# NAG Library Function Document

# nag_lars_xtx (g02mbc)

## 1 Purpose

nag_lars_xtx (g02mbc) performs Least Angle Regression (LARS), forward stagewise linear regression or Least Absolute Shrinkage and Selection Operator (LASSO) using cross-product matrices.

## 2 Specification

```
#include <nag.h>
#include <nagg02.h>
void nag_lars_xtx (Nag_LARSModelType mtype, Nag_LARSPreProcess pred,
     Nag_LARSPreProcess intcpt, Integer n, Integer m, const double dtd[],
     Integer pddtd, const Integer isx[], const double dty[], double yty,
     Integer mnstep, Integer *ip, Integer *nstep, double b[], Integer pdb,
     double fitsum[], const double ropt[], Integer lropt, NagError *fail)
```

## 3 Description

nag_lars_xtx (g02mbc) implements the LARS algorithm of Efron *et al.* (2004) as well as the modifications needed to perform forward stagewise linear regression and fit LASSO and positive LASSO models.

Given a vector of $n$ observed values, $y = \{y_i : i = 1, 2, \ldots, n\}$ and an $n \times p$ design matrix $X$, where the $j$th column of $X$, denoted $x_j$, is a vector of length $n$ representing the $j$th independent variable $x_j$, standardized such that $\sum_{i=1}^{n} x_{ij} = 0$, and $\sum_{i=1}^{n} x_{ij}^2 = 1$ and a set of model parameters $\beta$ to be estimated from the observed values, the LARS algorithm can be summarised as:

1. Set $k = 1$ and all coefficients to zero, that is $\beta = 0$.
2. Find the variable most correlated with $y$, say $x_{j_1}$. Add $x_{j_1}$ to the 'most correlated' set $\mathcal{A}$. If $p = 1$ go to 8.
3. Take the largest possible step in the direction of $x_{j_1}$ (i.e., increase the magnitude of $\beta_{j_1}$) until some other variable, say $x_{j_2}$, has the same correlation with the current residual, $y - x_{j_1}\beta_{j_1}$.
4. Increment $k$ and add $x_{j_k}$ to $\mathcal{A}$.
5. If $|\mathcal{A}| = p$ go to 8.
6. Proceed in the 'least angle direction', that is, the direction which is equiangular between all variables in $\mathcal{A}$, altering the magnitude of the parameter estimates of those variables in $\mathcal{A}$, until the $k$th variable, $x_{j_k}$, has the same correlation with the current residual.
7. Go to 4.
8. Let $K = k$.

As well as being a model selection process in its own right, with a small number of modifications the LARS algorithm can be used to fit the LASSO model of Tibshirani (1996), a positive LASSO model, where the independent variables enter the model in their defined direction, forward stagewise linear regression (Hastie *et al.* (2001)) and forward selection (Weisberg (1985)). Details of the required modifications in each of these cases are given in Efron *et al.* (2004).

The LASSO model of Tibshirani (1996) is given by

$$\underset{\alpha,\beta_k \in \mathbb{R}^p}{\text{minimize}} \left\| y - \alpha - X^{\mathrm{T}}\beta_k \right\|^2 \quad \text{subject to} \quad \|\beta_k\|_1 \leq t_k$$

for all values of $t_k$, where $\alpha = \bar{y} = n^{-1}\sum_{i=1}^{n} y_i$. The positive LASSO model is the same as the standard LASSO model, given above, with the added constraint that

$$\beta_{kj} \geq 0, \quad j = 1, 2, \ldots, p.$$

Unlike the standard LARS algorithm, when fitting either of the LASSO models, variables can be dropped as well as added to the set $\mathcal{A}$. Therefore the total number of steps $K$ is no longer bounded by $p$.

Forward stagewise linear regression is an iterative procedure of the form:

1. Initialize $k = 1$ and the vector of residuals $r_0 = y - \alpha$.
2. For each $j = 1, 2, \ldots, p$ calculate $c_j = x_j^{\mathrm{T}} r_{k-1}$. The value $c_j$ is therefore proportional to the correlation between the $j$th independent variable and the vector of previous residual values, $r_k$.
3. Calculate $j_k = \underset{j}{\operatorname{argmax}} \left|c_j\right|$, the value of $j$ with the largest absolute value of $c_j$.
4. If $\left|c_{j_k}\right| < \epsilon$ then go to 7.
5. Update the residual values, with
$$r_k = r_{k-1} + \delta \ \operatorname{sign}\left(c_{j_k}\right) x_{j_k}$$
where $\delta$ is a small constant and $\operatorname{sign}\left(c_{j_k}\right) = -1$ when $c_{j_k} < 0$ and 1 otherwise.
6. Increment $k$ and go to 2.
7. Set $K = k$.

