

NAG Library Routine Document

G02HBF

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

G02HBF finds, for a real matrix X of full column rank, a lower triangular matrix A such that $(A^T A)^{-1}$ is proportional to a robust estimate of the covariance of the variables. G02HBF is intended for the calculation of weights of bounded influence regression using G02HDF.

2 Specification

```

SUBROUTINE G02HBF (UCV, N, M, X, LDX, A, Z, BL, BD, TOL, MAXIT, NITMON,      &
                  NIT, WK, IFAIL)
INTEGER          N, M, LDX, MAXIT, NITMON, NIT, IFAIL
REAL (KIND=nag_wp) UCV, X(LDX,M), A(M*(M+1)/2), Z(N), BL, BD, TOL,      &
                  WK(M*(M+1)/2)
EXTERNAL        UCV

```

3 Description

In fitting the linear regression model

$$y = X\theta + \epsilon,$$

where y is a vector of length n of the dependent variable,

X is an n by m matrix of independent variables,

θ is a vector of length m of unknown parameters,

and ϵ is a vector of length n of unknown errors,

it may be desirable to bound the influence of rows of the X matrix. This can be achieved by calculating a weight for each observation. Several schemes for calculating weights have been proposed (see Hampel *et al.* (1986) and Marazzi (1987)). As the different independent variables may be measured on different scales one group of proposed weights aims to bound a standardized measure of influence. To obtain such weights the matrix A has to be found such that

$$\frac{1}{n} \sum_{i=1}^n u(\|z_i\|_2) z_i z_i^T = I \quad (I \text{ is the identity matrix})$$

and

$$z_i = Ax_i,$$

where x_i is a vector of length m containing the elements of the i th row of X ,

A is an m by m lower triangular matrix,

z_i is a vector of length m ,

and u is a suitable function.

The weights for use with G02HDF may then be computed using

$$w_i = f(\|z_i\|_2)$$

for a suitable user-supplied function f .

G02HBF finds A using the iterative procedure

$$A_k = (S_k + I)A_{k-1},$$

where $S_k = (s_{jl})$, for $j = 1, 2, \dots, m$ and $l = 1, 2, \dots, m$, is a lower triangular matrix such that

$$s_{jl} = \begin{cases} -\min[\max(h_{jl}/n, -BL), BL], & j > l \\ -\min[\max(\frac{1}{2}(h_{jj}/n - 1), -BD), BD], & j = l \end{cases}$$

$$h_{jl} = \sum_{i=1}^n u(\|z_i\|_2) z_{ij} z_{il}$$

and BD and BL are suitable bounds.

In addition the values of $\|z_i\|_2$, for $i = 1, 2, \dots, n$, are calculated.

G02HBF is based on routines in ROBETH; see Marazzi (1987).

4 References

Hampel F R, Ronchetti E M, Rousseeuw P J and Stahel W A (1986) *Robust Statistics. The Approach Based on Influence Functions* Wiley

Huber P J (1981) *Robust Statistics* Wiley

Marazzi A (1987) Weights for bounded influence regression in ROBETH *Cah. Rech. Doc. IUMSP, No. 3 ROB 3* Institut Universitaire de Médecine Sociale et Préventive, Lausanne

5 Parameters

1: UCV – REAL (KIND=nag_wp) FUNCTION, supplied by the user. *External Procedure*

UCV must return the value of the function u for a given value of its argument. The value of u must be non-negative.

The specification of UCV is:

```
FUNCTION UCV (T)
```

```
REAL (KIND=nag_wp) UCV
```

```
REAL (KIND=nag_wp) T
```

```
1: T – REAL (KIND=nag_wp)
```

Input

On entry: the argument for which UCV must be evaluated.

UCV must either be a module subprogram USED by, or declared as EXTERNAL in, the (sub)program from which G02HBF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

2: N – INTEGER

Input

On entry: n , the number of observations.

Constraint: $N > 1$.

3: M – INTEGER

Input

On entry: m , the number of independent variables.

Constraint: $1 \leq M \leq N$.

