NAG Library Function Document

nag_lars_param (g02mcc)

1 Purpose

nag_lars_param (g02mcc) calculates additional parameter estimates following Least Angle Regression (LARS), forward stagewise linear regression or Least Absolute Shrinkage and Selection Operator (LASSO) as performed by nag_lars (g02mac) and nag_lars_xtx (g02mbc).

2 Specification

```
#include <nag.h>
#include <nagg02.h>
void nag_lars_param (Integer nstep, Integer ip, const double b[],
     Integer pdb, const double fitsum[], Nag_LARSTargetType ktype,
     const double nk[], Integer lnk, double nb[], Integer pdnb,
     NagError *fail)
```

3 Description

nag_lars (g02mac) and nag_lars_xtx (g02mbc) fit either a LARS, forward stagewise linear regression, LASSO or positive LASSO model to a vector of n observed values, $y = \{y_i : i = 1, 2, \ldots, n\}$ and an $n \times p$ design matrix X, where the jth column of X is given by the jth independent variable x_j. The models are fit using the LARS algorithm of Efron *et al.* (2004).

The full solution path for all four of these models follow a similar pattern where the parameter estimate for a given variable is piecewise linear. One such path, for a LARS model with six variables $(p = 6)$ can be seen in Figure 1. Both nag_lars (g02mac) and nag_lars_xtx (g02mbc) return the vector of p parameter estimates, β_k, at K points along this path (so $k = 1, 2, \ldots, K$). Each point corresponds to a step of the LARS algorithm. The number of steps taken depends on the model being fitted. In the case of a LARS model, $K = p$ and each step corresponds to a new variable being included in the model. In the case of the LASSO models, each step corresponds to either a new variable being included in the model or an existing variable being removed from the model; the value of K is therefore no longer bound by the number of parameters. For forward stagewise linear regression, each step no longer corresponds to the addition or removal of a variable; therefore the number of possible steps is often markedly greater than for a corresponding LASSO model.