If the largest possible step were to be taken, that is $\delta = \left|c_{j_k}\right|$ then forward stagewise linear regression reverts to the standard forward selection method as implemented in nag_step_regsn (g02eec).

The LARS procedure results in $K$ models, one for each step of the fitting process. In order to aid in choosing which is the most suitable Efron *et al.* (2004) introduced a $C_p$-type statistic given by

$$C_p^{(k)} = \frac{\left\|y - X^{\mathrm{T}}\beta_k\right\|^2}{\sigma^2} - n + 2\nu_k,$$

where $\nu_k$ is the approximate degrees of freedom for the $k$th step and

$$\sigma^2 = \frac{n - y^{\mathrm{T}} y}{\nu_K}.$$

One way of choosing a model is therefore to take the one with the smallest value of $C_p^{(k)}$.

## 4 References

Efron B, Hastie T, Johnstone I and Tibshirani R (2004) Least Angle Regression *The Annals of Statistics (Volume 32)* **2** 407–499

Hastie T, Tibshirani R and Friedman J (2001) *The Elements of Statistical Learning: Data Mining, Inference and Prediction* Springer (New York)

Tibshirani R (1996) Regression Shrinkage and Selection via the Lasso *Journal of the Royal Statistics Society, Series B (Methodological) (Volume 58)* **1** 267–288

Weisberg S (1985) *Applied Linear Regression* Wiley

# 5 Arguments

1: **mtype** – Nag_LARSModelType *Input*

*On entry*: indicates the type of model to fit.

**mtype** = Nag_LARS_LAR
LARS is performed.

**mtype** = Nag_LARS_ForwardStagewise
Forward linear stagewise regression is performed.

**mtype** = Nag_LARS_LASSO
LASSO model is fit.

**mtype** = Nag_LARS_PositiveLASSO
A positive LASSO model is fit.

*Constraint*: **mtype** = Nag_LARS_LAR, Nag_LARS_ForwardStagewise, Nag_LARS_LASSO or Nag_LARS_PositiveLASSO.

2: **pred** – Nag_LARSPreProcess *Input*

*On entry*: indicates the type of preprocessing to perform on the cross-products involving the independent variables, i.e., those supplied in **dtd** and **dty**.

**pred** = Nag_LARS_None
No preprocessing is performed.

**pred** = Nag_LARS_Normalized
Each independent variable is normalized, with the $j$th variable scaled by $1/\sqrt{x_j^{\mathrm{T}}x_j}$. The scaling factor used by variable $j$ is returned in $\mathbf{b}[\mathbf{nstep} \times \mathbf{pdb} + j - 1]$.

*Constraint*: **pred** = Nag_LARS_None or Nag_LARS_Normalized.

3: **intcpt** – Nag_LARSPreProcess *Input*

*On entry*: indicates the type of data preprocessing that was perform on the dependent variable, $y$, prior to calling this function.

**intcpt** = Nag_LARS_None
No preprocessing was performed.

**intcpt** = Nag_LARS_Centered
The dependent variable, $y$, was mean centred.

*Constraint*: **intcpt** = Nag_LARS_None or Nag_LARS_Centered.

4: **n** – Integer *Input*

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

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

5: **m** – Integer *Input*

*On entry*: $m$, the total number of independent variables.

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

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

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

$\mathbf{pddtd} \times (\mathbf{m}(\mathbf{m}+1)/2)$ when $\mathbf{pddtd} = 1$;
$\mathbf{pddtd} \times \mathbf{m}$ when.

*On entry*: $D^{\mathrm{T}}D$, the cross-product matrix, which along with **isx**, defines the design matrix cross-product $X^{\mathrm{T}}X$.

If **pddtd** $= 1$, **dtd**$[(i \times (i-1)/2 + j - 1) \times$ **pddtd**$]$ must contain the cross-product of the $i$th and $j$th variable, for $i = 1, 2, \ldots,$ **m** and $j = 1, 2, \ldots,$ **m**. That is the cross-product stacked by columns as returned by nag_sum_sqs (g02buc), for example.

