NAG Library Routine Document

G02HDF

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

G02HDF performs bounded influence regression (M-estimates) using an iterative weighted least squares algorithm.

2 Specification

```
SUBROUTINE G02HDF (CHI, PSI, PSIP0, BETA, INDW, ISIGMA, N, M, X, LDX, Y,    &
                   WGT, THETA, K, SIGMA, RS, TOL, EPS, MAXIT, NITMON,      &
                   NIT, WK, IFAIL)
INTEGER            INDW, ISIGMA, N, M, LDX, K, MAXIT, NITMON, NIT,         &
                   IFAIL
REAL (KIND=nag_wp) CHI, PSI, PSIP0, BETA, X(LDX,M), Y(N), WGT(N),          &
                   THETA(M), SIGMA, RS(N), TOL, EPS, WK((M+4)*N)
EXTERNAL           CHI, PSI
```

3 Description

For the linear regression model

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

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

X is a n by m matrix of independent variables of column rank k,

θ is a vector of length m of unknown arguments,

and ϵ is a vector of length n of unknown errors with var $(\epsilon_i) = \sigma^2$,

G02HDF calculates the M-estimates given by the solution, $\hat{\theta}$, to the equation

$$\sum_{i=1}^{n} \psi(r_i/(\sigma w_i)) w_i x_{ij} = 0, \quad j = 1, 2, \ldots, m, \tag{1}$$

where r_i is the ith residual, i.e., the ith element of the vector $r = y - X\hat{\theta}$,

ψ is a suitable weight function,

w_i are suitable weights such as those that can be calculated by using output from G02HBF,

and σ may be estimated at each iteration by the median absolute deviation of the residuals $\hat{\sigma} = \text{med}_i\,[|r_i|]/\beta_1$

or as the solution to

$$\sum_{i=1}^{n} \chi(r_i/(\hat{\sigma} w_i)) w_i^2 = (n-k)\beta_2$$

for a suitable weight function χ, where β_1 and β_2 are constants, chosen so that the estimator of σ is asymptotically unbiased if the errors, ϵ_i, have a Normal distribution. Alternatively σ may be held at a constant value.

The above describes the Schweppe type regression. If the w_i are assumed to equal 1 for all i, then Huber type regression is obtained. A third type, due to Mallows, replaces (1) by

$$\sum_{i=1}^{n} \psi(r_i/\sigma) w_i x_{ij} = 0, \quad j = 1, 2, \ldots, m.$$

This may be obtained by use of the transformations

$$\begin{aligned} w_i^* &\leftarrow \sqrt{w_i} \\ y_i^* &\leftarrow y_i \sqrt{w_i} \\ x_{ij}^* &\leftarrow x_{ij} \sqrt{w_i}, \quad j = 1, 2, \ldots, m \end{aligned}$$

(see Marazzi (1987)).

The calculation of the estimates of θ can be formulated as an iterative weighted least squares problem with a diagonal weight matrix G given by

$$G_{ii} = \begin{cases} \dfrac{\psi(r_i/(\sigma w_i))}{(r_i/(\sigma w_i))}, & r_i \neq 0 \\ \\ \psi'(0), & r_i = 0. \end{cases}.$$

The value of θ at each iteration is given by the weighted least squares regression of y on X. This is carried out by first transforming the y and X by

$$\begin{aligned} \tilde{y}_i &= y_i \sqrt{G_{ii}} \\ \tilde{x}_{ij} &= x_{ij} \sqrt{G_{ii}}, \quad j = 1, 2, \ldots, m \end{aligned}$$

and then using F04JGF . If X is of full column rank then an orthogonal-triangular (QR) decomposition is used; if not, a singular value decomposition is used.

Observations with zero or negative weights are not included in the solution.

Note: there is no explicit provision in the routine for a constant term in the regression model. However, the addition of a dummy variable whose value is 1.0 for all observations will produce a value of $\hat{\theta}$ corresponding to the usual constant term.

