1. **Purpose**

The `nag_opt_estimate_deriv (e04xac)` function computes an approximation to the gradient vector and/or the Hessian matrix for use in conjunction with, or following the use of an optimization function (such as `nag_opt_nlp (e04ucc)`).

2. **Specification**

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

void e04xac(Integer n, double x[],
               void (*objfun)(Integer n, double x[], double *objf,
                              double g[], Nag_Comm *comm),
               double *objf, double g[], double h_forward[],
               double h_central[], double h[], Integer tdh,
               Nag_DerivInfo deriv_info[], Nag_E04_Opt *options,
               Nag_Comm *comm, NagError *fail)
```

3. **Description**

The `nag_opt_estimate_deriv` function is based on the routine FDCALC described in Gill et al. (1983a). It computes finite-difference approximations to the gradient vector and the Hessian matrix for a given function, and aims to provide sufficiently accurate estimates for use with an optimization algorithm.

The simplest approximation of the gradients involves the forward-difference formula, in which the derivative of \( f'(x) \) of a univariate function \( f(x) \) is approximated by the quantity

\[
\rho_f(f,h) = \frac{f(x + h) - f(x)}{h}
\]

for some interval \( h > 0 \), where the subscript ‘F’ denotes ‘forward-difference’ (see Gill et al. (1983b)).

The choice of which gradients are returned by `nag_opt_estimate_deriv` is controlled by the optional parameter `deriv_want` (see Section 7.2 for a description of this parameter). To summarize the procedure used by `nag_opt_estimate_deriv` when `deriv_want` has its default value of `Nag_Grad_HessFull` (i.e., for the case when the objective function is available and the user requires estimates of gradient values and the full Hessian matrix) consider a univariate function \( f \) at the point \( x \). (In order to obtain the gradient of a multivariate function \( F(x) \), where \( x \) is an \( n \)-vector, the procedure is applied to each component of \( x \), keeping the other components fixed.) Roughly speaking, the method is based on the fact that the bound on the relative truncation error in the forward-difference approximation tends to be an increasing function of \( h \), while the relative condition error bound is generally a decreasing function of \( h \), hence changes in \( h \) will tend to have opposite effects on these errors (see Gill et al. (1983b)).

The ‘best’ interval \( h \) is given by

\[
h_F = 2\sqrt{\frac{(1 + |f(x)|)c_R}{\Phi}}
\]

where \( \Phi \) is an estimate of \( f''(x) \), and \( c_R \) is an estimate of the relative error associated with computing the function (see Gill et al. (1981), Chapter 8). Given an interval \( h \), \( \Phi \) is defined by the second-order approximation

\[
\Phi = \frac{f(x + h) - 2f(x) + f(x - h)}{h^2}.
\]

The decision as to whether a given value of \( \Phi \) is acceptable involves \( \hat{c}(\Phi) \), the following bound on the relative condition error in \( \Phi \):

\[
\hat{c}(\Phi) = \frac{4c_R(1 + |f|)}{h^2|\Phi|}
\]
(When $\Phi$ is zero, $\hat{c}(\Phi)$ is taken as an arbitrary large number.)

The procedure selects the interval $h_\phi$ (to be used in computing $\Phi$) from a sequence of trial intervals $(h_k)$. The initial trial interval is taken as

$$h = 2(1 + |x|)\sqrt{e_R},$$

unless the user specifies the initial value to be used.