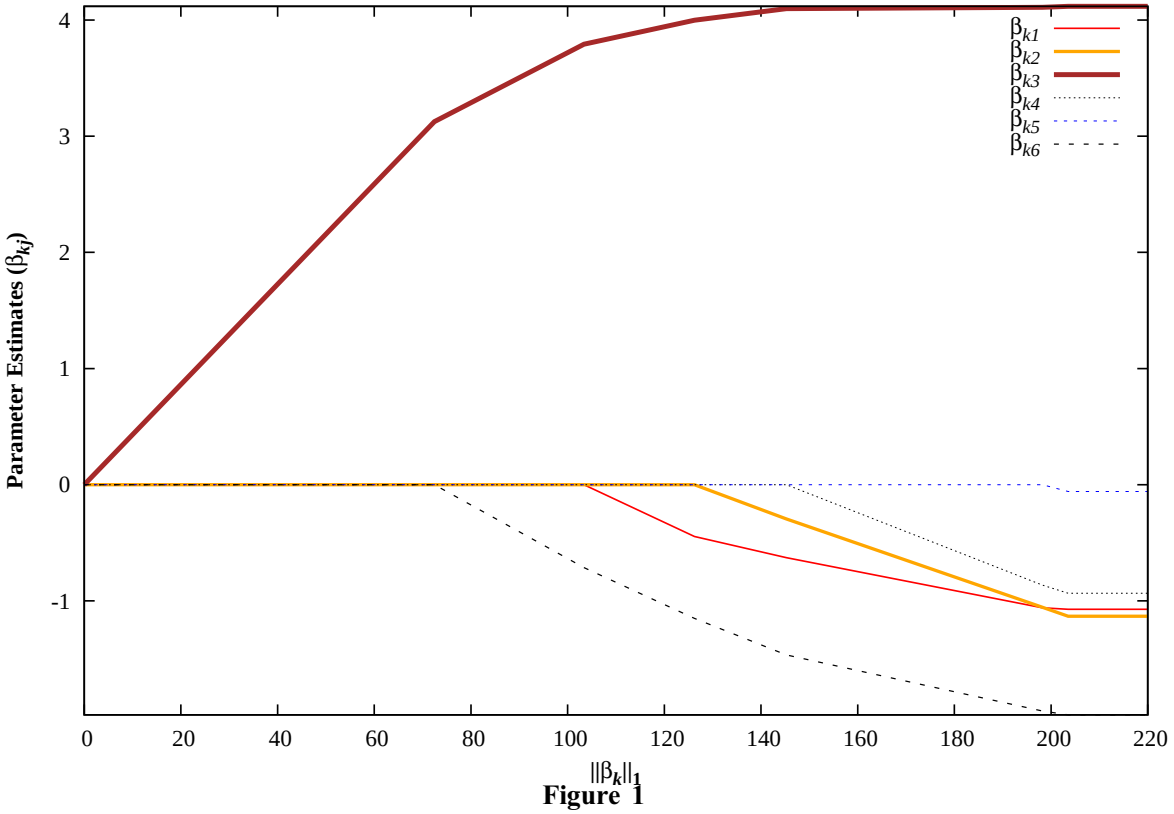

Figure 1

nag_lars_param (g02mcc) uses the piecewise linear nature of the solution path to predict the parameter estimates, $\tilde{\beta}$, at a different point on this path. The location of the solution can either be defined in terms of a (fractional) step number or a function of the L_1 norm of the parameter estimates.

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: **nstep** – Integer *Input*

On entry: K, the number of steps carried out in the model fitting process, as returned by nag_lars (g02mac) and nag_lars_xtx (g02mbc).

Constraint: **nstep** ≥ 0.

2: **ip** – Integer *Input*

On entry: p, number of parameter estimates, as returned by nag_lars (g02mac) and nag_lars_xtx (g02mbc).

Constraint: **ip** ≥ 1.

3: **b**[dim] – const double *Input*

Note: the dimension, *dim*, of the array **b** must be at least **pdb** × (**nstep** + 1).

On entry: β the parameter estimates, as returned by nag_lars (g02mac) and nag_lars_xtx (g02mbc), with **b**[$(k-1) \times$ **pdb** $+ j - 1] = \beta_{kj}$, the parameter estimate for the jth variable, for $j = 1, 2, \ldots, p$, at the kth step of the model fitting process.

Constraint: **b** should be unchanged since the last call to nag_lars (g02mac) or nag_lars_xtx (g02mbc).

4: **pdb** – Integer *Input*

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

Constraint: **pdb** $\geq$ **ip**.

5: **fitsum**[**6** × (**nstep** + **1**)] – const double *Input*

On entry: summaries of the model fitting process, as returned by nag_lars (g02mac) and nag_lars_xtx (g02mbc).

Constraint: **fitsum** should be unchanged since the last call to nag_lars (g02mac) or nag_lars_xtx (g02mbc)..

6: **ktype** – Nag_LARSTargetType *Input*

On entry: indicates what target values are held in **nk**.

ktype = Nag_LARS_StepNumber
nk holds (fractional) LARS step numbers.

ktype = Nag_LARS_ScaledNorm
nk holds values for L_1 norm of the (scaled) parameters.

ktype = Nag_LARS_ProportionScaledNorm
nk holds ratios with respect to the largest (scaled) L_1 norm.

ktype = Nag_LARS_UnscaledNorm
nk holds values for the L_1 norm of the (unscaled) parameters.

ktype = Nag_LARS_ProportionUnscaledNorm
nk holds ratios with respect to the largest (unscaled) L_1 norm.

If nag_lars (g02mac) was called with **pred** = Nag_LARS_None or Nag_LARS_Centered or nag_lars_xtx (g02mbc) was called with **pred** = Nag_LARS_None then the model fitting routine did not rescale the independent variables, X, prior to fitting the model and therefore there is no difference between **ktype** = Nag_LARS_ScaledNorm or Nag_LARS_ProportionScaledNorm and **ktype** = Nag_LARS_UnscaledNorm or Nag_LARS_ProportionUnscaledNorm.

Constraint: **ktype** = Nag_LARS_StepNumber, Nag_LARS_ScaledNorm, Nag_LARS_ProportionScaledNorm, Nag_LARS_UnscaledNorm or Nag_LARS_ProportionUnscaledNorm.

7: **nk**[**lnk**] – const double *Input*

On entry: target values used for predicting the new set of parameter estimates.

Constraints:

if **ktype** = Nag_LARS_StepNumber, $0 \leq$ **nk**$[i-1] \leq$ **nstep**, for $i = 1, 2, \ldots,$ **lnk**;
if **ktype** = Nag_LARS_ScaledNorm, $0 \leq$ **nk**$[i-1] \leq$ **fitsum**$[($**nstep** $- 1) \times 6]$, for $i = 1, 2, \ldots,$ **lnk**;
if **ktype** = Nag_LARS_ProportionScaledNorm or Nag_LARS_ProportionUnscaledNorm, $0 \leq$ **nk**$[i-1] \leq 1$, for $i = 1, 2, \ldots,$ **lnk**;
if **ktype** = Nag_LARS_UnscaledNorm, $0 \leq$ **nk**$[i-1] \leq \|\beta_K\|_1$, for $i = 1, 2, \ldots,$ **lnk**.

