NAG Library Function Document
nag_opt_lsq_covariance (e04ycc)

1 Purpose

nag_opt_lsq_covariance (e04ycc) returns estimates of elements of the variance-covariance matrix of the estimated regression coefficients for a nonlinear least squares problem. The estimates are derived from the Jacobian of the function \( f(x) \) at the solution.

nag_opt_lsq_covariance (e04ycc) may be used following either of the NAG C Library nonlinear least squares functions nag_opt_lsq_no_deriv (e04fcc), nag_opt_lsq_deriv (e04gbc).

2 Specification

```c
#include <nag.h>
#include <nage04.h>

void nag_opt_lsq_covariance (Integer job, Integer m, Integer n,
 double fsumsq, double cj[], Nag_E04_Opt *options, NagError *fail)
```

3 Description

nag_opt_lsq_covariance (e04ycc) is intended for use when the nonlinear least squares function, \( F(x) = f^T(x)f(x) \), represents the goodness-of-fit of a nonlinear model to observed data. It assumes that the Hessian of \( F(x) \), at the solution, can be adequately approximated by \( 2J^TJ \), where \( J \) is the Jacobian of \( f(x) \) at the solution. The estimated variance-covariance matrix \( C \) is then given by

\[
C = \sigma^2 (J^TJ)^{-1} \quad J^TJ \text{ nonsingular},
\]

where \( \sigma^2 \) is the estimated variance of the residual at the solution, \( \bar{x} \), given by

\[
\sigma^2 = \frac{F(\bar{x})}{m - n},
\]

\( m \) being the number of observations and \( n \) the number of variables.

The diagonal elements of \( C \) are estimates of the variances of the estimated regression coefficients. See the e04 Chapter Introduction, Bard (1974) and Wolberg (1967) for further information on the use of the matrix \( C \).

When \( J^TJ \) is singular then \( C \) is taken to be

\[
C = \sigma^2 (J^TJ)^\dagger,
\]

where \( (J^TJ)^\dagger \) is the pseudo-inverse of \( J^TJ \), and \( \sigma^2 = \frac{F(\bar{x})}{m - k}, k = \text{rank}(J) \) but in this case the argument fail is returned with fail.Code = NW_LINDEPEND as a warning to you that \( J \) has linear dependencies in its columns. The assumed rank of \( J \) can be obtained from fail.errnum.

The function can be used to find either the diagonal elements of \( C \), or the elements of the \( j \)th column of \( C \), or the whole of \( C \).

nag_opt_lsq_covariance (e04ycc) must be preceded by one of the nonlinear least squares functions mentioned in Section 1, and requires the arguments fsumsq and options to be supplied by those functions. fsumsq is the residual sum of squares \( F(\bar{x}) \) while the structure options contains the members options->s and options->v which give the singular values and right singular vectors respectively in the singular value decomposition of \( J \).
5 Arguments

1: job – Integer
   Input
   On entry: indicates which elements of C are returned as follows:
   - job = -1
     The n by n symmetric matrix C is returned.
   - job = 0
     The diagonal elements of C are returned.
   - job > 0
     The elements of column job of C are returned.
   Constraint: -1 ≤ job ≤ n.

2: m – Integer
   Input
   On entry: the number m of observations (residuals f_i(x)).
   Constraint: m ≥ n.

3: n – Integer
   Input
   On entry: the number n of variables (x_j).
   Constraint: 1 ≤ n ≤ m.

4: fsumsq – double
   Input
   On entry: the sum of squares of the residuals, F(\bar{x}), at the solution \bar{x}, as returned by the nonlinear least squares function.
   Constraint: fsumsq ≥ 0.0.

5: cj[n] – double
   Output
   On exit: with job = 0, cj returns the n diagonal elements of C. With job = j > 0, cj returns the n elements of the jth column of C. When job = -1, cj is not referenced.

6: options – Nag_E04_Opt *
   Input/Output
   On entry/exit: the structure used in the call to the nonlinear least squares function. The following members are relevant to nag_opt_lsq_covariance (e04ycc), their values should not be altered between the call to the least squares function and the call to nag_opt_lsq_covariance (e04ycc).

   s – double
   Input
   On entry: the pointer to the n singular values of the Jacobian as returned by the nonlinear least squares function.

   v – double
   Input/Output
   On entry: the pointer to the n by n right-hand orthogonal matrix (the right singular vectors) of J as returned by the nonlinear least squares function.
   On exit: when job ≥ 0 then v is unchanged.