The value of $\hat{c}(\Phi)$ for a trial value $h_k$ is defined as ‘acceptable’ if it lies in the interval $[0.0001, 0.01]$. In this case $h_\phi$ is taken as $h_k$, and the current value of $\Phi$ is used to compute $h_F$ from (1). If $\hat{c}(\Phi)$ is unacceptable, the next trial interval is chosen so that the relative condition error bound will either decrease or increase, as required. If the bound on the relative condition error is too large, a larger interval is used as the next trial value in an attempt to reduce the condition error bound. On the other hand, if the relative condition error bound is too small, $h_k$ is reduced.

The procedure will fail to produce an acceptable value of $\hat{c}(\Phi)$ in two situations. Firstly, if $f''(x)$ is extremely small, then $\hat{c}(\Phi)$ may never become small, even for a very large value of the interval. Alternatively, $\hat{c}(\Phi)$ may never exceed 0.0001, even for a very small value of the interval. This usually implies that $f''(x)$ is extremely large, and occurs most often near a singularity.

As a check on the validity of the estimated first derivative, the procedure provides a comparison of the forward-difference approximation computed with $h_F$ (as above) and the central-difference approximation computed with $h_\phi$. Using the central-difference formula the first derivative can be approximated by

$$\rho_e(f,h) = \frac{f(x + h) - f(x - h)}{2h}$$

where $h > 0$. If the values $h_F$ and $h_\phi$ do not display some agreement, neither can be considered reliable.

The approximate Hessian matrix $G$ is defined as in Chapter 2 of Gill et al (1981) by

$$G_{ij}(x) = \frac{1}{h_i h_j} (f(x + h_i e_i + h_j e_j) - f(x + h_i e_i) - f(x + h_j e_j) + f(x)),$$

where $h_j$ is the best forward-difference interval associated with the $j$th component of $f$ and $e_j$ is the vector with unity in the $j$th position and zeros elsewhere.

If the user requires the gradients and only the diagonal of the Hessian matrix (i.e., `deriv_want = Nag_Grad_Hess_Diag`; see Section 7.2), `nag_opt_estimate_deriv` follows a similar procedure to the default case, except that the initial trial interval is taken as $10h$, where

$$\bar{h} = 2(1 + |x|)\sqrt{e_R},$$

and the value of $\hat{c}(\Phi)$ for a trial value $h_k$ is defined as acceptable if it lies in the interval $[0.001, 0.1]$. The elements of the Hessian diagonal which are returned in this case are the values of $\Phi$ corresponding to the ‘best’ intervals.

When both function and gradients are available and the user requires the Hessian matrix (i.e., `deriv_want = Nag_Hess_Full`; see Section 7.2), `nag_opt_estimate_deriv` follows a similar procedure to the case above with the exception that the gradient function $g(x)$ is substituted for the objective function and so the forward-difference interval for the first derivative of $g(x)$ with respect to variable $x_j$ is computed. The $j$th column of the approximate Hessian matrix is then defined as in Chapter 2 of Gill et al (1981) by

$$\frac{g(x + h_j e_j) - g(x)}{h_j}$$

where $h_j$ is the best forward-difference interval associated with the $j$th component of $g$. 

3.e04xac.2 [NP:3275/5/pdf]
4. Parameters

\(n\)

Input: the number \(n\) of variables.
Constraint: \(n \geq 1\).

\(x[n]\)

Input: the point \(x\) at which derivatives are required.

\textbf{objfun}

\textbf{objfun} must evaluate the objective function \(F(x)\) and (optionally) its gradient \(g(x) = \partial F/\partial x_j\) for a specified \(n\) element vector \(x\).

The specification for \textbf{objfun} is:

