# NAG C Library Function Document

# nag_robust_m_corr_user_fn (g02hlc)

## 1 Purpose

nag_robust_m_corr_user_fn (g02hlc) calculates a robust estimate of the covariance matrix for user-supplied weight functions and their derivatives.

## 2 Specification

```
#include <nag.h>
#include <nagg02.h>

void nag_robust_m_corr_user_fn (Nag_OrderType order,
      void (*ucv)(double t, double *u, double *ud, double *w, double *wd,
           Nag_Comm *comm),
      Integer indm, Integer n, Integer m, const double x[], Integer pdx,
      double cov[], double a[], double wt[], double theta[], double bl, double bd,
      Integer maxit, Integer nitmon, const char *outfile, double tol, Integer *nit,
      Nag_Comm *comm, NagError *fail)
```

## 3 Description

For a set of $n$ observations on $m$ variables in a matrix $X$, a robust estimate of the covariance matrix, $C$, and a robust estimate of location, $\theta$, are given by:

$$C = \tau^2 \left(A^{\mathrm{T}} A\right)^{-1},$$

where $\tau^2$ is a correction factor and $A$ is a lower triangular matrix found as the solution to the following equations.

$$z_i = A(x_i - \theta)$$

$$\frac{1}{n}\sum_{i=1}^{n} w\left(\|z_i\|_2\right) z_i = 0$$

and

$$\frac{1}{n}\sum_{i=1}^{n} u\left(\|z_i\|_2\right) z_i z_i^{\mathrm{T}} - v\left(\|z_i\|_2\right) I = 0,$$

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

$z_i$ is a vector of length $m$,

$I$ is the identity matrix and 0 is the zero matrix,

and $w$ and $u$ are suitable functions.

nag_robust_m_corr_user_fn (g02hlc) covers two situations:

(i) $v(t) = 1$ for all $t$,

(ii) $v(t) = u(t)$.

The robust covariance matrix may be calculated from a weighted sum of squares and cross-products matrix about $\theta$ using weights $wt_i = u(\|z_i\|)$. In case (i) a divisor of $n$ is used and in case (ii) a divisor of $\sum_{i=1}^{n} wt_i$ is used. If $w(.) = \sqrt{u(.)}$, then the robust covariance matrix can be calculated by scaling each row of $X$ by $\sqrt{wt_i}$ and calculating an unweighted covariance matrix about $\theta$.

In order to make the estimate asymptotically unbiased under a Normal model a correction factor, $\tau^2$, is needed. The value of the correction factor will depend on the functions employed (see Huber (1981) and Marazzi (1987a)).

nag_robust_m_corr_user_fn (g02hlc) finds $A$ using the iterative procedure as given by Huber.

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

and

$$\theta_{j_k} = \frac{b_j}{D_1} + \theta_{j_{k-1}},$$

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

$$s_{jl} = \begin{cases} -\min\left[\max\left(h_{jl}/D_3, -BL\right), BL\right], & j > l \\ -\min\left[\max\left(\left(h_{jj}/(2D_3 - D_4/D_2)\right), -BD\right), BD\right], & j = l \end{cases}$$

where

$$D_1 = \sum_{i=1}^{n} \left\{w\left(\|z_i\|_2\right) + \tfrac{1}{m}w'\left(\|z_i\|_2\right)\|z_i\|_2\right\}$$

$$D_2 = \sum_{i=1}^{n} \left\{\tfrac{1}{p}\left(u'\left(\|z_i\|_2\right)\|z_i\|_2 + 2u\left(\|z_i\|_2\right)\right)\|z_i\|_2 - v'\left(\|z_i\|_2\right)\right\}\|z_i\|_2$$

$$D_3 = \tfrac{1}{m+2}\sum_{i=1}^{n} \left\{\tfrac{1}{m}\left(u'\left(\|z_i\|_2\right)\|z_i\|_2 + 2u\left(\|z_i\|_2\right)\right) + u\left(\|z_i\|_2\right)\right\}\|z_i\|_2{}^2$$

$$D_4 = \sum_{i=1}^{n} \left\{\tfrac{1}{m}u\left(\|z_i\|_2\right)\|z_i\|_2{}^2 - v\left(\|z_i\|_2{}^2\right)\right\}$$

$$h_{jl} = \sum_{i=1}^{n} u\left(\|z_i\|_2\right)z_{ij}z_{il}, \text{ for } j > l$$

$$h_{jj} = \sum_{i=1}^{n} u\left(\|z_i\|_2\right)\left(z_{ij}^2 - \|z_{ij}\|_2{}^2/m\right)$$

$$b_j = \sum_{i=1}^{n} w\left(\|z_i\|_2\right)\left(x_{ij} - b_j\right)$$

and $BD$ and $BL$ are suitable bounds.

nag_robust_m_corr_user_fn (g02hlc) is based on routines in ROBETH; see Marazzi (1987a).

