# NAG Library Function Document

# nag_regsn_mult_linear_addrem_obs (g02dcc)

## 1 Purpose

nag_regsn_mult_linear_addrem_obs (g02dcc) adds or deletes an observation from a general regression model fitted by nag_regsn_mult_linear (g02dac).

## 2 Specification

```
#include <nag.h>
#include <nagg02.h>
void nag_regsn_mult_linear_addrem_obs (Nag_UpdateObserv update,
     Nag_IncludeMean mean, Integer m, const Integer sx[], double q[],
     Integer tdq, Integer ip, const double x[], Integer nr, Integer tdx,
     Integer ix, double y, const double wt[], double *rss, NagError *fail)
```

## 3 Description

nag_regsn_mult_linear (g02dac) fits a general linear regression model to a dataset. You may wish to change the model by either adding or deleting an observation from the dataset. nag_regsn_mult_linear_addrem_obs (g02dcc) takes the results from nag_regsn_mult_linear (g02dac) and makes the required changes to the vector $c$ and the upper triangular matrix $R$ produced by nag_regsn_mult_linear (g02dac). The regression coefficients, standard errors and the variance-covariance matrix of the regression coefficients can be obtained from nag_regsn_mult_linear_upd_model (g02ddc) after all required changes to the dataset have been made.

nag_regsn_mult_linear (g02dac) performs a $QR$ decomposition on the (weighted) $X$ matrix of independent variables. To add a new observation to a model with $p$ arguments the upper triangular matrix $R$ and vector $c_1$, the first $p$ elements of $c$, are augmented by the new observation on independent variables in $x^{\mathrm{T}}$ and dependent variable $y$. Givens rotations are then used to restore the upper triangular form.

$$\begin{pmatrix} R & \\ : & \\ c_1 & \\ x & y \end{pmatrix} \longrightarrow \begin{pmatrix} R^* & c_1^* \\ y^* & 0 \end{pmatrix}$$

To delete an observation Givens rotations are applied to give:

$$\begin{pmatrix} R & c_1 \end{pmatrix} \longrightarrow \begin{pmatrix} R^* & c_1^* \\ x & y \end{pmatrix}$$

Note: only the $R$ and upper part of the $c$ are updated, the remainder of the $Q$ matrix is unchanged.

## 4 References

Golub G H and Van Loan C F (1996) *Matrix Computations* (3rd Edition) Johns Hopkins University Press, Baltimore

Hammarling S (1985) The singular value decomposition in multivariate statistics *SIGNUM Newsl.* **20(3)** 2–25

# 5 Arguments

1: **update** – Nag_UpdateObserv *Input*

*On entry*: indicates if an observation is to be added or deleted.

**update** = Nag_ObservAdd
The observation is added.

**update** = Nag_ObservDel
The observation is deleted.

*Constraint*: **update** = Nag_ObservAdd or Nag_ObservDel.

2: **mean** – Nag_IncludeMean *Input*

*On entry*: indicates if a mean has been used in the model.

**mean** = Nag_MeanInclude
A mean term or intercept will have been included in the model by nag_regsn_mult_linear (g02dac).

**mean** = Nag_MeanZero
A model with no mean term or intercept will have been fitted by nag_regsn_mult_linear (g02dac).

*Constraint*: **mean** = Nag_MeanInclude or Nag_MeanZero.

3: **m** – Integer *Input*

*On entry*: the total number of independent variables in the dataset.

*Constraint*: $\mathbf{m} \geq 1$.

4: **sx**[**m**] – const Integer *Input*

*On entry*: if **sx**[$j$] is greater than 0, then the value contained in $\mathbf{x}[\mathbf{tdx} \times (\mathbf{ix} - 1) + j]$ is to be included as a value of $x^{\mathrm{T}}$, an observation on an independent variable, for $j = 0, 1, \ldots, m - 1$.

*Constraint*: if **mean** = Nag_MeanInclude, then exactly $\mathbf{ip} - 1$ elements of **sx** must be $> 0$ and if **mean** = Nag_MeanZero, then exactly **ip** elements of **sx** must be $> 0$.

5: **q**[**ip** × **tdq**] – double *Input/Output*

**Note**: the $(i, j)$th element of the matrix $Q$ is stored in $\mathbf{q}[(i - 1) \times \mathbf{tdq} + j - 1]$.

*On entry*: **q** must be array **q** as output by nag_regsn_mult_linear (g02dac), nag_regsn_mult_linear_add_var (g02dec), nag_regsn_mult_linear_delete_var (g02dfc), or a previous call to nag_regsn_mult_linear_addrem_obs (g02dcc).