G02HDF 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) Subroutines for robust and bounded influence regression in ROBETH *Cah. Rech. Doc. IUMSP, No. 3 ROB 2* Institut Universitaire de Médecine Sociale et Préventive, Lausanne

5 Arguments

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

If ISIGMA > 0, CHI must return the value of the weight function χ for a given value of its argument. The value of χ must be non-negative.

The specification of CHI is:

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

1: T – REAL (KIND=nag_wp) *Input*

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

CHI must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub) program from which G02HDF is called. Arguments denoted as *Input* must **not** be changed by this procedure.

If ISIGMA ≤ 0, the actual argument CHI may be the dummy routine G02HDZ. (G02HDZ is included in the NAG Library.)

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

PSI must return the value of the weight function ψ for a given value of its argument.

The specification of PSI is:

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

1: T – REAL (KIND=nag_wp) *Input*

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

PSI must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub) program from which G02HDF is called. Arguments denoted as *Input* must **not** be changed by this procedure.

3: PSIP0 – REAL (KIND=nag_wp) *Input*

On entry: the value of $\psi\prime(0)$.

4: BETA – REAL (KIND=nag_wp) *Input*

On entry: if ISIGMA < 0, BETA must specify the value of β_1.

For Huber and Schweppe type regressions, β_1 is the 75th percentile of the standard Normal distribution (see G01FAF). For Mallows type regression β_1 is the solution to

$$\frac{1}{n}\sum_{i=1}^{n}\Phi(\beta_1/\sqrt{w_i}) = 0.75,$$

where Φ is the standard Normal cumulative distribution function (see S15ABF).

If ISIGMA > 0, BETA must specify the value of β_2.

$$\beta_2 = \int_{-\infty}^{\infty}\chi(z)\phi(z)\,dz, \qquad \text{in the Huber case;}$$

$$\beta_2 = \frac{1}{n}\sum_{i=1}^{n}w_i\int_{-\infty}^{\infty}\chi(z)\phi(z)\,dz, \qquad \text{in the Mallows case;}$$

$$\beta_2 = \frac{1}{n}\sum_{i=1}^{n}w_i^2\int_{-\infty}^{\infty}\chi(z/w_i)\phi(z)\,dz, \quad \text{in the Schweppe case;}$$

where ϕ is the standard normal density, i.e., $\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}x^2\right)$.

If ISIGMA $= 0$, BETA is not referenced.

Constraint: if ISIGMA $\neq 0$, BETA > 0.0.

5: INDW – INTEGER *Input*

On entry: determines the type of regression to be performed.

INDW $= 0$
Huber type regression.

INDW < 0
Mallows type regression.

INDW > 0
Schweppe type regression.

6: ISIGMA – INTEGER *Input*

On entry: determines how σ is to be estimated.

ISIGMA $= 0$
σ is held constant at its initial value.

ISIGMA < 0
σ is estimated by median absolute deviation of residuals.

ISIGMA > 0
σ is estimated using the χ function.

7: N – INTEGER *Input*

On entry: n, the number of observations.

Constraint: N > 1.

8: M – INTEGER *Input*

On entry: m, the number of independent variables.

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

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

On entry: the values of the X matrix, i.e., the independent variables. X(i,j) must contain the ijth element of X, for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$.

If INDW < 0, during calculations the elements of X will be transformed as described in Section 3. Before exit the inverse transformation will be applied. As a result there may be slight differences between the input X and the output X.

On exit: unchanged, except as described above.

10: LDX – INTEGER *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which G02HDF is called.

Constraint: LDX $\geq$ N.

11: Y(N) – REAL (KIND=nag_wp) array *Input/Output*

On entry: the data values of the dependent variable.

Y(i) must contain the value of y for the ith observation, for $i = 1, 2, \ldots, n$.

If INDW < 0, during calculations the elements of Y will be transformed as described in Section 3. Before exit the inverse transformation will be applied. As a result there may be slight differences between the input Y and the output Y.