## 4 References

Huber P J (1981) *Robust Statistics* Wiley

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

## 5 Arguments

1: **order** – Nag_OrderType *Input*

*On entry*: the **order** argument specifies the two-dimensional storage scheme being used, i.e., row-major ordering or column-major ordering. C language defined storage is specified by **order** = **Nag_RowMajor**. See Section 2.2.1.4 of the Essential Introduction for a more detailed explanation of the use of this argument.

*Constraint*: **order** = **Nag_RowMajor** or **Nag_ColMajor**.

2: **ucv** – function, supplied by the user *External Function*

**ucv** must return the values of the functions $u$ and $w$ and their derivatives for a given value of its argument.

Its specification is:

```
void ucv (double t, double *u, double *ud, double *w, double *wd,
      Nag_Comm *comm)
```

1: **t** – double *Input*

*On entry*: the argument for which the functions $u$ and $w$ must be evaluated.

2: **u** – double * *Output*

*On exit*: the value of the $u$ function at the point **t**.

3: **ud** – double * *Output*

*On exit*: the value of the derivative of the $u$ function at the point **t**.

4: **w** – double * *Output*

*On exit*: the value of the $w$ function at the point **t**.

5: **wd** – double * *Output*

*On exit*: the value of the derivative of the $w$ function at the point **t**.

6: **comm** – Nag_Comm * *Communication Structure*

Pointer to structure of type **Nag_Comm**; the following members are relevant to **ucv**.

**user** – double *
**iuser** – Integer *
**p** – Pointer

The type Pointer will be `void *`. Before calling nag_robust_m_corr_user_fn (g02hlc) these pointers may be allocated memory by the user and initialized with various quantities for use by **ucv** when called from nag_robust_m_corr_user_fn (g02hlc).

3: **indm** – Integer *Input*

*On entry*: indicates which form of the function $v$ will be used.

**indm** $= 1$

$v = 1.$

**indm** $\neq 1$

$v = u.$

4: **n** – Integer *Input*

*On entry*: $n$, the number of observations.

*Constraint*: **n** $> 1$.

5: **m** – Integer *Input*

*On entry*: $m$, the number of columns of the matrix $X$, i.e., number of independent variables.

*Constraint*: $1 \leq$ **m** $\leq$ **n**.

6: **x**[$dim$] – const double *Input*

**Note**: the dimension, *dim*, of the array **x** must be at least

$\max(1, \mathbf{pdx} \times \mathbf{m})$ when **order** = **Nag_ColMajor**;
$\max(1, \mathbf{n} \times \mathbf{pdx})$ when **order** = **Nag_RowMajor**.

Where $\mathbf{X}(i,j)$ appears in this document, it refers to the array element

if **order** = **Nag_ColMajor**, $\mathbf{x}[(j-1) \times \mathbf{pdx} + i - 1]$;
if **order** = **Nag_RowMajor**, $\mathbf{x}[(i-1) \times \mathbf{pdx} + j - 1]$.

*On entry*: $\mathbf{X}(i,j)$ must contain the $i$th observation on the $j$th variable, for $i = 1, 2, \ldots, n$; $j = 1, 2, \ldots, m$.

7: **pdx** – Integer *Input*

*On entry*: the stride separating matrix row or column elements (depending on the value of **order**) in the array **x**.

*Constraints*:

if **order** = **Nag_ColMajor**, $\mathbf{pdx} \geq \mathbf{n}$;
if **order** = **Nag_RowMajor**, $\mathbf{pdx} \geq \mathbf{m}$.

8: **cov**[$dim$] – double *Output*

**Note**: the dimension, *dim*, of the array **cov** must be at least $\mathbf{m} \times (\mathbf{m} + 1)/2$.

*On exit*: contains a robust estimate of the covariance matrix, $C$. The upper triangular part of the matrix $C$ is stored packed by columns (lower triangular stored by rows), $C_{ij}$ is returned in $\mathbf{cov}(j \times (j-1)/2 + i)$, $i \leq j$.