8: **lnk** – Integer *Input*

On entry: number of values supplied in **nk**.

Constraint: **lnk** ≥ 1.

9: **nb**[dim] – double *Output*

Note: the dimension, *dim*, of the array **nb** must be at least **pdnb** $\times$ **lnk**.

On exit: $\tilde{\beta}$ the predicted parameter estimates, with $\mathbf{b}[(i-1) \times \mathbf{pdb} + j - 1] = \tilde{\beta}_{ij}$, the parameter estimate for variable j, $j = 1, 2, \ldots, p$ at the point in the fitting process associated with $\mathbf{nk}[i-1]$, $i = 1, 2, \ldots,$ **lnk**.

10: **pdnb** – Integer *Input*

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

Constraint: **pdnb** $\geq$ **ip**.

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

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

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

NE_ARRAY_SIZE

On entry, **pdb** = ⟨*value*⟩ and **ip** = ⟨*value*⟩
Constraint: **pdb** $\geq$ **ip**.

On entry, **pdnb** = ⟨*value*⟩ and **ip** = ⟨*value*⟩.
Constraint: **pdnb** $\geq$ **ip**.

NE_BAD_PARAM

On entry, argument ⟨*value*⟩ had an illegal value.

NE_INT

On entry, **ip** = ⟨*value*⟩.
Constraint: **ip** ≥ 1.

On entry, **lnk** = ⟨*value*⟩.
Constraint: **lnk** ≥ 1.

On entry, **nstep** = ⟨*value*⟩.
Constraint: **nstep** ≥ 0.

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 3.6.6 in the Essential Introduction for further information.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
See Section 3.6.5 in the Essential Introduction for further information.

NE_OUT_OF_RANGE

On entry, **ktype** = Nag_LARS_ProportionScaledNorm or Nag_LARS_ProportionUnscaledNorm, **nk**[⟨*value*⟩] = ⟨*value*⟩.
Constraint: $0 \leq \mathbf{nk}[i] \leq 1$ for all i.

On entry, **ktype** = Nag_LARS_ScaledNorm, **nk**[⟨*value*⟩] = ⟨*value*⟩, **nstep** = ⟨*value*⟩ and **fitsum**[(**nstep** − 1) × 6] = ⟨*value*⟩.
Constraint: $0 \leq \mathbf{nk}[i] \leq \mathbf{fitsum}[(\mathbf{nstep} - 1) \times 6]$ for all i.

On entry, **ktype** = Nag_LARS_StepNumber, **nk**[⟨*value*⟩] = ⟨*value*⟩ and **nstep** = ⟨*value*⟩
Constraint: $0 \leq \mathbf{nk}[i] \leq \mathbf{nstep}$ for all i.

On entry, **ktype** = Nag_LARS_UnscaledNorm, **nk**[⟨*value*⟩] = ⟨*value*⟩ and $\|\beta_K\|_1$ = ⟨*value*⟩
Constraint: $0 \leq \mathbf{nk}[i] \leq \|\beta_K\|_1$ for all i.

NE_REAL_ARRAY

b has been corrupted since the last call to nag_lars (g02mac) or nag_lars_xtx (g02mbc).

fitsum has been corrupted since the last call to nag_lars (g02mac) or nag_lars_xtx (g02mbc).

7 Accuracy

Not applicable.

8 Further Comments

None.

9 Example

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

Additional parameter estimates are obtained corresponding to a LARS step number of $0.2, 1.2, 3.2, 4.5$ and 5.2. Where, for example, 4.5 corresponds to the solution halfway between that obtained at step 4 and that obtained at step 5.