Otherwise **dtd**$[(j-1) \times$ **pddtd** $+ i - 1]$ must contain the cross-product of the $i$th and $j$th variable, for $i = 1, 2, \ldots,$ **m** and $j = 1, 2, \ldots,$ **m**. It should be noted that, even though $D^{\mathrm{T}}D$ is symmetric, the full matrix must be supplied.

The matrix specified in **dtd** must be a valid cross-products matrix.

7: **pddtd** – Integer *Input*

*On entry*: the stride separating row elements in the two-dimensional data stored in the array **dtd**.

*Constraint*: **pddtd** $= 1$ or **pddtd** $\geq$ **m**.

8: **isx**[**m**] – const Integer *Input*

*On entry*: indicates which independent variables from **dtd** will be included in the design matrix, $X$.

If **isx** is **NULL**, all variables are included in the design matrix.

Otherwise,, for $j = 1, 2, \ldots,$ **m** when **isx**$[j-1]$ must be set as follows:

**isx**$[j-1] = 1$
To indicate that the $j$th variable, as supplied in **dtd**, is included in the design matrix;

**isx**$[j-1] = 0$
To indicate that the $j$th variable, as supplied in **dtd**, is not included in the design matrix;

and $p = \sum_{j=1}^{m}$ **isx**$[j-1]$.

*Constraint*: **isx**$[j-1] = 0$ or $1$ and at least one value of **isx**$[j-1] \neq 0$, for $j = 1, 2, \ldots,$ **m**.

9: **dty**[**m**] – const double *Input*

*On entry*: $D^{\mathrm{T}}y$, the cross-product between the dependent variable, $y$, and the independent variables $D$.

10: **yty** – double *Input*

*On entry*: $y^{\mathrm{T}}y$, the sums of squares of the dependent variable.

*Constraint*: **yty** $> 0.0$.

11: **mnstep** – Integer *Input*

*On entry*: the maximum number of steps to carry out in the model fitting process.

If **mtype** $=$ Nag_LARS_LAR, the maximum number of steps the algorithm will take is $\min(p, n)$ if **intcpt** $=$ Nag_LARS_None, otherwise $\min(p, n-1)$.

If **mtype** $=$ Nag_LARS_ForwardStagewise, the maximum number of steps the algorithm will take is likely to be several orders of magnitude more and is no longer bound by $p$ or $n$.

If **mtype** $=$ Nag_LARS_LASSO or Nag_LARS_PositiveLASSO, the maximum number of steps the algorithm will take lies somewhere between that of the LARS and forward linear stagewise regression, again it is no longer bound by $p$ or $n$.

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

12: **ip** – Integer * *Output*

*On exit*: $p$, number of parameter estimates.

If **isx** is **NULL**, $p =$ **m**, i.e., the number of variables in **dtd**.

Otherwise $p$ is the number of nonzero values in **isx**.

13: **nstep** – Integer * *Output*

*On exit*: $K$, the actual number of steps carried out in the model fitting process.

14: **b**$[dim]$ – double *Output*

**Note**: the dimension, *dim*, of the array **b** must be at least **pdb** $\times$ (**mnstep** $+ 1$).

*On exit*: $\beta$ the parameter estimates, with **b**$[(k-1) \times$ **pdb** $+ j - 1] = \beta_{kj}$, the parameter estimate for the $j$th variable, $j = 1, 2, \ldots, p$ at the $k$th step of the model fitting process, $k = 1, 2, \ldots,$ **nstep**.

By default, when **pred** = Nag_LARS_Normalized the parameter estimates are rescaled prior to being returned. If the parameter estimates are required on the normalized scale, then this can be overridden via **ropt**.

The values held in the remaining part of **b** depend on the type of preprocessing performed.

If **pred** = Nag_LARS_None **b**[**nstep** $\times$ **pdb** $+ j - 1$] $= 1$,

if **pred** = Nag_LARS_Normalized **b**[**nstep** $\times$ **pdb** $+ j - 1$] $= 1/\sqrt{x_j^{\mathrm{T}} x_j}$,

for $j = 1, 2, \ldots p$.

15: **pdb** – Integer *Input*

*On entry*: the stride separating row elements in the two-dimensional data stored in the array **b**.