9: **a**[$dim$] – double *Input/Output*

**Note**: the dimension, *dim*, of the array **a** must be at least $\mathbf{m} \times (\mathbf{m} + 1)/2$.

*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 $\neq 0$, and 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 $X$.

*Constraint*: $\mathbf{a}[j \times (j-1)/2 + j] \neq 0.0$, for $j = 0, 1, \ldots, m - 1$.

*On exit*: the lower triangular elements of the inverse of the matrix $A$, stored row-wise.

10: **wt**[**n**] – double *Output*

*On exit*: $\mathbf{wt}[i-1]$ contains the weights, $wt_i = u\left(\|z_i\|_2\right)$, for $i = 1, 2, \ldots, n$.

11: **theta**[**m**] – double *Input/Output*

*On entry*: an initial estimate of the location argument, $\theta_j$, for $j = 1, 2, \ldots, m$.

In many cases an initial estimate of $\theta_j = 0$, for $j = 1, 2, \ldots, m$, will be adequate. Alternatively medians may be used as given by nag_median_1var (g07dac).

*On exit*: contains the robust estimate of the location argument, $\theta_j$, for $j = 1, 2, \ldots, m$.

12: **bl** – double *Input*

*On entry*: the magnitude of the bound for the off-diagonal elements of $S_k$, $BL$.

*Suggested value*: 0.9.

*Constraint*: $\mathbf{bl} > 0.0$.

13: **bd** – double *Input*

*On entry*: the magnitude of the bound for the diagonal elements of $S_k$, $BD$.

*Suggested value*: 0.9.

*Constraint*: **bd** $> 0.0$.

14: **maxit** – Integer *Input*

*On entry*: the maximum number of iterations that will be used during the calculation of $A$.

*Suggested value*: 150.

*Constraint*: **maxit** $> 0$.

15: **nitmon** – Integer *Input*

*On entry*: indicates the amount of information on the iteration that is printed.

**nitmon** $> 0$

The value of $A$, $\theta$ and $\delta$ (see Section 7) will be printed at the first and every **nitmon** iterations.

**nitmon** $\leq 0$

No iteration monitoring is printed.

16: **outfile** – const char * *Input*

*On entry*: a null terminated character string giving the name of the file to which results should be printed. If **outfile** = **NULL** or an empty string then the `stdout` stream is used. Note that the file will be opened in the append mode.

17: **tol** – double *Input*

*On entry*: the relative precision for the final estimates of the covariance matrix. Iteration will stop when maximum $\delta$ (see Section 7) is less than **tol**.

*Constraint*: **tol** $> 0.0$.

18: **nit** – Integer * *Output*

*On exit*: the number of iterations performed.

19: **comm** – Nag_Comm * *Communication Structure*

The NAG communication argument (see Section 2.2.1.1 of the Essential Introduction).

20: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 2.6 of the Essential Introduction).

## 6 Error Indicators and Warnings

**NE_ALLOC_FAIL**

Dynamic memory allocation failed.

**NE_BAD_PARAM**

On entry, argument $\langle value \rangle$ had an illegal value.

**NE_CONST_COL**

Column $\langle value \rangle$ of **x** has constant value.

**NE_CONVERGENCE**

Iterations to calculate weights failed to converge.

**NE_FUN_RET_VAL**

$u$ value returned by **ucv** $< 0.0$: $u(\langle value \rangle) = \langle value \rangle$.

$w$ value returned by **ucv** $< 0.0$: $w(\langle value \rangle) = \langle value \rangle$.

**NE_INT**

On entry, **m** $= \langle value \rangle$.
Constraint: **m** $\geq 1$.

On entry, **maxit** $= \langle value \rangle$.
Constraint: **maxit** $> 0$.

On entry, **n** $= \langle value \rangle$.
Constraint: **n** $> 1$.

On entry, **pdx** $= \langle value \rangle$.
Constraint: **pdx** $> 0$.

**NE_INT_2**

On entry, **m** $= \langle value \rangle$, **n** $= \langle value \rangle$.
Constraint: $1 \leq$ **m** $\leq$ **n**.

On entry, **n** $= \langle value \rangle$, **m** $= \langle value \rangle$.
Constraint: **n** $\geq$ **m**.

On entry, **pdx** $= \langle value \rangle$, **m** $= \langle value \rangle$.
Constraint: **pdx** $\geq$ **m**.