On exit: unchanged, except as described above.

12: WGT(N) – REAL (KIND=nag_wp) array *Input/Output*

On entry: the weight for the ith observation, for $i = 1, 2, \ldots, n$.

If INDW < 0, during calculations elements of WGT will be transformed as described in Section 3. Before exit the inverse transformation will be applied. As a result there may be slight differences between the input WGT and the output WGT.

If WGT$(i) \leq 0$, the ith observation is not included in the analysis.

If INDW $= 0$, WGT is not referenced.

On exit: unchanged, except as described above.

13: THETA(M) – REAL (KIND=nag_wp) array *Input/Output*

On entry: starting values of the argument vector θ. These may be obtained from least squares regression. Alternatively if ISIGMA < 0 and SIGMA $= 1$ or if ISIGMA > 0 and SIGMA approximately equals the standard deviation of the dependent variable, y, then THETA$(i) = 0.0$, for $i = 1, 2, \ldots, m$ may provide reasonable starting values.

On exit: the M-estimate of θ_i, for $i = 1, 2, \ldots, m$.

14: K – INTEGER *Output*

On exit: the column rank of the matrix X.

15: SIGMA – REAL (KIND=nag_wp) *Input/Output*

On entry: a starting value for the estimation of σ. SIGMA should be approximately the standard deviation of the residuals from the model evaluated at the value of θ given by THETA on entry.

Constraint: SIGMA > 0.0.

On exit: the final estimate of σ if ISIGMA $\neq 0$ or the value assigned on entry if ISIGMA $= 0$.

16: RS(N) – REAL (KIND=nag_wp) array *Output*

On exit: the residuals from the model evaluated at final value of THETA, i.e., RS contains the vector $\left(y - X\hat{\theta}\right)$.

17: TOL – REAL (KIND=nag_wp) *Input*

On entry: the relative precision for the final estimates. Convergence is assumed when both the relative change in the value of SIGMA and the relative change in the value of each element of THETA are less than TOL.

It is advisable for TOL to be greater than $100 \times$ ***machine precision***.

Constraint: TOL > 0.0.

18: EPS – REAL (KIND=nag_wp) *Input*

On entry: a relative tolerance to be used to determine the rank of X. See F04JGF for further details.

If EPS $<$ ***machine precision*** or EPS > 1.0 then ***machine precision*** will be used in place of TOL.

A reasonable value for EPS is 5.0×10^{-6} where this value is possible.

19: MAXIT – INTEGER *Input*

On entry: the maximum number of iterations that should be used during the estimation.

A value of MAXIT $= 50$ should be adequate for most uses.

Constraint: MAXIT > 0.

20: NITMON – INTEGER *Input*

On entry: determines the amount of information that is printed on each iteration.

NITMON ≤ 0
No information is printed.

NITMON > 0
On the first and every NITMON iterations the values of SIGMA, THETA and the change in THETA during the iteration are printed.

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

21: NIT – INTEGER *Output*

On exit: the number of iterations that were used during the estimation.

22: WK$((\mathrm{M}+4)\times \mathrm{N})$ – REAL (KIND=nag_wp) array *Workspace*

23: 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, because for this routine the values of the output arguments may be useful even if IFAIL $\neq 0$ on exit, the recommended value is -1. **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).

Note: G02HDF may return useful information for one or more of the following detected errors or warnings.

Errors or warnings detected by the routine:

IFAIL $= 1$

On entry, N ≤ 1,
or N < 1,
or N $\leq$ M,
or LDX $<$ N.

IFAIL $= 2$

On entry, BETA ≤ 0.0, and ISIGMA $\neq 0$,
or SIGMA ≤ 0.0.

IFAIL $= 3$

On entry, TOL ≤ 0.0,
or MAXIT ≤ 0.

IFAIL $= 4$

A value returned by the CHI function is negative.

IFAIL $= 5$

During iterations a value of SIGMA ≤ 0.0 was encountered.