*Constraint*: **pdb** $\geq p$, where $p$ is the number of parameter estimates as described in **ip**.

16: **fitsum**[**6** $\times$ (**mnstep** $+$ **1**)] – double *Output*

*On exit*: summaries of the model fitting process. When $k = 1, 2, \ldots,$ **nstep**

**fitsum**$[(k-1) \times 6]$
$\|\beta_k\|_1$, the sum of the absolute values of the parameter estimates for the $k$th step of the modelling fitting process. If **pred** = Nag_LARS_Normalized, the scaled parameter estimates are used in the summation.

**fitsum**$[(k-1) \times 6 + 1]$
$\mathrm{RSS}_k$, the residual sums of squares for the $k$th step, where $\mathrm{RSS}_k = \|y - X^{\mathrm{T}}\beta_k\|^2$.

**fitsum**$[(k-1) \times 6 + 2]$
$\nu_k$, approximate degrees of freedom for the $k$th step.

**fitsum**$[(k-1) \times 6 + 3]$
$C_p^{(k)}$, a $C_p$-type statistic for the $k$th step, where $C_p^{(k)} = \frac{\mathrm{RSS}_k}{\sigma^2} - n + 2\nu_k$.

**fitsum**$[(k-1) \times 6 + 4]$
$\hat{C}_k$, correlation between the residual at step $k-1$ and the most correlated variable not yet in the active set $\mathcal{A}$, where the residual at step 0 is $y$.

**fitsum**$[(k-1) \times 6 + 5]$
$\hat{\gamma}_k$, the step size used at step $k$.

In addition

**fitsum**[**nstep** $\times$ 6]
0.

**fitsum**[**nstep** $\times 6 + 1$]
$\mathrm{RSS}_0$, the residual sums of squares for the null model, where $\mathrm{RSS}_0 = y^{\mathrm{T}}y$.

**fitsum**[**nstep** $\times 6 + 2$]
$\nu_0$, the degrees of freedom for the null model, where $\nu_0 = 0$ if **intcpt** = Nag_LARS_None and $\nu_0 = 1$ otherwise.

**fitsum**[**nstep** $\times 6 + 3$]
$C_p^{(0)}$, a $C_p$-type statistic for the null model, where $C_p^{(0)} = \frac{\mathrm{RSS}_0}{\sigma^2} - n + 2\nu_0$.

**fitsum**[**nstep** $\times 6 + 4$]
$\sigma^2$, where $\sigma^2 = \frac{n - \mathrm{RSS}_K}{\nu_K}$ and $K =$ **nstep**.

Although the $C_p$ statistics described above are returned when **fail.code** = NW_LIMIT_REACHED they may not be meaningful due to the estimate $\sigma^2$ not being based on the saturated model.

17: **ropt**[**lropt**] – const double *Input*

*On entry*: optional parameters to control various aspects of the LARS algorithm.

The default value will be used for **ropt**$[i-1]$ if **lropt** $< i$, therefore setting **lropt** $= 0$ will use the default values for all optional arguments and **ropt** need not be set and may be **NULL**. The default value will also be used if an invalid value is supplied for a particular argument, for example, setting **ropt**$[i-1] = -1$ will use the default value for argument $i$.

**ropt**[0]
The minimum step size that will be taken.

Default is $100 \times eps$ is used, where $eps$ is the ***machine precision*** returned by nag_machine_precision (X02AJC).

**ropt**[1]
General tolerance, used amongst other things, for comparing correlations.

Default is **ropt**[0].

**ropt**[2]
If set to 1 then parameter estimates are rescaled before being returned. If set to 0 then no rescaling is performed. This argument has no effect when **pred** = Nag_LARS_None.

Default is for the parameter estimates to be rescaled.

*Constraints*:

**ropt**[0] $>$ ***machine precision***;
**ropt**[1] $>$ ***machine precision***.

18: **lropt** – Integer *Input*

*On entry*: length of the options array **ropt**.

*Constraint*: $0 \leq$ **lropt** $\leq 3$.

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

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

## 6 Error Indicators and Warnings

**NE_ALLOC_FAIL**

Dynamic memory allocation failed.
See Section 2.3.1.2 in How to Use the NAG Library and its Documentation for further information.