```c
void objfun(Integer n, double x[], double *objf, double g[], Nag_Comm *comm)
```

\(n\)

Input: the number \(n\) of variables.

\(x[n]\)

Input: the point \(x\) at which the value of \(F\) and, if \(\text{comm} \rightarrow \text{flag} = 2\), the \(\partial F/\partial x_j\), are required.

\textbf{objf}

Output: \textbf{objfun} must set \textbf{objf} to the value of the objective function \(F\) at the current point \(x\). If it is not possible to evaluate \(F\) then \textbf{objfun} should assign a negative value to \textbf{comm} \rightarrow \textbf{flag}; \texttt{nag_opt_estimate_deriv} will then terminate.

\(g[n]\)

Output: if \(\text{comm} \rightarrow \text{flag} = 2\) on entry, then \textbf{objfun} must set \(g[j-1]\) to the value of the first derivative \(\partial F/\partial x_j\) at the current point \(x\) for \(j = 1, 2, \ldots, n\). If it is not possible to evaluate the first derivatives then \textbf{objfun} should assign a negative value to \textbf{comm} \rightarrow \textbf{flag}; \texttt{nag_opt_estimate_deriv} will then terminate.

If \(\text{comm} \rightarrow \text{flag} = 0\) on entry, then \(g\) is not referenced.

\textbf{comm}

Pointer to structure of type \texttt{Nag_Comm}; the following members are relevant to \textbf{objfun}.

\textbf{flag} – Integer
Input: \textbf{comm} \rightarrow \textbf{flag} will be set to 0 or 2. The value 0 indicates that only \(F\) itself needs to be evaluated. The value 2 indicates that both \(F\) and its first derivatives must be calculated.
Output: if \textbf{objfun} resets \textbf{comm} \rightarrow \textbf{flag} to a negative number then \texttt{nag_opt_estimate_deriv} will terminate immediately with the error indicator \texttt{NE_USER_STOP}. If \texttt{fail} is supplied to \texttt{nag_opt_estimate_deriv}, \texttt{fail.errnum} will be set to the user’s setting of \textbf{comm} \rightarrow \textbf{flag}.

\textbf{first} – Boolean
Input: will be set to \texttt{TRUE} on the first call to \textbf{objfun} and \texttt{FALSE} for all subsequent calls.

\textbf{nf} – Integer
Input: the number of evaluations of the objective function; this value will be equal to the number of calls made to \textbf{objfun} (including the current one).
nag_opt_estimate_deriv

user - double *
user - Integer *
p - Pointer

The type Pointer will be void * with a C compiler that defines void *
and char * otherwise.

Before calling nag_opt_estimate_deriv these pointers may be allocated
memory by the user and initialized with various quantities for use by objfun
when called from nag_opt_estimate_deriv.

Note: objfun should be thoroughly tested before being used in conjunction with
nag_opt_estimate_deriv. The array x must not be changed by objfun.

objf
Output: the value of the objective function evaluated at the input vector in x.

g[n]
Output: if options.deriv_want = Nag_Grad_HessFull (the default; see Section 7.2) or
Nag_Grad_HessDiag. g[j - 1] contains the best estimate of the first partial derivative for
the jth variable, j = 1, 2, \ldots, n. If options.deriv_want = Nag_HessFull, g[j - 1] contains the
first partial derivative for the jth variable as evaluated by objfun.

h_forward[n]
Input: if the optional parameter use_hfwd_init = FALSE (the default; see Section 7.2), the
values contained in h_forward on entry to nag_opt_estimate_deriv are ignored. If use_hfwd_init
= TRUE, h_forward is assumed to contain meaningful values on entry: if h_forward[j - 1]
> 0 then it is used as the initial trial interval for computing the appropriate partial derivative
to the jth variable, j = 1, 2, \ldots, n; if h_forward[j - 1] \leq 0.0, then the initial trial interval for
the jth variable is computed by nag_opt_estimate_deriv (see Section 3).
Output: h_forward[j - 1] is the best interval found for computing a forward-difference
approximation to the appropriate partial derivative for the jth variable. If the user does not
require this information, a NULL pointer may be provided, and nag_opt_estimate_deriv
will allocate memory internally to calculate the difference intervals.
Constraint: h_forward must not be NULL if options.use_hfwd_init = TRUE.

h_central[n]
Output: h_central[j - 1] is the best interval found for computing a central-difference
approximation to the appropriate partial derivative for the jth variable. If the user does
not require this information, a NULL pointer may be provided, and nag_opt_estimate_deriv
will allocate memory internally to calculate the difference intervals.

h[n][tdh]
Output: if the optional parameter deriv_want = Nag_Grad_HessFull (the default; see Section
7.2) or Nag_HessFull, the estimated Hessian matrix is contained in the leading n by n part
of this array. If deriv_want = Nag_Grad_HessDiag, the n elements of the estimated Hessian
diagonal are contained in the first row of this array.

tdh