IFAIL $= 6$

A failure occurred in F04JGF . This is an extremely unlikely error. If it occurs, please contact NAG.

IFAIL $= 7$

The weighted least squares equations are not of full rank. This may be due to the X matrix not being of full rank, in which case the results will be valid. It may also occur if some of the G_{ii} values become very small or zero, see Section 9. The rank of the equations is given by K. If the matrix just fails the test for nonsingularity then the result IFAIL $= 7$ and K $=$ M is possible (see F04JGF).

IFAIL $= 8$

The routine has failed to converge in MAXIT iterations.

IFAIL $= 9$

Having removed cases with zero weight, the value of N $-$ K ≤ 0, i.e., no degree of freedom for error. This error will only occur if ISIGMA > 0.

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 results is controlled by TOL. For the accuracy of the weighted least squares see F04JGF.

8 Parallelism and Performance

G02HDF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

G02HDF 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

In cases when ISIGMA $\neq 0$ it is important for the value of SIGMA to be of a reasonable magnitude. Too small a value may cause too many of the winsorized residuals, i.e., $\psi(r_i/\sigma)$, to be zero, which will lead to convergence problems and may trigger the IFAIL $= 7$ error.

By suitable choice of the functions CHI and PSI this routine may be used for other applications of iterative weighted least squares.

For the variance-covariance matrix of θ see G02HFF.

10 Example

Having input X, Y and the weights, a Schweppe type regression is performed using Huber's ψ function. The subroutine BETCAL calculates the appropriate value of β_2.