9.1 Program Text

```
/* nag_lars_param (g02mcc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 25, 2014.
 */
/* 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, ldd, m, mnstep, n, nstep, lropt, ldnb, lnk;
  Integer *isx = 0;
  Integer exit_status = 0;

  /* NAG structures and types */
  NagError fail;
  Nag_LARSModelType mtype;
  Nag_LARSPreProcess pred, prey;
  Nag_LARSTargetType ktype;
```

```
  /* Double scalar and array declarations */
  double *b = 0, *d = 0, *fitsum = 0, *y = 0, *ropt = 0, *nb = 0, *nk = 0;

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

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

  printf("nag_lars (g02mac) 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, _countof(cmtype), cpred,
          _countof(cpred), cprey, _countof(cprey), &mnstep);
#else
  scanf("%39s%39s%39s%"NAG_IFMT"%*[^\n] ", cmtype, cpred, cprey, &mnstep);
#endif
  mtype = (Nag_LARSModelType) nag_enum_name_to_value(cmtype);
  pred = (Nag_LARSPreProcess) nag_enum_name_to_value(cpred);
  prey = (Nag_LARSPreProcess) nag_enum_name_to_value(cprey);

  /* Using all variables */
  isx = 0;

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

  /* Allocate memory for the data */
  ldd = n;
  if (!(y = NAG_ALLOC(n, double)) ||
      !(d = NAG_ALLOC(ldd*m, double)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  /* Read in the data */
  for (i = 0; i < n; i++)
    {
      for (j = 0; j < m; j++)
        {
#ifdef _WIN32
          scanf_s("%lf",&d[j*ldd + i]);
#else
          scanf("%lf",&d[j*ldd + i]);
#endif
        }
#ifdef _WIN32
      scanf_s("%lf",&y[i]);
#else
      scanf("%lf",&y[i]);
#endif
    }
#ifdef _WIN32
```

```
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

  /* Allocate output arrays */
  ldb = m;
  if (!(b = NAG_ALLOC(ldb*(mnstep+2), double)) ||
      !(fitsum = NAG_ALLOC(6*(mnstep+1), double)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  /* Call nag_lars (g02mac) to fit the model */
  nag_lars(mtype, pred, prey, n, m, d, ldd, isx, y, 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 (g02mac).\n%s\n", fail.message);
          exit_status = 1;
          goto END;
        }
      else
        {
          printf("Warning from nag_lars (g02mac).\n%s\n", fail.message);
          exit_status = 2;
        }
    }

  /* Read in the number of additional parameter estimates required and the
     way they will be specified */
#ifdef _WIN32
  scanf_s("%39s%"NAG_IFMT"%*[^\n] ",cktype,_countof(cktype),&lnk);
#else
  scanf("%39s%"NAG_IFMT"%*[^\n] ",cktype,&lnk);
#endif
  ktype = (Nag_LARSTargetType) nag_enum_name_to_value(cktype);

  ldnb = ip;
  if (!(nk = NAG_ALLOC(lnk, double)) ||
      !(nb = NAG_ALLOC(ip*lnk, double)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  /* Read in the target values */
  for (i = 0; i < lnk; i++)
    {
#ifdef _WIN32
      scanf_s("%lf",&nk[i]);
#else
      scanf("%lf",&nk[i]);
#endif
    }
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

  /* nag_lars_param (g02mcc) Calculate the additional parameter estimates */
  nag_lars_param(nstep,ip,b,ldb,fitsum,ktype,nk,lnk,nb,ldnb,&fail);
  if (fail.code != NE_NOERROR)
    {
```

```
      printf("Error from nag_lars_param (g02mcc).\n%s\n", fail.message);
      exit_status = 3;
      goto END;
    }

  printf("Parameter Estimates from nag_lars (g02mac)\n");
  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("Additional Parameter Estimates from nag_lars_param (g02mcc)\n");
  printf("   nk  ");
  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 < lnk; k++)
    {
      printf("  %3.1f",nk[k]);
      for (j = 0; j < ip; j++)
        {
          printf(" %9.3f",nb[k*ldnb + j]);
        }
      printf("\n");
    }

 END:
  NAG_FREE(y);
  NAG_FREE(d);
  NAG_FREE(b);
  NAG_FREE(fitsum);
  NAG_FREE(ropt);
  NAG_FREE(nb);
  NAG_FREE(nk);

  return(exit_status);
}
```

9.2 Program Data

```
nag_lars_param (g02mcc) Example Program Data
20 6                          :: n,m
Nag_LARS_LAR Nag_LARS_CenteredNormalized
Nag_LARS_Centered  6          :: mtype,pred,prey,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
Nag_LARS_StepNumber  5        :: ktype,lnk
0.2 1.2 3.2 4.5 5.2           :: End of nk
```

9.3 Program Results

```
nag_lars (g02mac) Example Program Results

Parameter Estimates from nag_lars (g02mac)
  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

Additional Parameter Estimates from nag_lars_param (g02mcc)
   nk                         Parameter Estimate
 ------------------------------------------------------------------
  0.2     0.000     0.000     0.625     0.000     0.000     0.000
  1.2     0.000     0.000     3.258     0.000     0.000    -0.143
  3.2    -0.483    -0.059     4.018     0.000     0.000    -1.214
  4.5    -0.844    -0.676     4.104    -0.432     0.000    -1.707
  5.2    -1.062    -1.071     4.112    -0.878    -0.012    -1.955
```