On entry, **pdx** $= \langle value \rangle$, **n** $= \langle value \rangle$.
Constraint: **pdx** $\geq$ **n**.

**NE_INTERNAL_ERROR**

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please consult NAG for assistance.

**NE_NOT_CLOSE_FILE**

Cannot close file $\langle value \rangle$.

**NE_NOT_WRITE_FILE**

Cannot open file $\langle value \rangle$ for writing.

**NE_REAL**

On entry, **bd** $= \langle value \rangle$.
Constraint: **bd** $> 0.0$.

On entry, **bl** $= \langle value \rangle$.
Constraint: **bl** $> 0.0$.

On entry, **tol** $= \langle value \rangle$.
Constraint: **tol** $> 0.0$.

**NE_ZERO_DIAGONAL**

On entry, diagonal element $\langle value \rangle$ of **a** is 0.0.

**NE_ZERO_SUM**

The sum $D1$ is zero.

The sum $D2$ is zero.

The sum $D3$ is zero.

## 7 Accuracy

On successful exit the accuracy of the results is related to the value of **tol**; see Section 5. At an iteration let

(i) $d1 =$ the maximum value of $|s_{jl}|$

(ii) $d2 =$ the maximum absolute change in $wt(i)$

(iii) $d3 =$ the maximum absolute relative change in $\theta_j$

and let $\delta = \max(d1, d2, d3)$. Then the iterative procedure is assumed to have converged when $\delta <$ **tol**.

## 8 Further Comments

The existence of $A$ will depend upon the function $u$ (see Marazzi (1987a)); 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

A sample of 10 observations on three variables is read in along with initial values for $A$ and **theta** and argument values for the $u$ and $w$ functions, $c_u$ and $c_w$. The covariance matrix computed by nag_robust_m_corr_user_fn (g02hlc) is printed along with the robust estimate of $\theta$. The function **ucv** computes the Huber's weight functions:

$$u(t) = 1, \quad \text{if} \quad t \leq c_u^2$$

$$u(t) = \frac{c_u}{t^2}, \quad \text{if} \quad t > c_u^2$$

and

$$w(t) = 1, \quad \text{if} \quad t \leq c_w$$

$$w(t) = \frac{c_w}{t}, \quad \text{if} \quad t > c_w$$

and their derivatives.