10.1 Program Text

```
!   G02HDF Example Program Text
!   Mark 26 Release. NAG Copyright 2016.

    Module g02hdfe_mod

!     G02HDF Example Program Module:
!            Parameters and User-defined Routines

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Accessibility Statements ..
      Private
      Public                            :: betcal, chi, psi
!     .. Parameters ..
      Real (Kind=nag_wp), Parameter     :: dchi = 1.5_nag_wp
      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, Public        :: iset = 1, nin = 5, nout = 6
    Contains
      Function psi(t)

!       .. Function Return Value ..
        Real (Kind=nag_wp)              :: psi
!       .. Parameters ..
        Real (Kind=nag_wp), Parameter   :: c = 1.5_nag_wp
!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (In) :: t
!       .. Intrinsic Procedures ..
        Intrinsic                       :: abs
!       .. Executable Statements ..
        If (t<=-c) Then
          psi = -c
        Else If (abs(t)<c) Then
          psi = t
        Else
          psi = c
        End If
        Return
      End Function psi

      Function chi(t)

!       .. Function Return Value ..
        Real (Kind=nag_wp)              :: chi
!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (In) :: t
!       .. Local Scalars ..
```

```
        Real (Kind=nag_wp)             :: ps
!       .. Intrinsic Procedures ..
        Intrinsic                      :: abs
!       .. Executable Statements ..
        ps = dchi
        If (abs(t)<dchi) Then
          ps = t
        End If
        chi = ps*ps/two
        Return
      End Function chi

      Subroutine betcal(n,wgt,beta)
!       Calculate BETA for Schweppe type regression

!       .. Use Statements ..
        Use nag_library, Only: s15abf, x01aaf, x02akf
!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (Out) :: beta
        Integer, Intent (In)           :: n
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (In) :: wgt(n)
!       .. Local Scalars ..
        Real (Kind=nag_wp)             :: amaxex, anormc, b, d2, dc, dw, dw2,  &
                                          pc, w2
        Integer                        :: i, ifail
!       .. Intrinsic Procedures ..
        Intrinsic                      :: exp, log, real, sqrt
!       .. Executable Statements ..
        amaxex = -log(x02akf())
        anormc = sqrt(x01aaf(zero)*two)
        d2 = dchi*dchi
        beta = zero
        Do i = 1, n
          w2 = wgt(i)*wgt(i)
          dw = wgt(i)*dchi
          ifail = 0
          pc = s15abf(dw,ifail)
          dw2 = dw*dw
          dc = zero
          If (dw2<amaxex) Then
            dc = exp(-dw2/two)/anormc
          End If
          b = (-dw*dc+pc-0.5_nag_wp)/w2 + (one-pc)*d2
          beta = b*w2/real(n,kind=nag_wp) + beta
        End Do
        Return
      End Subroutine betcal
    End Module g02hdfe_mod
    Program g02hdfe
!     G02HDF Example Main Program

!     .. Use Statements ..
      Use nag_library, Only: g02hdf, nag_wp, x04abf
      Use g02hdfe_mod, Only: betcal, chi, iset, nin, nout, psi
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: beta, eps, psip0, sigma, tol
      Integer                          :: i, ifail, indw, isigma, k, ldx, m,   &
                                          maxit, n, nadv, nit, nitmon
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: rs(:), theta(:), wgt(:), wk(:),      &
                                          x(:,:), y(:)
!     .. Executable Statements ..
      Write (nout,*) 'G02HDF Example Program Results'
      Write (nout,*)

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

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

      ldx = n
      Allocate (x(ldx,m),y(n),wgt(n),theta(m),rs(n),wk((m+4)*n))

!     Read in data
      Read (nin,*)(x(i,2:m),y(i),wgt(i),i=1,n)

!     Set first column of X to 1 for the constant term
      x(1:n,1) = 1.0E0_nag_wp

!     Set BETA
      Call betcal(n,wgt,beta)

!     Read in value for PSI(0)
      Read (nin,*) psip0

!     Read in control parameters
      Read (nin,*) indw, isigma, nitmon, maxit, tol, eps

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

!     Read in initial values
      Read (nin,*) sigma
      Read (nin,*) theta(1:m)

!     Perform bounded influence regression
      ifail = -1
      Call g02hdf(chi,psi,psip0,beta,indw,isigma,n,m,x,ldx,y,wgt,theta,k,      &
        sigma,rs,tol,eps,maxit,nitmon,nit,wk,ifail)
      If (ifail==7) Then
        Write (nout,*) 'Some of the following results may be unreliable'
      Else If (ifail/=0) Then
        Go To 100
      End If

!     Display results
      Write (nout,99999) 'G02HDF required ', nit, ' iterations to converge'
      Write (nout,99999) '                   K = ', k
      Write (nout,99998) '               Sigma = ', sigma
      Write (nout,*) '    THETA'
      Write (nout,99997)(theta(i),i=1,m)
      Write (nout,*)
      Write (nout,*) '  Weights  Residuals'
      Write (nout,99996)(wgt(i),rs(i),i=1,n)

100   Continue

99999 Format (1X,A,I0,A)
99998 Format (1X,A,F9.4)
99997 Format (1X,F9.4)
99996 Format (1X,2F9.4)
    End Program g02hdfe
```

10.2 Program Data

```
G02HDF Example Program Data
  5    3                    :: N,M
 -1.0  -1.0   10.5   0.4039
 -1.0   1.0   11.3   0.5012
  1.0  -1.0   12.6   0.4039
  1.0   1.0   13.4   0.5012
```

```
  0.0   3.0   17.1   0.3862 :: End of X,Y and WT
  1.0                       :: PSIP0
  1  1  0  50 1.0E-5 1.0E-5 :: INDW,ISIGMA,NITMON,MAXIT,TOL,EPS
  1.0                       :: SIGMA
  0.0  0.0  0.0             :: THETA
```

10.3 Program Results

```
 G02HDF Example Program Results

 G02HDF required 5 iterations to converge
                   K = 3
               Sigma =    2.7783
    THETA
  12.2321
   1.0500
   1.2464

  Weights  Residuals
   0.4039   0.5643
   0.5012  -1.1286
   0.4039   0.5643
   0.5012  -1.1286
   0.3862   1.1286
```