**NE_ARRAY_SIZE**

On entry, **lropt** $= \langle value \rangle$.
Constraint: $0 \leq$ **lropt** $\leq 3$.

On entry, **pdb** $= \langle value \rangle$ and **m** $= \langle value \rangle$.
Constraint: if **isx** is **NULL** then **pdb** $\geq$ **m**.

On entry, **pdb** $= \langle value \rangle$ and $p = \langle value \rangle$.
Constraint: if **isx** is not **NULL**, **pdb** $\geq p$.

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

**NE_BAD_PARAM**

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

**NE_INT**

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

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

**NE_INT_ARRAY**

On entry, all values of **isx** are zero.
Constraint: at least one value of **isx** must be nonzero.

On entry, **isx**$[\langle value \rangle] = \langle value \rangle$.
Constraint: **isx**$[i] = 0$ or $1$ for all $i$.

**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 contact NAG for assistance.

An unexpected error has been triggered by this function. Please contact NAG.
See Section 2.7.6 in How to Use the NAG Library and its Documentation for further information.

**NE_MAX_STEP**

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

**NE_NEG_ELEMENT**

On entry, **dtd**$[\langle value \rangle \times$ **pddtd**$] = \langle value \rangle$.
Constraint: diagonal elements of $D^{\mathrm{T}}D$ must be positive.

On entry, $i = \langle value \rangle$ and **dtd**$[(i-1) \times$ **pddtd** $+ i - 1] = \langle value \rangle$.
Constraint: diagonal elements of $D^{\mathrm{T}}D$ must be positive.

**NE_NEG_SX**

A negative value for the residual sums of squares was obtained. Check the values of **dtd**, **dty** and **yty**.

**NE_NO_LICENCE**

Your licence key may have expired or may not have been installed correctly.
See Section 2.7.5 in How to Use the NAG Library and its Documentation for further information.

**NE_REAL**

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

**NE_SYMM_MATRIX**

The cross-product matrix supplied in **dtd** is not symmetric.

**NW_LIMIT_REACHED**

Fitting process did not finished in **mnstep** steps. Try increasing the size of **mnstep** and supplying larger output arrays.
All output is returned as documented, up to step **mnstep**, however, $\sigma$ and the $C_p$ statistics may not be meaningful.

**NW_OVERFLOW_WARN**

$\nu_K = n$, therefore sigma has been set to a large value. Output is returned as documented.

$\sigma^2$ is approximately zero and hence the $C_p$-type criterion cannot be calculated. All other output is returned as documented.

**NW_POTENTIAL_PROBLEM**

Degenerate model, no variables added and **nstep** $= 0$. Output is returned as documented.

## 7 Accuracy

Not applicable.

## 8 Further Comments

The solution path to the LARS, LASSO and stagewise regression analysis is a continuous, piecewise linear. nag_lars_xtx (g02mbc) returns the parameter estimates at various points along this path. nag_lars_param (g02mcc) can be used to obtain estimates at different points along the path.

If you have the raw data values, that is $D$ and $y$, then nag_lars (g02mac) can be used instead of nag_lars_xtx (g02mbc).

## 9 Example

This example performs a LARS on a simulated dataset with 20 observations and 6 independent variables.

The example uses nag_sum_sqs (g02buc) to get the cross-products of the augmented matrix $[D\ y]$. The first $m(m+1)/2$ elements of the (column packed) cross-products matrix returned therefore contain the elements of $D^{\mathrm{T}}D$, the next $m$ elements contain $D^{\mathrm{T}}y$ and the last element $y^{\mathrm{T}}y$.