   When job = -1 then the leading n by n part of v is overwritten by the n by n matrix C. Matrix element i, j is held in v[(i-1) × tdv + j - 1] for i = 1, 2, . . . , n and j = 1, 2, . . . , n.
6 Error Indicators and Warnings

NE_2_INT_ARG_GT
On entry, \( \text{job} = \langle \text{value} \rangle \) while \( n = \langle \text{value} \rangle \). These arguments must satisfy \( \text{job} \leq n \).

NE_2_INT_ARG_LT
On entry, \( m = \langle \text{value} \rangle \) while \( n = \langle \text{value} \rangle \). These arguments must satisfy \( m \geq n \).

NE_INT_ARG_LT
On entry, \( \text{job} \) must not be less than 1: \( \text{job} = \langle \text{value} \rangle \).
On entry, \( n \) must not be less than 1: \( n = \langle \text{value} \rangle \).

NE_REAL_ARG_LT
On entry, \( fsumsq \) must not be less than 0.0: \( fsumsq = \langle \text{value} \rangle \).

NE_SINGULAR_VALUES
The singular values are all zero, so that at the solution the Jacobian matrix has rank 0.

NW_LIN_DEPEND
At the solution the Jacobian matrix contains linear, or near linear, dependencies amongst its columns. \( J \) assumed to have rank \( \langle \text{value} \rangle \).
In this case the required elements of \( C \) have still been computed based upon \( J \) having an assumed rank given by \( \text{fail.errnum} \). The rank is computed by regarding singular values \( \text{options.s}[j] \) that are not larger than \( 10 \epsilon \times \text{options.s}[0] \) as zero, where \( \epsilon \) is the machine precision (see \text{nag_machine_precision (X02AJC)}). If you expect near linear dependencies at the solution and are happy with this tolerance in determining rank you should not call \text{nag_opt_lsq_covariance (e04ycf)} with the null pointer \text{NAGERR_DEFAULT} as the argument \text{fail} but should specifically declare and initialize a NagError structure for the argument \text{fail}.

Overflow
If overflow occurs then either an element of \( C \) is very large, or the singular values or singular vectors have been incorrectly supplied.

7 Accuracy
The computed elements of \( C \) will be the exact covariances corresponding to a closely neighbouring Jacobian matrix \( J \).

8 Parallelism and Performance
Not applicable.
9 Further Comments

When $job = -1$ the time taken by the function is approximately proportional to $n^3$. When $job \geq 0$ the time taken by the function is approximately proportional to $n^2$.

10 Example

This example estimates the variance-covariance matrix $C$ for the least squares estimates of $x_1$, $x_2$ and $x_3$ in the model

$$y = x_1 + \frac{t_1}{x_2 t_2 + x_3 t_3}$$

using the 15 sets of data given in the following table:

<table>
<thead>
<tr>
<th>$y$</th>
<th>0.14</th>
<th>1.00</th>
<th>1.82</th>
<th>2.23</th>
<th>2.95</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>15.01</td>
<td>182.00</td>
<td>14.02</td>
<td>223.00</td>
<td>13.03</td>
</tr>
</tbody>
</table>

The program uses (0.5,1.0,1.5) as the initial guess at the position of the minimum and computes the least squares solution using nag_opt_lsq_no_deriv (e04fcc). Note that the structure options is initialized by nag_opt_init (e04xxc) before calling nag_opt_lsq_no_deriv (e04fcc). See the function documents for nag_opt_lsq_no_deriv (e04fcc), nag_opt_init (e04xxc) and nag_opt_free (e04xzc) for further information.

