

## nag\_regsn\_mult\_linear\_upd\_model (g02ddc)

### 1. Purpose

**nag\_regsn\_mult\_linear\_upd\_model (g02ddc)** calculates the regression parameters for a general linear regression model. It is intended to be called after `nag_regsn_mult_linear_addrem_obs (g02dcc)`, `nag_regsn_mult_linear_add_var (g02dec)` or `nag_regsn_mult_linear_delete_var (g02dfc)`.

### 2. Specification

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

void nag_regsn_mult_linear_upd_model(Integer n, Integer ip, double q[],
    Integer tdq, double *rss, double *df, double b[], double se[],
    double cov[], Boolean *svd, Integer *rank, double p[], double tol,
    NagError *fail)
```

### 3. Description

A general linear regression model fitted by `nag_regsn_mult_linear (g02dac)` may be adjusted by adding or deleting an observation using `nag_regsn_mult_linear_addrem_obs (g02dcc)`, adding a new independent variable using `nag_regsn_mult_linear_add_var (g02dec)` or deleting an existing independent variable using `nag_regsn_mult_linear_delete_var (g02dfc)`. These functions compute the vector  $c$  and the upper triangular matrix  $R$ . `nag_regsn_mult_linear_upd_model` takes these basic results and computes the regression coefficients,  $\hat{\beta}$ , their standard errors and their variance-covariance matrix.

If  $R$  is of full rank, then  $\hat{\beta}$  is the solution to:

$$R\hat{\beta} = c_1,$$

where  $c_1$  is the first  $p$  elements of  $c$ .

If  $R$  is not of full rank a solution is obtained by means of a singular value decomposition (SVD) of  $R$ ,

$$R = Q_* \begin{pmatrix} D & 0 \\ 0 & 0 \end{pmatrix} P^T$$

where  $D$  is a  $k$  by  $k$  diagonal matrix with non-zero diagonal elements,  $k$  being the rank of  $R$ , and  $Q_*$  and  $P$  are  $p$  by  $p$  orthogonal matrices. This gives the solution

$$\hat{\beta} = P_1 D^{-1} Q_{*1}^T c_1$$

$P_1$  being the first  $k$  columns of  $P$ , i.e.,  $P = (P_1 P_0)$  and  $Q_{*1}$  being the first  $k$  columns of  $Q_*$ .

Details of the SVD, are made available, in the form of the matrix  $P^*$ :

$$P^* = \begin{pmatrix} D^{-1} P_1^T \\ P_0^T \end{pmatrix}$$

This will be only one of the possible solutions. Other estimates may be obtained by applying constraints to the parameters. These solutions can be obtained by calling `nag_regsn_mult_linear_tran_model (g02dkc)` after calling `nag_regsn_mult_linear_upd_model`. Only certain linear combinations of the parameters will have unique estimates, these are known as estimable functions. These can be estimated using `nag_regsn_mult_linear_est_func (g02dnc)`.

The residual sum of squares required to calculate the standard errors and the variance-covariance matrix can either be input or can be calculated if additional information on  $c$  for the whole sample is provided.

## 4. Parameters

**n**

Input: number of observations.  
 Constraint:  $n \geq 1$ .

**ip**

Input: the number of terms in the regression model,  $p$ .  
 Constraint:  $ip \geq 1$ .

**q[n][tdq]**

Input: **q** must be the array **q** as output by nag\_regsn\_mult\_linear\_addrem\_obs (g02dcc), nag\_regsn\_mult\_linear\_add\_var (g02dec) or nag\_regsn\_mult\_linear\_delete\_var (g02dfc). If on entry  $rss \leq 0.0$  then all **n** elements of  $c$  are needed. This is provided by functions nag\_regsn\_mult\_linear\_add\_var (g02dec) or nag\_regsn\_mult\_linear\_delete\_var (g02dfc).

**tdq**

Input: **tdq** the last dimension of the array **q** as declared in the function from which nag\_regsn\_mult\_linear\_upd\_model is called.  
 Constraint:  $tdq \geq ip+1$ .

**rss**

Input: either the residual sum of squares or a value less than or equal to 0.0 to indicate that the residual sum of squares is to be calculated by the function.  
 Output: if  $rss \leq 0.0$  on entry, then on exit **rss** will contain the residual sum of squares as calculated by nag\_regsn\_mult\_linear\_upd\_model.

If **rss** was positive on entry, then it will be unchanged.

**df**

Output: the degrees of freedom associated with the residual sum of squares.

**b[ip]**

Output: the estimates of the  $p$  parameters,  $\hat{\beta}$ .

