# NAG Toolbox

# nag_opt_lsq_uncon_covariance (e04yc)

## 1 Purpose

nag_opt_lsq_uncon_covariance (e04yc) 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.

This function may be used following any one of the nonlinear least squares functions nag_opt_lsq_uncon_mod_func_comp (e04fc), nag_opt_lsq_uncon_mod_func_easy (e04fy), nag_opt_lsq_uncon_quasi_deriv_comp (e04gb), nag_opt_lsq_uncon_mod_deriv_comp (e04gd), nag_opt_lsq_uncon_quasi_deriv_easy (e04gy), nag_opt_lsq_uncon_mod_deriv_easy (e04gz), nag_opt_lsq_uncon_mod_deriv2_comp (e04he) or nag_opt_lsq_uncon_mod_deriv2_easy (e04hy).

## 2 Syntax

```
[v, cj, ifail] = nag_opt_lsq_uncon_covariance(job, m, fsumsq, s, v, 'n', n)
[v, cj, ifail] = e04yc(job, m, fsumsq, s, v, 'n', n)
```

## 3 Description

nag_opt_lsq_uncon_covariance (e04yc) is intended for use when the nonlinear least squares function, $F(x) = f^{\mathrm{T}}(x)f(x)$, represents the goodness-of-fit of a nonlinear model to observed data. The function assumes that the Hessian of $F(x)$, at the solution, can be adequately approximated by $2J^{\mathrm{T}}J$, where $J$ is the Jacobian of $f(x)$ at the solution. The estimated variance-covariance matrix $C$ is then given by

$$C = \sigma^2 \left(J^{\mathrm{T}}J\right)^{-1}, \quad J^{\mathrm{T}}J \quad \text{nonsingular},$$

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 $C$.

When $J^{\mathrm{T}}J$ is singular then $C$ is taken to be

$$C = \sigma^2 \left(J^{\mathrm{T}}J\right)^{\dagger},$$

where $\left(J^{\mathrm{T}}J\right)^{\dagger}$ is the pseudo-inverse of $J^{\mathrm{T}}J$, and

$$\sigma^2 = \frac{F(\bar{x})}{m-k}, \quad k = \text{rank } (J)$$

but in this case the argument **ifail** is returned as nonzero as a warning to you that $J$ has linear dependencies in its columns. The assumed rank of $J$ can be obtained from **ifail**.

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_uncon_covariance (e04yc) must be preceded by one of the nonlinear least squares functions mentioned in Section 1, and requires the arguments **fsumsq**, **s** and **v** to be supplied by those functions (e.g., see nag_opt_lsq_uncon_mod_func_comp (e04fc)). **fsumsq** is the residual sum of squares $F(\bar{x})$ and **s** and **v** contain the singular values and right singular vectors respectively in the singular value decomposition of $J$. **s** and **v** are returned directly by the comprehensive functions

nag_opt_lsq_uncon_mod_func_comp (e04fc), nag_opt_lsq_uncon_quasi_deriv_comp (e04gb), nag_opt_lsq_uncon_mod_deriv_comp (e04gd) and nag_opt_lsq_uncon_mod_deriv2_comp (e04he), but are returned as part of the workspace argument **w** (from one of the easy-to-use functions). In the case of nag_opt_lsq_uncon_mod_func_easy (e04fy), **s** starts at $\mathbf{w}(ns)$, where

$$ns = 6 \times \mathbf{n} + 2 \times \mathbf{m} + \mathbf{m} \times \mathbf{n} + 1 + \max(1, \mathbf{n} \times (\mathbf{n} - 1)/2)$$

and in the cases of the remaining easy-to-use functions, **s** starts at $\mathbf{w}(ns)$, where

$$ns = 7 \times \mathbf{n} + 2 \times \mathbf{m} + \mathbf{m} \times \mathbf{n} + \mathbf{n} \times (\mathbf{n} + 1)/2 + 1 + \max(1, \mathbf{n} \times (\mathbf{n} - 1)/2).$$