10.1 Program Text

```c
/* nag_opt_lsq_covariance (e04ycc) Example Program. */
*/

#include <nag.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <nag_stdlib.h>
#include <nage04.h>

#ifndef __cplusplus
extern "C" {
#endif

static void NAG_CALL lsqfun(Integer m, Integer n, const double x[], double fvec[], Nag_Comm *comm);

#ifndef __cplusplus
}
#endif

/* Define a user structure template to store data in lsqfun */
struct user {
    double *y;
    double *t;
};
#define T(I, J) t[(I) *tdt + J]

int main(void) {
    Integer exit_status = 0, i, j, job, m, n, nt, tdj, tdt;
    NagError fail;
    Nag_Comm comm;
    Nag_E04_Opt options;
```
double *cj = 0, *fjac = 0, fsumsq, *fvec = 0, *x = 0;
struct user s;
INIT_FAIL(fail);
s.y = 0;
s.t = 0;
printf("nag_opt_lsq_covariance (e04ycc) Example Program Results\n");
#ifdef _WIN32
scanf_s(" %*[\n"]); /* Skip heading in data file */
#else
scanf(" %*[\n"]); /* Skip heading in data file */
#endif
n=3;
m = 15;
nt = 3;
if (n >= 1 && n <= m) {
  if (!fjac = NAG_ALLOC(m*n, double)) ||
    !fvec = NAG_ALLOC(m, double) ||
    !x = NAG_ALLOC(n, double)) ||
    !cj = NAG_ALLOC(n, double)) ||
    !s.y = NAG_ALLOC(m, double)) ||
    !s.t = NAG_ALLOC(m*nt, double))
    { 
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }
  tdj = n;
  tdt = nt;
} else
  { 
    printf("Invalid n or m.\n");
    exit_status = 1;
    return exit_status;
  }
/* Read data into structure. *
* Observations t (j = 0, 1, 2) are held in s->t[i][j]
* (i = 0, 1, 2, . . ., 14)
*/
for (i = 0; i < m; ++i)
  {
    #ifdef _WIN32
    scanf_s("%lf", &s.y[i]);
    #else
    scanf("%lf", &s.y[i]);
    #endif
    #ifdef _WIN32
    for (j = 0; j < nt; ++j) scanf_s("%lf", &s.T(i, j));
    #else
    for (j = 0; j < nt; ++j) scanf("%lf", &s.T(i, j));
    #endif
  }
/* Set up the starting point */
x[0] = 0.5;
x[1] = 1.0;
x[2] = 1.5;
/* nag_opt_init (e04xzc). *
* Initialization function for option setting *
*/
ag_opt_init(&options); /* Initialise options structure */
/* Assign address of user defined structure to *
* comm.p for communication to lsqfun(). *
*/
comm.p = (Pointer)&s;
/* nag_opt_lsq_no_deriv (e04fcc).
* Unconstrained nonlinear least-squares (no derivatives
* required)
*/
fflush(stdout);
nag_opt_lsq_no_deriv(m, n, lsqfun, x, &fsumsq, fvec, fjac, tdj,
&options, &comm, &fail);
if (fail.code != NE_NOERROR && fail.code != NW_COND_MIN)
{
    printf("Error from nag_opt_lsq_no_deriv (e04fcc).
", fail.message);
    exit_status = 1;
    goto END;
}
job = 0;
/* nag_opt_lsq_covariance (e04ycc).
* Covariance matrix for nonlinear least-squares
*/
nag_opt_lsq_covariance(job, m, n, fsumsq, cj, &options, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_opt_lsq_covariance (e04ycc).
", fail.message);
    exit_status = 1;
    goto END;
}
printf("Estimates of the variances of the sample regression coefficients are:
");
for (i = 0; i < n; ++i)
    printf(" %15.5e", cj[i]);
printf("\n");
/* Free memory allocated to pointers s and v */
/* nag_opt_free (e04xzc).
* Memory freeing function for use with option setting
*/
nag_opt_free(&options, "all", &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_opt_free (e04xzc).
", fail.message);
    exit_status = 1;
    goto END;
}
END:
NAG_FREE(fjac);
NAG_FREE(fvec);
NAG_FREE(x);
NAG_FREE(cj);
NAG_FREE(s.y);
NAG_FREE(s.t);
return exit_status;
}
static void NAG_CALL lsqfun(Integer m, Integer n, const double x[],
double fvec[], Nag_Comm *comm)
{
    /* Function to evaluate the residuals.
     * The address of the user defined structure is recovered in each call
     * to lsqfun() from comm->p and the structure used in the calculation
     * of the residuals.
     */
    Integer i, tdt;
    struct user *s = (struct user *) comm->p;
tdt = n;
}
for (i = 0; i < m; ++i)
  fvec[i] = x[0] +
    s->T(i, 0) / (x[1]*s->T(i, 1) + x[2]*s->T(i, 2)) - s->y[i];
}
/* lsqfun */