### 9.1 Program Text

```
/* nag_lars_xtx (g02mbc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */
/* Pre-processor includes */
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg02.h>
```

```
int main(void)
{
  /* Integer scalar and array declarations */
  Integer i, j, k, ip, ldb, lddtd, m, mnstep, n, nstep, lropt, pm, pm2, pddy;
  Integer *isx = 0;
  Integer exit_status = 0;

  /* NAG structures and types */
  NagError fail;
  Nag_LARSModelType mtype;
  Nag_LARSPreProcess intcpt, pred;
  Nag_SumSquare mean;

  /* Double scalar and array declarations */
  double sw;
  double *b = 0, *dtd = 0, *fitsum = 0, *dy = 0, *ropt = 0, *wmean = 0;

  /* Character scalar and array declarations */
  char cmtype[40], cpred[40], cintcpt[40];

  /* Initialize the error structure */
  INIT_FAIL(fail);

  printf("nag_lars_xtx (g02mbc) Example Program Results\n\n");

  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

  /* Read in the problem size */
#ifdef _WIN32
  scanf_s("%" NAG_IFMT "%" NAG_IFMT "%*[^\n] ", &n, &m);
#else
  scanf("%" NAG_IFMT "%" NAG_IFMT "%*[^\n] ", &n, &m);
#endif

  /* Read in the model specification */
#ifdef _WIN32
  scanf_s("%39s%39s%39s%" NAG_IFMT "%*[^\n] ", cmtype,
          (unsigned)_countof(cmtype), cpred,
          (unsigned)_countof(cpred), cintcpt,
          (unsigned)_countof(cintcpt), &mnstep);
#else
  scanf("%39s%39s%39s%" NAG_IFMT "%*[^\n] ", cmtype, cpred, cintcpt, &mnstep);
#endif
  mtype = (Nag_LARSModelType) nag_enum_name_to_value(cmtype);
  pred = (Nag_LARSPreProcess) nag_enum_name_to_value(cpred);
  intcpt = (Nag_LARSPreProcess) nag_enum_name_to_value(cintcpt);

  /* Using all variables */
  isx = 0;
  ip = m;

  /* Optional arguments (using defaults) */
  lropt = 0;
  ropt = 0;

  /* Allocate memory for the augmented matrix [D y] and its cross-product */
  pddy = n;
  if (!(dy = NAG_ALLOC(pddy * (m + 1), double)))
  {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }

  /* Read in the augmented matrix [D y] and calculate cross-product matrices
```

```
     (NB: Datasets with a large number of observations can be split into
     blocks with the resulting cross-product matrices being combined
     using g02bzc) */
  for (i = 0; i < n; i++) {
    for (j = 0; j < m + 1; j++) {
#ifdef _WIN32
      scanf_s("%lf", &dy[j * pddy + i]);
#else
      scanf("%lf", &dy[j * pddy + i]);
#endif
    }
  }
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

  pm = m * (m + 1) / 2;
  pm2 = (m + 1) * (m + 2) / 2 - 1;

  /* We are calculating the cross-product matrix using nag_sum_sqs (g02buc)
     which returns it in packed storage */
  lddtd = 1;
  if (!(wmean = NAG_ALLOC(m + 1, double)) ||
      !(dtd = NAG_ALLOC(pm2 + 1, double)))
  {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }

  mean = (intcpt == Nag_LARS_Centered) ? Nag_AboutMean : Nag_AboutZero;

  /* Calculate the cross-product matrices using g02buc */
  nag_sum_sqs(Nag_ColMajor, mean, n, m + 1, dy, pddy, NULL, &sw, wmean, dtd,
              NAGERR_DEFAULT);
  /* The first PM+1 elements of dtd contain the cross-products of D
     elements PM to PM2-1 contains cross-product of D with y and element PM2
     contains cross-product of y with itself */

  /* Allocate output arrays */
  ldb = ip;
  if (!(b = NAG_ALLOC(ldb * (mnstep + 1), double)) ||
      !(fitsum = NAG_ALLOC(6 * (mnstep + 1), double)))
  {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }

  /* Call g02mbc to the model */
  nag_lars_xtx(mtype, pred, intcpt, n, m, dtd, lddtd, isx, &dtd[pm], dtd[pm2],
               mnstep, &ip, &nstep, b, ldb, fitsum, ropt, lropt, &fail);
  if (fail.code != NE_NOERROR) {
    if (fail.code != NW_OVERFLOW_WARN && fail.code != NW_POTENTIAL_PROBLEM &&
        fail.code != NW_LIMIT_REACHED) {
      printf("Error from nag_lars_xtx (g02mbc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
    else {
      printf("Warning from nag_lars_xtx (g02mbc).\n%s\n", fail.message);
      exit_status = 2;
    }
  }

  /* Display the parameter estimates */
  printf("  Step ");
  for (i = 0; i < MAX(ip - 2, 0) * 5; i++)
    printf(" ");
```

```
  printf(" Parameter Estimate\n ");
  for (i = 0; i < 5 + ip * 10; i++)
    printf("-");
  printf("\n");
  for (k = 0; k < nstep; k++) {
    printf("  %3" NAG_IFMT "", k + 1);
    for (j = 0; j < ip; j++) {
      printf(" %9.3f", b[k * ldb + j]);
    }
    printf("\n");
  }
  printf("\n");
  printf(" alpha: %9.3f\n", wmean[m]);
  printf("\n");
  printf("  Step     Sum      RSS       df       Cp       Ck     Step Size\n ");
  for (i = 0; i < 64; i++)
    printf("-");
  printf("\n");
  for (k = 0; k < nstep; k++) {
    printf("  %3" NAG_IFMT " %9.3f %9.3f %6.0f  %9.3f %9.3f %9.3f\n",
           k + 1, fitsum[k * 6], fitsum[k * 6 + 1], fitsum[k * 6 + 2],
           fitsum[k * 6 + 3], fitsum[k * 6 + 4], fitsum[k * 6 + 5]);
  }
  printf("\n");
  printf(" sigma^2: %9.3f\n", fitsum[nstep * 6 + 4]);

END:
  NAG_FREE(dy);
  NAG_FREE(wmean);
  NAG_FREE(dtd);
  NAG_FREE(b);
  NAG_FREE(fitsum);
  NAG_FREE(ropt);

  return (exit_status);
}
```