*On exit*: the first **ip** elements of the first column of **q** will contain $c_1^*$, the upper triangular part of columns 2 to $\mathbf{ip} + 1$ will contain $R^*$, the remainder is unchanged.

6: **tdq** – Integer *Input*

*On entry*: the stride separating matrix column elements in the array **q**.

*Constraint*: $\mathbf{tdq} \geq \mathbf{ip} + 1$.

7: **ip** – Integer *Input*

*On entry*: the number of linear terms in general linear regression model (including mean if there is one).

*Constraint*: $\mathbf{ip} \geq 1$.

8: **x**[**nr** × **tdx**] – const double *Input*

*On entry*: the **ip** values for the dependent variables of the observation to be added or deleted, $x^{\mathrm{T}}$. The positions of the values **x** extracted depends on **ix** and **tdx**.

9: **nr** – Integer *Input*

*On entry*: the number of rows of the notional two-dimensional array **x**.

*Constraint*: **nr** $\geq 1$.

10: **tdx** – Integer *Input*

*On entry*: the stride separating matrix column elements in the array **x**.

*Constraint*: **tdx** $\geq$ **m**.

11: **ix** – Integer *Input*

*On entry*: the row of the notional two-dimensional array **x** that contains the values for the dependent variables of the observation to be added or deleted.

*Constraint*: $1 \leq$ **ix** $\leq nr$.

12: **y** – double *Input*

*On entry*: the value of the dependent variable for the observation to be added or deleted, $y$.

13: **wt**[**1**] – const double *Input*

*On entry*: if the new observation is to be weighted, then **wt** must contain the weight to be used with the new observation. If **wt**[0] $= 0.0$, then the observation is not included in the model. If the new observation is to be unweighted, then **wt** must be supplied as **NULL**.

*Constraint*: if the new observation is to be weighted **wt**[0] $\geq 0.0$.

14: **rss** – double * *Input/Output*

*On entry*: the value of the residual sums of squares for the original set of observations.

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

*On exit*: the updated values of the residual sums of squares.

**Note**: this will only be valid if the model is of full rank.

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

The NAG error argument (see Section 3.6 in the Essential Introduction).

## 6 Error Indicators and Warnings

**NE_2_INT_ARG_GT**

On entry, **ix** $= \langle value \rangle$ while **nr** $= \langle value \rangle$. These arguments must satisfy **ix** $\leq$ **nr**.

**NE_2_INT_ARG_LT**

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

On entry, **tdx** $= \langle value \rangle$ while **m** $= \langle value \rangle$. These arguments must satisfy **tdx** $\geq$ **m**.

**NE_ALLOC_FAIL**

Dynamic memory allocation failed.

**NE_BAD_PARAM**

On entry, **mean** had an illegal value.

On entry, **update** had an illegal value.

**NE_INT_ARG_LT**

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

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

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

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

**NE_IP_INCOMP_WITH_SX**

On entry, for **mean** = Nag_MeanInclude, number of nonzero values of **sx** must be equal to **ip** $- 1$: number of nonzero values of **sx** $= \langle value \rangle$, **ip** $- 1 = \langle value \rangle$.

On entry, for **mean** = Nag_MeanZero, number of nonzero values of **sx** must be equal to **ip**: number of nonzero values of **sx** $= \langle value \rangle$, **ip** $= \langle value \rangle$.

**NE_MAT_NOT_UPD**

The $R$ matrix could not be updated: to, either, delete nonexistent observation, or, add an observation to $R$ matrix with zero diagonal element.

**NE_REAL_ARG_LT**

On entry, **rss** $= \langle value \rangle$.
Constraint: **rss** $\geq 0.0$.

On entry, **wt**$[0] = \langle value \rangle$
Constraint: **wt**$[0] \geq 0.0$.

**NE_RSS_NOT_UPD**

The **rss** could not be updated because the input **rss** was less than the calculated decrease in **rss** when the new observation was deleted.

## 7 Accuracy

Higher accuracy is achieved by updating the $R$ matrix rather than the traditional methods of updating X'X.

## 8 Parallelism and Performance

Not applicable.

## 9 Further Comments

Care should be taken with the use of this function.

(a) It is possible to delete observations which were not included in the original model.

(b) If several additions/deletions have been performed you are advised to recompute the regression using nag_regsn_mult_linear (g02dac).