Input: the second dimension of the array h as declared in the function from which
nag_opt_estimate_deriv is called.
Constraint: tdh \geq n.

deriv_info[n]
Output: deriv_info[j - 1] contains diagnostic information on the jth variable, j = 1, 2, \ldots, n.
The possible values for deriv_info[j - 1] are:

Nag_Deriv_OK No unusual behaviour observed in estimating the appropriate
derivative.

Nag_Fun_Constant The appropriate function appears to be constant.

Nag_Fun_LinearOdd The appropriate function appears to be linear or odd.

Nag_2ndDeriv_Large The second derivative of the appropriate function appears to be so
large that it cannot be reliably estimated (e.g., near a singularity).
Nag_1stDeriv_Small The forward-difference and central-difference estimates of the appropriate first derivatives do not agree to half a decimal place; this usually occurs because the first derivative is small.

A more detailed explanation of these warnings is given in Section 9.2.

options
Input/Output: a pointer to a structure of type Nag_E04_Opt whose members are optional parameters for nag_opt_estimate_deriv. These structure members offer the means of adjusting some of the parameter values of the computation and on output will supply further details of the results. A description of the members of options is given below in Section 7.

If any of these optional parameters are required then the structure options should be declared and initialized by a call to nag_opt_init (e04xzc) and supplied as an argument to nag_opt_estimate_deriv. However, if the optional parameters are not required the NAG defined null pointer, E04_DEFAULT, can be used in the function call.

comm
Input/Output: structure containing pointers for communication with user-supplied functions; see the above description of objfun for details. If the user does not need to make use of this communication feature, the null pointer NAGCOMM_NULL may be used in the call to nag_opt_estimate_deriv; comm will then be declared internally for use in calls to user-supplied functions.

fail
The NAG error parameter, see the Essential Introduction to the NAG C Library. Users are recommended to declare and initialize fail and set fail.print = TRUE for this function.

4.1. Description of Printed Output

Results from nag_opt_estimate_deriv are printed out by default. The level of printed output can be controlled by the user with the structure members options.list and options.print_deriv (see Section 7.2). If list = TRUE then the parameter values to nag_opt_estimate_deriv are listed, whereas printout of results is governed by the value of print_deriv.

The default of Nag_D_Print provides the following line of output for each variable.

\[
\begin{align*}
\text{j} & \quad \text{the index of the variable for which the difference interval has been computed.} \\
\text{X(j)} & \quad \text{the value of } x_j \text{ as provided by the user in } x[j-1]. \\
\text{Fwd diff int} & \quad \text{the best interval found for computing a forward-difference approximation to the appropriate partial derivative with respect to } x_j. \\
\text{Cent diff int} & \quad \text{the best interval found for computing a central-difference approximation to the appropriate partial derivative with respect to } x_j. \\
\text{Error est} & \quad \text{a bound on the estimated error in the final forward-difference approximation. When deriv_info}[j-1] = \text{Nag_Fun_Constant}, \text{Error est} \text{ is set to zero.} \\
\text{Grad est} & \quad \text{best estimate of the first partial derivative with respect to } x_j. \\
\text{Hess diag est} & \quad \text{best estimate of the second partial derivative with respect to } x_j. \\
\text{Nfun} & \quad \text{the number of function evaluations used to compute the final difference intervals for } x_j. \\
\text{Info} & \quad \text{gives diagnostic information for } x_j. \text{ Info will be one of OK, Constant?, Linear or odd?, Large 2nd deriv?, or Small 1st deriv?, corresponding to deriv_info}[j-1] = \text{Nag_Deriv_OK}, \text{Nag_Fun_Constant, Nag_Fun_LinearOdd, Nag_2ndDeriv_Large} \text{ or Nag_1stDeriv_Small, respectively.}
\end{align*}
\]
5. Comments

A list of possible error exits and warnings from nag_opt_estimate_deriv is given in Section 8. Details of timing, accuracy, and diagnostic information returned in deriv_info, are given in Section 9.

6. Example 1

Compute the gradient vector and Hessian matrix of the following function:

\[ F(x) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4 \]

at the point \((3, -1, 0, 1)^T\).

This example shows the simple use of nag_opt_estimate_deriv where default values are used for all optional parameters. An example showing the use of optional parameters is given in Section 12. There is one example program file, the main program of which calls both examples. The main program and Example 1 are given below.

6.1. Program Text