- 4: X(LDX,M) – REAL (KIND=nag_wp) array Input
On entry: the real matrix X , i.e., the independent variables. $X(i, j)$ must contain the ij th element of X , for $i = 1, 2, \dots, n$ and $j = 1, 2, \dots, m$.
- 5: LDX – INTEGER Input
On entry: the first dimension of the array X as declared in the (sub)program from which G02HBF is called.
Constraint: $LDX \geq N$.
- 6: A(M × (M + 1)/2) – REAL (KIND=nag_wp) array Input/Output
On entry: an initial estimate of the lower triangular real matrix A . Only the lower triangular elements must be given and these should be stored row-wise in the array.
 The diagonal elements must be $\neq 0$, although in practice will usually be > 0 . If the magnitudes of the columns of X are of the same order the identity matrix will often provide a suitable initial value for A . If the columns of X are of different magnitudes, the diagonal elements of the initial value of A should be approximately inversely proportional to the magnitude of the columns of X .
On exit: the lower triangular elements of the matrix A , stored row-wise.
- 7: Z(N) – REAL (KIND=nag_wp) array Output
On exit: the value $\|z_i\|_2$, for $i = 1, 2, \dots, n$.
- 8: BL – REAL (KIND=nag_wp) Input
On entry: the magnitude of the bound for the off-diagonal elements of S_k .
Suggested value: $BL = 0.9$.
Constraint: $BL > 0.0$.
- 9: BD – REAL (KIND=nag_wp) Input
On entry: the magnitude of the bound for the diagonal elements of S_k .
Suggested value: $BD = 0.9$.
Constraint: $BD > 0.0$.
- 10: TOL – REAL (KIND=nag_wp) Input
On entry: the relative precision for the final value of A . Iteration will stop when the maximum value of $|s_{jl}|$ is less than TOL.
Constraint: $TOL > 0.0$.
- 11: MAXIT – INTEGER Input
On entry: the maximum number of iterations that will be used during the calculation of A .
 A value of $MAXIT = 50$ will often be adequate.
Constraint: $MAXIT > 0$.
- 12: NITMON – INTEGER Input
On entry: determines the amount of information that is printed on each iteration.
 $NITMON > 0$
 The value of A and the maximum value of $|s_{jl}|$ will be printed at the first and every NITMON iterations.
 $NITMON \leq 0$
 No iteration monitoring is printed.

When printing occurs the output is directed to the current advisory message unit (see X04ABF).

13: NIT – INTEGER *Output*

On exit: the number of iterations performed.

14: WK($M \times (M + 1)/2$) – REAL (KIND=nag_wp) array *Workspace*

15: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this parameter you should refer to Section 3.3 in the Essential Introduction 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 parameter, 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, $N < 1$,
or $M < 1$,
or $N < M$,
or $LDX < N$.

IFAIL = 2

On entry, $TOL \leq 0.0$,
or $MAXIT \leq 0$,
or diagonal element of $A = 0.0$,
or $BL \leq 0.0$,
or $BD \leq 0.0$.

IFAIL = 3

Value returned by $UCV < 0$.

IFAIL = 4

The routine has failed to converge in MAXIT iterations.

7 Accuracy

On successful exit the accuracy of the results is related to the value of TOL; see Section 5.

8 Further Comments

The existence of A will depend upon the function u ; (see Hampel *et al.* (1986) and Marazzi (1987)), also if X is not of full rank a value of A will not be found. If the columns of X are almost linearly related then convergence will be slow.

9 Example

This example reads in a matrix of real numbers and computes the Krasker–Welsch weights (see Marazzi (1987)). The matrix A and the weights are then printed.