(c) Adding or deleting observations can alter the rank of the model. Such changes will only be detected when a call to nag_regsn_mult_linear_upd_model (g02ddc) has been made. nag_regsn_mult_linear_upd_model (g02ddc) should also be used to compute the new residual sum of squares when the model is not of full rank.

nag_regsn_mult_linear_addrem_obs (g02dcc) may also be used after nag_regsn_mult_linear_add_var (g02dec) and nag_regsn_mult_linear_delete_var (g02dfc).

# 10 Example

A dataset consisting of 12 observations with four independent variables is read in and a general linear regression model fitted by nag_regsn_mult_linear (g02dac) and parameter estimates printed. The last observation is then dropped and the parameter estimates recalculated, using nag_regsn_mult_linear_upd_model (g02ddc), and printed.

## 10.1 Program Text

```
/* nag_regsn_mult_linear_addrem_obs (g02dcc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 2, 1991.
 * Mark 8 revised, 2004.
 */

#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nagg02.h>

#define XM(I, J) xm[(I) *tdxm + J]
#define Q(I, J)  q[(I) *tdq + J]

int main(void)
{
  Integer          exit_status = 0, i, ip, j, m, n, rank, *sx = 0, tdq, tdxm;
  double           *b = 0, *com_ar = 0, *cov = 0, df, *h = 0, *p = 0, *q = 0;
  double           *res = 0, rss, *se = 0, tol, *wt = 0, *wtptr, *xm = 0;
  double           *y = 0;
  char             nag_enum_arg[40];
  Nag_Boolean      svd, weight;
  Nag_IncludeMean  mean;
  Nag_UpdateObserv update;
  NagError         fail;

  INIT_FAIL(fail);

  printf("nag_regsn_mult_linear_addrem_obs (g02dcc) Example Program "
         "Results\n");
  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif
#ifdef _WIN32
  scanf_s("%"NAG_IFMT" %"NAG_IFMT"", &n, &m);
#else
  scanf("%"NAG_IFMT" %"NAG_IFMT"", &n, &m);
#endif
#ifdef _WIN32
  scanf_s(" %39s", nag_enum_arg, _countof(nag_enum_arg));
#else
  scanf(" %39s", nag_enum_arg);
#endif
  /* nag_enum_name_to_value (x04nac).
   * Converts NAG enum member name to value
   */
```

```
  weight = (Nag_Boolean) nag_enum_name_to_value(nag_enum_arg);
#ifdef _WIN32
  scanf_s(" %39s", nag_enum_arg, _countof(nag_enum_arg));
#else
  scanf(" %39s", nag_enum_arg);
#endif
  mean = (Nag_IncludeMean) nag_enum_name_to_value(nag_enum_arg);

  if (weight)
    wtptr = wt;
  else
    wtptr = (double *) 0;

  if (n >= 2 && m >= 1)
    {
      if (!(b = NAG_ALLOC(m, double)) ||
          !(h = NAG_ALLOC(n, double)) ||
          !(res = NAG_ALLOC(n, double)) ||
          !(wt = NAG_ALLOC(n, double)) ||
          !(xm = NAG_ALLOC(n*m, double)) ||
          !(y = NAG_ALLOC(n, double)) ||
          !(sx = NAG_ALLOC(m, Integer)))
        {
          printf("Allocation failure\n");
          exit_status = -1;
          goto END;
        }
      tdxm = m;
    }
  else
    {
      printf("Invalid n or m.\n");
      exit_status = 1;
      return exit_status;
    }
  if (wtptr)
    {
      for (i = 0; i < n; i++)
        {
          for (j = 0; j < m; j++)
#ifdef _WIN32
            scanf_s("%lf", &XM(i, j));
#else
            scanf("%lf", &XM(i, j));
#endif
#ifdef _WIN32
          scanf_s("%lf%lf", &y[i], &wt[i]);
#else
          scanf("%lf%lf", &y[i], &wt[i]);
#endif
        }
    }
  else
    {
      for (i = 0; i < n; i++)
        {
          for (j = 0; j < m; j++)
#ifdef _WIN32
            scanf_s("%lf", &XM(i, j));
#else
            scanf("%lf", &XM(i, j));
#endif
#ifdef _WIN32
          scanf_s("%lf", &y[i]);
#else
          scanf("%lf", &y[i]);
#endif
        }
    }
  for (j = 0; j < m; ++j)
#ifdef _WIN32
```

```
    scanf_s("%"NAG_IFMT"", &sx[j]);
#else
    scanf("%"NAG_IFMT"", &sx[j]);
#endif
#ifdef _WIN32
  scanf_s("%"NAG_IFMT"", &ip);
#else
  scanf("%"NAG_IFMT"", &ip);
#endif