The argument **v** starts immediately following the elements of **s**, so that **v** starts at $\mathbf{w}(nv)$, where

$$nv = ns + \mathbf{n}.$$

For all the easy-to-use functions the argument $ldv$ must be supplied as **n**. Thus a call to nag_opt_lsq_uncon_covariance (e04yc) following nag_opt_lsq_uncon_mod_func_easy (e04fy) can be illustrated as

```
.
.
.
[x, fsumsq, user, ifail] = e04fy(m, lsfun1, x);
.
.
.
ns = 6*n + 2*m + m*n + 1 + max((1,(n*(n-1))/2);
nv = ns + n;
[v, cj, ifail] = e04yc(job, m, fsumsq, w(ns:nv-1), w(nv:numel(w)));
```

where the arguments **m**, **n**, **fsumsq** and the $\left(n + n^2\right)$ elements $\mathbf{w}(ns), \mathbf{w}(ns+1), \ldots,$ $\mathbf{w}(nv + \mathbf{n}^2 - 1)$ must not be altered between the calls to nag_opt_lsq_uncon_mod_func_easy (e04fy) and nag_opt_lsq_uncon_covariance (e04yc). The above illustration also holds for a call to nag_opt_lsq_uncon_covariance (e04yc) following a call to one of nag_opt_lsq_uncon_quasi_deriv_easy (e04gy), nag_opt_lsq_uncon_mod_deriv_easy (e04gz) or nag_opt_lsq_uncon_mod_deriv2_easy (e04hy), except that $ns$ must be computed as

$$ns = 7 \times \mathbf{n} + 2 \times \mathbf{m} + \mathbf{m} \times \mathbf{n} + (\mathbf{n} \times (\mathbf{n} + 1))/2 + 1 + \max((1, \mathbf{n} \times (\mathbf{n} - 1))/2).$$

# 4 References

Bard Y (1974) *Nonlinear Parameter Estimation* Academic Press

Wolberg J R (1967) *Prediction Analysis* Van Nostrand

# 5 Parameters

## 5.1 Compulsory Input Parameters

1: **job** – INTEGER

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*: $-1 \leq$ **job** $\leq$ **n**.

2: **m** – INTEGER

The number $m$ of observations (residuals $f_i(x)$).

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

3: **fsumsq** – REAL (KIND=nag_wp)

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** $\geq 0.0$.

4: **s**(**n**) – REAL (KIND=nag_wp) array

The $n$ singular values of the Jacobian as returned by the nonlinear least squares function. See Section 3 for information on supplying **s** following one of the easy-to-use functions.

5: **v**($ldv$, **n**) – REAL (KIND=nag_wp) array

$ldv$, the first dimension of the array, must satisfy the constraint if **job** $= -1$, $ldv \geq$ **n**.

The $n$ by $n$ right-hand orthogonal matrix (the right singular vectors) of $J$ as returned by the nonlinear least squares function. See Section 3 for information on supplying **v** following one of the easy-to-use functions.

## 5.2 Optional Input Parameters

1: **n** – INTEGER

*Default*: the dimension of the array **s** and the second dimension of the array **v**. (An error is raised if these dimensions are not equal.)

The number $n$ of variables $\left(x_j\right)$.

*Constraint*: $1 \leq$ **n** $\leq$ **m**.

## 5.3 Output Parameters

1: **v**($ldv$, **n**) – REAL (KIND=nag_wp) array

If **job** $\geq 0$, **v** is unchanged.

If **job** $= -1$, the leading $n$ by $n$ part of **v** stores the $n$ by $n$ matrix $C$. When nag_opt_lsq_uncon_covariance (e04yc) is called with **job** $= -1$ following an easy-to-use function this means that $C$ is returned, column by column, in the $n^2$ elements of **w** given by $\mathbf{w}(nv), \mathbf{w}(nv+1), \ldots, \mathbf{w}(nv+\mathbf{n}^2-1)$. (See Section 3 for the definition of $nv$.)

2: **cj**(**n**) – REAL (KIND=nag_wp) array

If **job** $= 0$, **cj** returns the $n$ diagonal elements of $C$.

If **job** $= j > 0$, **cj** returns the $n$ elements of the $j$th column of $C$.

If **job** $= -1$, **cj** is not referenced.

3: **ifail** – INTEGER

**ifail** $= 0$ unless the function detects an error (see Section 5).

# 6 Error Indicators and Warnings

**Note**: nag_opt_lsq_uncon_covariance (e04yc) may return useful information for one or more of the following detected errors or warnings.