**se[ip]**

Output: the standard errors of the  $p$  parameters given in **b**.

**cov[ip\*(ip+1)/2]**

Output: the upper triangular part of the variance-covariance matrix of the  $p$  parameter estimates given in **b**. They are stored packed by column, i.e., the covariance between the parameter estimate given in  $\mathbf{b}[i]$  and the parameter estimate given in  $\mathbf{b}[j]$ ,  $j \geq i$ , is stored in  $\mathbf{cov}[j(j+1)/2 + i]$ , for  $i = 0, 1, \dots, ip - 1$  and  $j = i, i + 1, \dots, ip - 1$ .

**svd**

Output: if a singular value decomposition has been performed, then **svd** = **TRUE**, otherwise **svd** = **FALSE**.

**rank**

Output: the rank of the independent variables.

If **svd** = **FALSE**, then **rank** = **ip**.

If **svd** = **TRUE**, then **rank** is an estimate of the rank of the independent variables.

**rank** is calculated as the number of singular values greater than  $\mathbf{tol} \times$  (largest singular value).

It is possible for the singular value decomposition to be carried out but **rank** to be returned as **ip**.

**p[ip\*ip+2\*ip]**

Output: **p** contains details of the singular value decomposition if used.

If **svd** = **FALSE**, **p** is not referenced.

If **svd** = **TRUE**, the first **ip** elements of **p** will not be referenced, the next **ip** values contain the singular values. The following **ip\*ip** values contain the matrix  $P^*$  stored by rows.

**tol**

Input: the value of **tol** is used to decide if the independent variables are of full rank and, if not, what is the rank of the independent variables. The smaller the value of **tol** the stricter the criterion for selecting the singular value decomposition. If **tol** = 0.0, then the singular

value decomposition will never be used, this may cause run time errors or inaccuracies if the independent variables are not of full rank.

Suggested value: **tol** = 0.000001.

Constraint: **tol**  $\geq$  0.0.

#### fail

The NAG error parameter, see the Essential Introduction to the NAG C Library.

## 5. Error Indications and Warnings

### NE\_INT\_ARG\_LT

On entry, **n** must not be less than 1: **n** =  $\langle value \rangle$ .

On entry, **ip** must not be less than 1: **ip** =  $\langle value \rangle$ .

### NE\_2\_INT\_ARG\_LT

On entry **tdq** =  $\langle value \rangle$  while **ip** + 1 =  $\langle value \rangle$ . These parameters must satisfy **tdq**  $\geq$  **ip** + 1.

On entry, **n** =  $\langle value \rangle$  while **ip** =  $\langle value \rangle$ . These parameters must satisfy **n**  $\geq$  **ip**.

### NE\_DOF\_LE\_ZERO

The degrees of freedom for error are less than or equal to 0. In this case the estimates,  $\hat{\beta}$ , are returned but not the standard errors or covariances.

### NE\_SVD\_NOT\_CONV

The singular value decomposition has failed to converge.

See nag\_real\_svd (f02wec). This is an unlikely error exit.

### NE\_REAL\_ARG\_LT

On entry, **tol** must not be less than 0.0: **tol** =  $\langle value \rangle$ .

### NE\_ALLOC\_FAIL

Memory allocation failed.

## 6. Further Comments

### 6.1. Accuracy

The accuracy of the results will depend on the accuracy of the input *R* matrix, which may lose accuracy if a large number of observations or variables have been dropped.

### 6.2. References

Golub G H and Van Loan C F (1983) *Matrix Computations* Johns Hopkins University Press, Baltimore.

Hammarling S (1985) The Singular Value Decomposition in Multivariate Statistics *ACM Signum Newsletter* **20** (3) 2–25.

Searle S R (1971) *Linear Models* Wiley.

## 7. See Also

nag\_real\_svd (f02wec)

nag\_regsn\_mult\_linear (g02dac)

nag\_regsn\_mult\_linear\_addrem\_obs (g02dcc)

nag\_regsn\_mult\_linear\_add\_var (g02dec)

nag\_regsn\_mult\_linear\_delete\_var (g02dfc)

nag\_regsn\_mult\_linear\_tran\_model (g02dkc)

nag\_regsn\_mult\_linear\_est\_func (g02dnc)

## 8. Example

A data set consisting of 12 observations and four independent variables is input and a regression model fitted by calls to nag\_regsn\_mult\_linear\_add\_var (g02dec). The parameters are then calculated by nag\_regsn\_mult\_linear\_upd\_model and the results printed.

## 8.1. Program Text

