAT(LDY, M) – REAL (KIND=nag_wp) array *Output*

On exit: the fitted matrix, $\hat{Y}$.

10: R(LDR, M) – REAL (KIND=nag_wp) array *Output*

On exit: the matrix of rotations, R, see Section 9.

11: LDR – INTEGER *Input*

On entry: the first dimension of the array R as declared in the (sub)program from which G03BCF is called.

Constraint: LDR $\geq$ M.

12: ALPHA – REAL (KIND=nag_wp) *Output*

On exit: if PSCALE = 'S' the scaling factor, α; otherwise ALPHA is not set.

13: RSS – REAL (KIND=nag_wp) *Output*

On exit: the residual sum of squares.

14: RES(N) – REAL (KIND=nag_wp) array *Output*

On exit: the residuals, r_i, for $i = 1, 2, \ldots, n$.

15: WK(M $\times$ M + 7 $\times$ M) – REAL (KIND=nag_wp) array *Workspace*

16: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation for details.

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, if you are not familiar with this argument, the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

On exit: IFAIL = 0 unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

If on entry IFAIL $= 0$ or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

IFAIL $= 1$

On entry, N $<$ M,
or M $<$ 1,
or LDX $<$ N,
or LDY $<$ N,
or LDR $<$ M,
or STAND $\neq$ 'N', 'Z', 'C', 'U', 'S' or 'M',
or PSCALE $\neq$ 'S' or 'U'.

IFAIL $= 2$

On entry, either X or Y contain only zero-points (possibly after translation) when normalization is to be applied.

IFAIL $= 3$

The $\hat{Y}$ matrix contains only zero-points when least squares scaling is applied.

IFAIL $= 4$

The singular value decomposition has failed to converge. This is an unlikely error exit.

IFAIL $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

See Section 3.9 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -399$

Your licence key may have expired or may not have been installed correctly.

See Section 3.8 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.7 in How to Use the NAG Library and its Documentation for further information.

7 Accuracy

The accuracy of the calculation of the rotation matrix largely depends upon the singular value decomposition. See the F08 Chapter Introduction for further details.

8 Parallelism and Performance

G03BCF 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

Note that if the matrix $X_c^{\mathrm{T}}Y$ is not of full rank, then the matrix of rotations, R, may not be unique even if there is a unique solution in terms of the rotated matrix, $\hat{Y}_c$. The matrix R may also include reflections as well as pure rotations, see Krzanowski (1990).

If the column dimensions of the X and Y matrices are not equal, the smaller of the two should be supplemented by columns of zeros. Adding a column of zeros to both X and Y will have the effect of allowing reflections as well as rotations.

10 Example

Three points representing the vertices of a triangle in two dimensions are input. The points are translated and rotated to match the triangle given by $(0,0)$, $(1,0)$, $(0,2)$ and scaling is applied after rotation. The target matrix and fitted matrix are printed along with additional information.

10.1 Program Text

```
    Program g03bcfe

!     G03BCF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: g03bcf, nag_wp, x04caf
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: alpha, rss
      Integer                          :: i, ifail, ldr, ldx, ldy, lwk, m, n
      Character (1)                    :: pscale, stand
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: r(:,:), res(:), wk(:), x(:,:),       &
                                          y(:,:), yhat(:,:)
!     .. Executable Statements ..
      Write (nout,*) 'G03BCF Example Program Results'
      Write (nout,*)
      Flush (nout)

!     Skip heading in data file
      Read (nin,*)

!     Read in problem size
      Read (nin,*) n, m, stand, pscale

      ldx = n
      ldy = n
      ldr = m
      lwk = m*m + 7*m
      Allocate (x(ldx,m),y(ldy,m),yhat(ldy,m),r(ldr,m),res(n),wk(lwk))

!     Read in data
      Read (nin,*)(x(i,1:m),i=1,n)
      Read (nin,*)(y(i,1:m),i=1,n)

!     Calculate rotations
      ifail = 0
      Call g03bcf(stand,pscale,n,m,x,ldx,y,ldy,yhat,r,ldr,alpha,rss,res,wk,    &
        ifail)

!     Display results
      ifail = 0
      Call x04caf('General',' ',m,m,r,ldr,'Rotation Matrix',ifail)
      If (pscale=='S' .Or. pscale=='s') Then
```

```
        Write (nout,*)
        Write (nout,99999) ' Scale factor = ', alpha
      End If
      Write (nout,*)
      Flush (nout)
      ifail = 0
      Call x04caf('General',' ',n,m,y,ldy,'Target Matrix',ifail)
      Write (nout,*)
      Flush (nout)
      ifail = 0
      Call x04caf('General',' ',n,m,yhat,ldy,'Fitted Matrix',ifail)
      Write (nout,*)
      Write (nout,99999) 'RSS = ', rss

99999 Format (1X,A,F10.3)
    End Program g03bcfe
```

10.2 Program Data

```
G03BCF Example Program Data
 3 2 'c' 's'
 0.63 0.58
 1.36 0.39
 1.01 1.76
 0.0 0.0
 1.0 0.0
 0.0 2.0
```

10.3 Program Results

```
 G03BCF Example Program Results

 Rotation Matrix
          1       2
 1   0.9673  0.2536
 2  -0.2536  0.9673

  Scale factor =      1.556

 Target Matrix
             1          2
 1      0.0000     0.0000
 2      1.0000     0.0000
 3      0.0000     2.0000

 Fitted Matrix
             1          2
 1     -0.0934     0.0239
 2      1.0805     0.0259
 3      0.0130     1.9502

 RSS =      0.019
```