10.2 Program Data

nag_opt_lsq_covariance (e04ycc) Example Program Data
0.14 1.0 15.0 1.0
0.18 2.0 14.0 2.0
0.22 3.0 13.0 3.0
0.25 4.0 12.0 4.0
0.29 5.0 11.0 5.0
0.32 6.0 10.0 6.0
0.35 7.0 9.0 7.0
0.39 8.0 8.0 8.0
0.37 9.0 7.0 7.0
0.58 10.0 6.0 6.0
0.73 11.0 5.0 5.0
0.96 12.0 4.0 4.0
1.34 13.0 3.0 3.0
2.10 14.0 2.0 2.0
4.39 15.0 1.0 1.0

10.3 Program Results

nag_opt_lsq_covariance (e04ycc) Example Program Results

Parameters to e04fcc
-------------------
Number of residuals........... 15 Number of variables........... 3
optim_tol............... 1.05e-08 linesearch_tol........... 5.00e-01
step_max................ 1.00e+05 max_iter................ 50
print_level......... Nag_Soln_Iter machine precision....... 1.11e-16
outfile................. stdout

Memory allocation:
  s....................... Nag
  v....................... Nag
  tdv..................... 3

Results from e04fcc:
-------------------

Iteration results:

<table>
<thead>
<tr>
<th>Itn</th>
<th>Nfun</th>
<th>Objective</th>
<th>Norm g</th>
<th>Norm x</th>
<th>Norm (x(k-1)-x(k))</th>
<th>Step</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>1.0210e+01</td>
<td>3.2e+01</td>
<td>1.9e+00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>1.9873e-01</td>
<td>2.8e+00</td>
<td>2.4e+00</td>
<td>7.2e-01</td>
<td>1.0e+00</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>9.2324e-03</td>
<td>1.9e-01</td>
<td>2.6e+00</td>
<td>2.5e-01</td>
<td>1.0e+00</td>
</tr>
<tr>
<td>3</td>
<td>16</td>
<td>8.2149e-03</td>
<td>1.2e-03</td>
<td>2.6e+00</td>
<td>2.7e-02</td>
<td>1.0e+00</td>
</tr>
<tr>
<td>4</td>
<td>25</td>
<td>8.2149e-03</td>
<td>1.2e-07</td>
<td>2.6e+00</td>
<td>3.8e-04</td>
<td>1.0e+00</td>
</tr>
<tr>
<td>5</td>
<td>30</td>
<td>8.2149e-03</td>
<td>3.8e-10</td>
<td>2.6e+00</td>
<td>4.2e-06</td>
<td>1.0e+00</td>
</tr>
</tbody>
</table>

Final solution:

<table>
<thead>
<tr>
<th>x</th>
<th>g</th>
<th>Residuals</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.24106e-02</td>
<td>3.0423e-10</td>
<td>-5.8811e-03</td>
</tr>
<tr>
<td>1.13304e+00</td>
<td>-2.0975e-10</td>
<td>-2.6534e-04</td>
</tr>
<tr>
<td>2.34370e+00</td>
<td>-7.1256e-11</td>
<td>2.7469e-04</td>
</tr>
<tr>
<td>6.5415e-03</td>
<td>-8.2299e-04</td>
<td>-3.4631e-03</td>
</tr>
<tr>
<td>-1.2995e-03</td>
<td>1.9963e-02</td>
<td>8.2216e-02</td>
</tr>
<tr>
<td>-1.8212e-02</td>
<td>-1.4811e-02</td>
<td></td>
</tr>
</tbody>
</table>
The sum of squares is $8.2149 \times 10^{-3}$.

Estimates of the variances of the sample regression coefficients are:

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>$-1.4710 \times 10^{-2}$</td>
<td>$1.53120 \times 10^{-4}$</td>
</tr>
<tr>
<td>$-1.1208 \times 10^{-2}$</td>
<td>$9.48024 \times 10^{-2}$</td>
</tr>
<tr>
<td>$-4.2040 \times 10^{-3}$</td>
<td>$8.77806 \times 10^{-2}$</td>
</tr>
<tr>
<td>$6.8079 \times 10^{-3}$</td>
<td></td>
</tr>
</tbody>
</table>