```

/* nag_regsn_mult_linear_upd_model(g02ddc) Example Program
 *
 * Copyright 1991 Numerical Algorithms Group.
 *
 * Mark 2, 1991.
 */
#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nagg02.h>

#define NMAX 12
#define MMAX 5
#define TDX MMAX
#define TDQ MMAX+1

main()
{
    double  rss, tol;
    Integer i, ip, rank, j, m, n;
    double  df;
    Boolean svd;
    char    weight;
    double  b[MMAX], cov[MMAX*(MMAX+1)/2], p[MMAX*(MMAX+2)],
    q[NMAX][MMAX+1], se[MMAX], wt[NMAX], x[NMAX][MMAX], xe[NMAX];
    double  *wtptr;
    static  NagError fail;

    Vprintf("g02ddc Example Program Results\n");
    /* Skip heading in data file */
    Vscanf("%*[^\\n]");
    Vscanf("%ld %ld %c", &n, &m, &weight);
    if (weight=='w')
        wtptr = wt;
    else
        wtptr = (double *)0;

    if (n<=NMAX && m<MMAX)
    {
        if (wtptr)
        {
            for (i=0; i<n; i++)
            {
                for (j=0; j<m; j++)
                    Vscanf("%lf", &x[i][j]);
                Vscanf("%lf%lf", &q[i][0], &wt[i]);
            }
        }
        else
        {
            for (i=0; i<n; i++)
            {
                for (j=0; j<m; j++)
                    Vscanf("%lf", &x[i][j]);
                Vscanf("%lf", &q[i][0]);
            }
        }
        /* Set tolerance */
        tol = 0.000001e0;
        ip = 0;
        for (j=0; j<m; ++j)
        {
            /*
             * Fit model using g02dec
             */
            for (i=0; i<n; i++)
                xe[i] = x[i][j];
            g02dec(n, ip, (double *)q, (Integer)(TDQ), p, wtptr, xe, &rss,

```

```

        tol, &fail);
    if (fail.code==NE_NOERROR)
        ip += 1;
    else if (fail.code==NE_NVAR_NOT_IND)
        Vprintf(" * New variable not added * \n");
    else
    {
        Vprintf("%s\n", fail.message);
        exit(EXIT_FAILURE);
    }
}
rss = 0.0;
g02ddc(n, ip, (double *)q, (Integer)(TDQ), &rss, &df, b, se, cov, &svd,
        &rank, p, tol, NAGERR_DEFAULT);

Vprintf("\n");
if (svd)
    Vprintf("Model not of full rank\n\n");
Vprintf("Residual sum of squares = %12.4e\n", rss);
Vprintf("Degrees of freedom = %3.1f\n\n", df);
Vprintf("Variable   Parameter estimate   Standard error\n\n");
for (j=0; j<ip; j++)
    Vprintf("%6ld%20.4e%20.4e\n", j+1, b[j], se[j]);
Vprintf("\n");
}
else
{
    Vfprintf(stderr, "One or both of m and n are out of range:\n
m = %-3ld while n = %-3ld\n", m, n);
    exit(EXIT_FAILURE);
}
exit(EXIT_SUCCESS);
}

```

## 8.2. Program Data

```

g02ddc Example Program Data
 12 4 u
 1.0 0.0 0.0 0.0 33.63
 0.0 0.0 0.0 1.0 39.62
 0.0 1.0 0.0 0.0 38.18
 0.0 0.0 1.0 0.0 41.46
 0.0 0.0 0.0 1.0 38.02
 0.0 1.0 0.0 0.0 35.83
 0.0 0.0 0.0 1.0 35.99
 1.0 0.0 0.0 0.0 36.58
 0.0 0.0 1.0 0.0 42.92
 1.0 0.0 0.0 0.0 37.80
 0.0 0.0 1.0 0.0 40.43
 0.0 1.0 0.0 0.0 37.89

```

## 8.3. Program Results

```
g02ddc Example Program Results
```

```
Residual sum of squares = 2.2227e+01
Degrees of freedom = 8.0
```

Variable	Parameter estimate	Standard error
1	3.6003e+01	9.6235e-01
2	3.7300e+01	9.6235e-01
3	4.1603e+01	9.6235e-01
4	3.7877e+01	9.6235e-01