  if (!(cov = NAG_ALLOC(ip*(ip+1)/2, double)) ||
      !(p = NAG_ALLOC(ip*(ip+2), double)) ||
      !(q = NAG_ALLOC((n)*(ip+1), double)) ||
      !(com_ar = NAG_ALLOC(5*(ip-1)+ip*ip, double)) ||
      !(se = NAG_ALLOC(ip, double)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }
  tdq = ip+1;
  /* Set tolerance */
  tol = 0.00001e0;

  /* Fit initial model using nag_regsn_mult_linear (g02dac) */
  /* nag_regsn_mult_linear (g02dac).
   * Fits a general (multiple) linear regression model
   */
  nag_regsn_mult_linear(mean, n, xm, tdxm, m, sx, ip, y, wtptr, &rss,
                        &df, b, se, cov, res, h, q, tdq, &svd, &rank,
                        p, tol, com_ar, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_regsn_mult_linear (g02dac).\n%s\n",
              fail.message);
      exit_status = 1;
      goto END;
    }

  printf("Results from g02dac\n\n");
  if (svd)
    printf("Model not of full rank\n");
  printf("Residual sum of squares = %13.4e\n", rss);
  printf("Degrees of freedom = %3.1f\n\n", df);
  printf("Variable    Parameter estimate   Standard error\n\n");
  for (j = 0; j < ip; j++)
    printf("%6"NAG_IFMT"%20.4e%20.4e\n", j+1, b[j], se[j]);
  printf("\n");
  update = Nag_ObservDel;
  /* nag_regsn_mult_linear_addrem_obs (g02dcc).
   * Add/delete an observation to/from a general linear
   * regression model
   */
  nag_regsn_mult_linear_addrem_obs(update, mean, m, sx, q, tdq, ip,
                                   xm, n, tdxm, n, y[11], wtptr, &rss, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf(
              "Error from nag_regsn_mult_linear_addrem_obs (g02dcc).\n%s\n",
              fail.message);
      exit_status = 1;
      goto END;
    }

  printf("Results from dropping an observation using\n"
         "nag_regsn_mult_linear_addrem_obs (g02dcc)\n");
  n = n - 1;
  /* nag_regsn_mult_linear_upd_model (g02ddc).
   * Estimates of regression parameters from an updated model
   */
  nag_regsn_mult_linear_upd_model(n, ip, q, tdq, &rss, &df, b, se, cov,
```

```
                                    &svd, &rank, p, tol, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf(
              "Error from nag_regsn_mult_linear_upd_model (g02ddc).\n%s\n",
              fail.message);
      exit_status = 1;
      goto END;
    }


  printf("Residual sum of squares = %13.4e\n", rss);
  printf("Degrees of freedom = %3.1f\n\n", df);
  printf("Variable    Parameter estimate   Standard error\n\n");
  for (j = 0; j < ip; j++)
    printf("%6"NAG_IFMT"%20.4e%20.4e\n", j+1, b[j], se[j]);
 END:
  NAG_FREE(b);
  NAG_FREE(h);
  NAG_FREE(res);
  NAG_FREE(wt);
  NAG_FREE(xm);
  NAG_FREE(y);
  NAG_FREE(sx);
  NAG_FREE(cov);
  NAG_FREE(p);
  NAG_FREE(q);
  NAG_FREE(com_ar);
  NAG_FREE(se);
  return exit_status;
}
```

## 10.2 Program Data

```
nag_regsn_mult_linear_addrem_obs (g02dcc) Example Program Data
 12 4 Nag_FALSE Nag_MeanZero
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
1.0 1.0 1.0 1.0 37.89
 1   1   1   1   4
```

## 10.3 Program Results

```
nag_regsn_mult_linear_addrem_obs (g02dcc) Example Program Results
Results from g02dac

Residual sum of squares =   5.2748e+03
Degrees of freedom = 8.0

Variable    Parameter estimate   Standard error

     1          2.0724e+01          1.3801e+01
     2          1.4085e+01          1.6240e+01
     3          2.6324e+01          1.3801e+01
     4          2.2597e+01          1.3801e+01

Results from dropping an observation using
nag_regsn_mult_linear_addrem_obs (g02dcc)
Residual sum of squares =   2.1705e+01
Degrees of freedom = 7.0
```

| Variable | Parameter estimate | Standard error |
|---|---|---|
| 1 | 3.6003e+01 | 1.0166e+00 |
| 2 | 3.7005e+01 | 1.2451e+00 |
| 3 | 4.1603e+01 | 1.0166e+00 |
| 4 | 3.7877e+01 | 1.0166e+00 |