9.1 Program Text

```

! G02HBF Example Program Text
! Mark 24 Release. NAG Copyright 2012.

Module g02hbfe_mod

! G02HBF Example Program Module:
! Parameters and User-defined Routines

! .. Use Statements ..
Use nag_library, Only: nag_wp
! .. Implicit None Statement ..
Implicit None
! .. Parameters ..
Real (Kind=nag_wp), Parameter      :: one = 1.0_nag_wp
Real (Kind=nag_wp), Parameter      :: two = 2.0_nag_wp
Real (Kind=nag_wp), Parameter      :: zero = 0.0_nag_wp
Integer, Parameter                  :: iset = 1, nin = 5, nout = 6
Contains
Function ucv(t)
! UCV function for Krasker-Welsch weights

! .. Use Statements ..
Use nag_library, Only: s15abf, x01aaf, x02akf
! .. Function Return Value ..
Real (Kind=nag_wp)                  :: ucv
! .. Parameters ..
Real (Kind=nag_wp), Parameter      :: ucvc = 2.5_nag_wp
! .. Scalar Arguments ..
Real (Kind=nag_wp), Intent (In)    :: t
! .. Local Scalars ..
Real (Kind=nag_wp)                  :: pc, pd, q, q2
Integer                              :: ifail
! .. Intrinsic Procedures ..
Intrinsic                            :: exp, log, sqrt
! .. Executable Statements ..
ucv = one
If (t/=zero) Then
    q = ucvc/t
    q2 = q*q
    ifail = 0
    pc = s15abf(q,ifail)
    If (q2<-log(x02akf())) Then
        pd = exp(-q2/two)/sqrt(x01aaf(zero)*two)
    Else
        pd = zero
    End If
    ucv = (two*pc-one)*(one-q2) + q2 - two*q*pd
End If
Return
End Function ucv
End Module g02hbfe_mod
Program g02hbfe

! G02HBF Example Main Program

! .. Use Statements ..
Use nag_library, Only: g02hbf, nag_wp, x04abf, x04ccf
Use g02hbfe_mod, Only: iset, nin, nout, one, ucv
! .. Implicit None Statement ..
Implicit None
! .. Local Scalars ..
Real (Kind=nag_wp)                  :: bd, bl, tol

```

```

Integer                                :: i, ifail, la, ldx, m, maxit, n, &
                                       nadv, nit, nitmon
! .. Local Arrays ..
Real (Kind=nag_wp), Allocatable        :: a(:), wk(:), x(:,,:), z(:)
! .. Executable Statements ..
Write (nout,*) 'G02HBF Example Program Results'
Write (nout,*)
Flush (nout)

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

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

   ldx = n
   la = (m+1)*m/2
   Allocate (x(ldx,m),a(la),wk(la),z(n))

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

! Read in initial value of A
Read (nin,*) a(1:la)

! Read in control parameters
Read (nin,*) nitmon, bl, bd, maxit, tol

! Set the advisory channel to NOUT for monitoring information
If (nitmon/=0) Then
   nadv = nout
   Call x04abf(iset,nadv)
End If

! Calculate A
ifail = 0
Call g02hbf(ucv,n,m,x,ldx,a,z,bl,bd,tol,maxit,nitmon,nit,wk,ifail)

! Display results
Write (nout,99999) 'G02HBF required ', nit, ' iterations to converge'
Write (nout,*)
Flush (nout)
ifail = 0
Call x04ccf('Lower','Non-Unit',m,a,'Matrix A',ifail)
Write (nout,*)
Write (nout,*) 'Vector Z'
Write (nout,99998)(z(i),i=1,n)
Write (nout,*)
Write (nout,*) 'Vector of Krasker-Welsch weights'
Write (nout,99998)(one/z(i),i=1,n)

99999 Format (1X,A,I0,A)
99998 Format (1X,F9.4)
End Program g02hbfe

```

9.2 Program Data

G02HBF Example Program Data

```

   5      3      : N,M
   1.0 -1.0 -1.0
   1.0 -1.0  1.0
   1.0  1.0 -1.0
   1.0  1.0  1.0
   1.0  0.0  3.0      : End of X1,X2 and X3 values
   1.0 0.0 1.0 0.0 0.0 1.0 : Initial values for A
0 0.9 0.9 50 5.0E-5      : NITMON,BL,BD,MAXIT,TOL

```

9.3 Program Results

G02HBF Example Program Results

G02HBF required 16 iterations to converge

Matrix A

	1	2	3
1	1.3208E+00		
2	1.7023E-17	-5.7532E-01	
3	1.4518E+00	2.7351E-17	9.3403E-01

Vector Z

2.4760
1.9953
2.4760
1.9953
2.5890

Vector of Krasker-Welsch weights

0.4039
0.5012
0.4039
0.5012
0.3862