### 9.1 Program Text

```
/* nag_robust_m_corr_user_fn (g02hlc) Example Program.
 *
 * Copyright 2002 Numerical Algorithms Group.
 *
 * Mark 7, 2002.
 * Mark 7b revised, 2004.
 */

#include <stdio.h>
#include <nag.h>
#include <nagg02.h>
#include <nag_stdlib.h>
static void ucv(double t, double *u, double *ud, double *w, double *wd,
                Nag_Comm *comm);

int main(void)
{
```

```
  /* Scalars */
  double bd, bl, tol;
  Integer exit_status, i__, indm, j, k, l1, l2, m, maxit, mm, n, nit, nitmon;
  Integer pdx;
  NagError fail;
  Nag_OrderType order;
  Nag_Comm comm;


  /* Arrays */
  double *a=0, *cov=0, *theta=0, *userp=0, *wt=0, *x=0;

#ifdef NAG_COLUMN_MAJOR
#define X(I,J) x[(J-1)*pdx + I - 1]
  order = Nag_ColMajor;
#else
#define X(I,J) x[(I-1)*pdx + J - 1]
  order = Nag_RowMajor;
#endif

  INIT_FAIL(fail);
  exit_status = 0;
  Vprintf("nag_robust_m_corr_user_fn (g02hlc) Example Program Results\n");

  /* Skip heading in data file */
  Vscanf("%*[^\n] ");

  /* Read in the dimensions of X */
  Vscanf("%ld%ld%*[^\n] ", &n, &m);

  /* Allocate memory */
  if ( !(a = NAG_ALLOC(m*(m+1)/2, double)) ||
       !(cov = NAG_ALLOC(m*(m+1)/2, double)) ||
       !(theta = NAG_ALLOC(m, double)) ||
       !(userp = NAG_ALLOC(2, double)) ||
       !(wt = NAG_ALLOC(n, double)) ||
       !(x = NAG_ALLOC(n * m, double)) )
    {
      Vprintf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }
#ifdef NAG_COLUMN_MAJOR
  pdx = n;
#else
  pdx = m;
#endif

  /* Read in the X matrix */
  for (i__ = 1; i__ <= n; ++i__)
    {
      for (j = 1; j <= m; ++j)
        Vscanf("%lf", &X(i__,j));
      Vscanf("%*[^\n] ");
    }
  /* Read in the initial value of A */
  mm = (m + 1) * m / 2;
  for (j = 1; j <= mm; ++j)
    Vscanf("%lf", &a[j - 1]);
  Vscanf("%*[^\n] ");

  /* Read in the initial value of theta */
  for (j = 1; j <= m; ++j)
    Vscanf("%lf", &theta[j - 1]);
  Vscanf("%*[^\n] ");

  /* Read in the values of the parameters of the ucv functions */
  Vscanf("%lf%lf%*[^\n] ", &userp[0], &userp[1]);
  /* Set the values of remaining parameters */
  indm = 1;
```

```
  bl = 0.9;
  bd = 0.9;
  maxit = 50;
  tol = 5e-5;
  /* Change nitmon to a positive value if monitoring information
   *         is required
   */
  nitmon = 0;
  comm.p = (void *)userp;

  /* nag_robust_m_corr_user_fn (g02hlc).
   * Calculates a robust estimation of a correlation matrix,
   * user-supplied weight function plus derivatives
   */
  nag_robust_m_corr_user_fn(order, ucv, indm, n, m, x, pdx, cov, a, wt,
                            theta, bl, bd, maxit, nitmon, 0, tol, &nit, &comm,
                            &fail);
  if (fail.code != NE_NOERROR)
    {
      Vprintf("Error from nag_robust_m_corr_user_fn (g02hlc).\n%s\n",
              fail.message);
      exit_status = 1;
      goto END;
    }

  Vprintf("\n");
  Vprintf("nag_robust_m_corr_user_fn (g02hlc) required %4ld "
          "iterations to converge\n\n", nit);
  Vprintf("Robust covariance matrix\n");
  l2 = 0;
  for (j = 1; j <= m; ++j)
    {
      l1 = l2 + 1;
      l2 += j;
      for (k = l1; k <= l2; ++k)
        Vprintf("%10.3f%s", cov[k - 1], k%6 == 0 || k == l2 ?"\n":" ");
    }
  Vprintf("\n");

  Vprintf("Robust estimates of theta\n");
  for (j = 1; j <= m; ++j)
    Vprintf(" %10.3f\n", theta[j - 1]);

 END:
  if (a) NAG_FREE(a);
  if (cov) NAG_FREE(cov);
  if (theta) NAG_FREE(theta);
  if (userp) NAG_FREE(userp);
  if (wt) NAG_FREE(wt);
  if (x) NAG_FREE(x);

  return exit_status;
}
static void ucv(double t, double *u, double *ud, double *w, double *wd,
                Nag_Comm *comm)
                {
                  double t2, cu, cw;

                  double *userp = (double *)comm->p;

                  /* Function Body */
                  cu = userp[0];
                  *u = 1.0;
                  *ud = 0.0;
                  if (t != 0.0)
                    {
                      t2 = t * t;
                      if (t2 > cu)
                        {
                          *u = cu / t2;
                          *ud = *u * -2.0 / t;
```

```
          }
      }
    /* w function and derivative */
    cw = userp[1];
    if (t > cw)
      {
        *w = cw / t;
        *wd = -(*w) / t;
      }
    else
      {
        *w = 1.0;
        *wd = 0.0;
      }
    return;
  }
```

## 9.2 Program Data

```
nag_robust_m_corr_user_fn (g02hlc) Example Program Data
    10    3                     : N  M
  3.4  6.9  12.2                : X1  X2  X3
  6.4  2.5  15.1
  4.9  5.5  14.2
  7.3  1.9  18.2
  8.8  3.6  11.7
  8.4  1.3  17.9
  5.3  3.1  15.0
  2.7  8.1   7.7
  6.1  3.0  21.9
  5.3  2.2  13.9                : End of X1 X2 and X3 values
  1.0 0.0 1.0 0.0 0.0 1.0       : A
  0.0 0.0 0.0                   : THETA
  4.0 2.0                       : CU CW
```

## 9.3 Program Results

```
nag_robust_m_corr_user_fn (g02hlc) Example Program Results

nag_robust_m_corr_user_fn (g02hlc) required   25 iterations to converge

Robust covariance matrix
     3.278
    -3.692      5.284
     4.739     -6.409     11.837

Robust estimates of theta
      5.700
      3.864
     14.704
```