Errors or warnings detected by the function:

**ifail** $= 1$

On entry, **job** $< -1$,
or **job** $>$ **n**,
or **n** $< 1$,
or **m** $<$ **n**,
or **fsumsq** $< 0.0$,
or $ldv <$ **n**.

**ifail** $= 2$ (*warning*)

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

**ifail** $> 2$ (*warning*)

At the solution the Jacobian matrix contains linear, or near linear, dependencies amongst its columns. In this case the required elements of $C$ have still been computed based upon $J$ having an assumed rank given by **ifail** $- 2$. The rank is computed by regarding singular values $SV(j)$ that are not larger than $10\epsilon \times SV(1)$ as zero, where $\epsilon$ is the ***machine precision*** (see nag_machine_precision (x02aj)). If you expect near linear dependencies at the solution and are happy with this tolerance in determining rank you should call nag_opt_lsq_uncon_covariance (e04yc) with **ifail** $= 1$ in order to prevent termination. It is then essential to test the value of **ifail** on exit from nag_opt_lsq_uncon_covariance (e04yc).

**Overflow**

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

**ifail** $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

**ifail** $= -399$

Your licence key may have expired or may not have been installed correctly.

**ifail** $= -999$

Dynamic memory allocation failed.

## 7 Accuracy

The computed elements of $C$ will be the exact covariances corresponding to a closely neighbouring Jacobian matrix $J$.

## 8 Further Comments

When **job** $= -1$ the time taken by nag_opt_lsq_uncon_covariance (e04yc) is approximately proportional to $n^3$. When **job** $\geq 0$ the time taken by the function is approximately proportional to $n^2$.

## 9 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:

| $y$ | $t_1$ | $t_2$ | $t_3$ |
|---|---|---|---|
| 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 |

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_uncon_mod_func_easy (e04fy). See the function document nag_opt_lsq_uncon_mod_func_easy (e04fy) for further information.

## 9.1 Program Text

```
     function e04yc_example

fprintf('e04yc example results\n\n');

global y t;

% Fit data
n = 3;
m = nag_int(15);
y = [0.14, 0.18, 0.22, 0.25, 0.29, 0.32, 0.35, 0.39, 0.37, ...
     0.58, 0.73, 0.96, 1.34, 2.10, 4.39];

t = [1.0, 15.0, 1.0;
     2.0, 14.0, 2.0;
     3.0, 13.0, 3.0;
     4.0, 12.0, 4.0;
     5.0, 11.0, 5.0;
     6.0, 10.0, 6.0;
     7.0,  9.0, 7.0;
     8.0,  8.0, 8.0;
     9.0,  7.0, 7.0;
     10.0, 6.0, 6.0;
     11.0, 5.0, 5.0;
     12.0, 4.0, 4.0;
     13.0, 3.0, 3.0;
     14.0, 2.0, 2.0;
     15.0, 1.0, 1.0];

% Initial guess
x = [0.5;  1;  1.5];

% Minimize
[x, fsumsq, fvec, fjac, s, v, niter, nf, user, ifail] = ...
e04fc( ...
       m, @lsqfun, @lsqmon, x);

fprintf('The sum of squares is %12.4f\n', fsumsq)
fprintf('at the point\n');
fprintf('%12.4f', x);
fprintf('\n\n');

job = nag_int(0);
```

```
[~, cj, ifail] = e04yc(...
                       job, m, fsumsq, s, v);

fprintf('Estimated variances of the sample regression coefficients are:\n');
fprintf('%12.4f', cj);
fprintf('\n');

function [iflag,fvecc, user] = lsqfun(iflag, m, n, xc, user)

  global y t;

  fvecc = zeros(m,1);
  for i=1:double(m)
    fvecc(i) = xc(1) + t(i,1)/(xc(2)*t(i,2)+xc(3)*t(i,3)) - y(i);
  end

function [user] = lsqmon(m, n, xc, fvecc, fjacc, ljc, ...
                         s, igrade, niter, nf, user)
```

## 9.2 Program Results

```
     e04yc example results

The sum of squares is       0.0082
at the point
      0.0824      1.1330      2.3437

Estimated variances of the sample regression coefficients are:
      0.0002      0.0948      0.0878
```