## 9.2 Program Data

```
nag_lars_xtx (g02mbc) Example Program Data
20 6                          :: n,m
Nag_LARS_LAR Nag_LARS_Normalized
Nag_LARS_Centered  6          :: mtype,pred,intcpt,mnstep
10.28  1.77  9.69 15.58  8.23 10.44    -46.47
 9.08  8.99 11.53  6.57 15.89 12.58    -35.80
17.98 13.10  1.04 10.45 10.12 16.68   -129.22
14.82 13.79 12.23  7.00  8.14  7.79    -42.44
17.53  9.41  6.24  3.75 13.12 17.08    -73.51
 7.78 10.38  9.83  2.58 10.13  4.25    -26.61
11.95 21.71  8.83 11.00 12.59 10.52    -63.90
14.60 10.09 -2.70  9.89 14.67  6.49    -76.73
 3.63  9.07 12.59 14.09  9.06  8.19    -32.64
 6.35  9.79  9.40 12.79  8.38 16.79    -83.29
 4.66  3.55 16.82 13.83 21.39 13.88    -16.31
 8.32 14.04 17.17  7.93  7.39 -1.09     -5.82
10.86 13.68  5.75 10.44 10.36 10.06    -47.75
 4.76  4.92 17.83  2.90  7.58 11.97     18.38
 5.05 10.41  9.89  9.04  7.90 13.12    -54.71
 5.41  9.32  5.27 15.53  5.06 19.84    -55.62
 9.77  2.37  9.54 20.23  9.33  8.82    -45.28
14.28  4.34 14.23 14.95 18.16 11.03    -22.76
10.17  6.80  3.17  8.57 16.07 15.93   -104.32
 5.39  2.67  6.37 13.56 10.68  7.35    -55.94 :: End of d, y
```

## 9.3 Program Results

```
nag_lars_xtx (g02mbc) Example Program Results

 Step                        Parameter Estimate
-------------------------------------------------------------------
    1     0.000     0.000     3.125     0.000     0.000     0.000
    2     0.000     0.000     3.792     0.000     0.000    -0.713
    3    -0.446     0.000     3.998     0.000     0.000    -1.151
    4    -0.628    -0.295     4.098     0.000     0.000    -1.466
    5    -1.060    -1.056     4.110    -0.864     0.000    -1.948
    6    -1.073    -1.132     4.118    -0.935    -0.059    -1.981

alpha:   -50.037

 Step      Sum       RSS       df        Cp        Ck      Step Size
-------------------------------------------------------------------
    1    72.446  8929.855       2    13.355   123.227    72.446
    2   103.385  6404.701       3     7.054    50.781    24.841
    3   126.243  5258.247       4     5.286    30.836    16.225
    4   145.277  4657.051       5     5.309    19.319    11.587
    5   198.223  3959.401       6     5.016    12.266    24.520
    6   203.529  3954.571       7     7.000     0.910     2.198

sigma^2:   304.198
```

This example plot shows the regression coefficients ($\beta_k$) plotted against the scaled absolute sum of the parameter estimates ($\|\beta_k\|_1$).