```c
/* nag_opt_estimate_deriv(e04xac) Example Program.
 */
#include <nag.h>
#include <nag_stdlib.h>
#include <stdio.h>
#include <nage04.h>

#ifdef NAG_PROTO
static void ex1(void);
static void ex2(void);
static void objfun(Integer n, double x[], double *objf, double g[], Nag_Comm *comm);
#else
static void ex1();
static void ex2();
static void objfun();
#endif

#ifdef NAG_PROTO
static void objfun(Integer n, double x[], double *objf, double g[], Nag_Comm *comm)
#else
static void objfun(n, x, objf, g, comm)
#endif
{
    Integer n;
    double x[], double *objf, g[];
    Nag_Comm *comm;
{
    double a, asq, b, bsq, c, csq, d, dsq;
    a = x[0] + 10.0*x[1];
    b = x[2] - x[3];
    c = x[1] - 2.0*x[2];
    d = x[0] - x[3];
    asq = a*a;
    bsq = b*b;
    csq = c*c;
    dsq = d*d;
    *objf = asq + 5.0*bsq + csq*csq + 10.0*dsq*dsq;
    if (comm->flag == 2)
    {
        g[0] = 2.0*a + 40.0*d*dsq;
        g[1] = 20.0*a + 4.0*c*csq;
        g[2] = 10.0*b - 8.0*c*csq;
```
```c
double g[3] = -10.0*b - 40.0*d*dsq;
}

main()
{
    Vprintf("e04xac Example Program Results\n");
    ex1();
    ex2();
    exit(EXIT_SUCCESS);
}

static void ex1(void)
{
    #define MAXN 4
    /* Local variables */
    Integer n, tdh;
    double objf;
    double x[MAXN];
    double g[MAXN], h[MAXN][MAXN];
    Nag_DerivInfo deriv_info[MAXN];
    static NagError fail;
    n = MAXN;
    tdh = MAXN;
    Vprintf("\nExample 1: default options\n");
    x[0] = 3.0;
    x[1] = -1.0;
    x[2] = 0.0;
    x[3] = 1.0;
    fail.print = TRUE;
    e04xac(n, x, objfun, &objf, g, (double*)0, (double*)0, 
            (double*)h, tdh, deriv_info, E04_DEFAULT, NAGCOMM_NULL, &fail);
} /* ex1 */
```

### 6.2. Program Data

None.

### 6.3. Program Results

e04xac Example Program Results

Example 1: default options

<table>
<thead>
<tr>
<th>Parameters to e04xac</th>
</tr>
</thead>
<tbody>
<tr>
<td>deriv_want...........Nag_Grad_HessFull</td>
</tr>
<tr>
<td>f_prec.......................4.37e-15</td>
</tr>
<tr>
<td>print_deriv............Nag_D_Print</td>
</tr>
</tbody>
</table>

| j X(j) Fwd diff int Cent diff int Error est Grad est Hess diag est |
|-----------------------|-----------------|-----------------|----------------|-----------------|-----------------|-----------------|
[NP3275/5/pdf] 7
7. Optional Parameters

A number of optional input and output parameters to nag_opt_estimate_deriv are available through the structure argument options, type Nag_E04_Opt. A parameter may be selected by assigning an appropriate value to the relevant structure member; those parameters not selected will be assigned default values. If no use is to be made of any of the optional parameters the user should use the NAG defined null pointer, E04_DEFAULT, in place of options when calling nag_opt_estimate_deriv; the default settings will then be used for all parameters.

Before assigning values to options directly the structure must be initialized by a call to the function nag_opt_init (e04xcc). Values may then be assigned to the structure members in the normal C manner.

Option settings may also be read from a text file using the function nag_opt_read (e04xcy) in which case initialization of the options structure will be performed automatically if not already done. Any subsequent direct assignment to the options structure must not be preceded by initialization.

7.1. Optional Parameter Checklist and Default Values

For easy reference, the following list shows the members of options which are valid for nag_opt_estimate_deriv together with their default values where relevant. The number $\epsilon$ is a generic notation for machine precision (see nag_machine_precision (X02AJC)).

Boolean list  TRUE
Nag_DPrintType print_deriv Nag_D_Print
char outfile[80] stdout
Nag_DWantType deriv_want Nag_Grad_HessFull
Boolean use_hfwd_init FALSE
double f_prec $\epsilon^{0.9}$
double f_prec_used
Integer nf

7.2. Description of Optional Parameters

list – Boolean Default = TRUE
Input: if options.list = TRUE the parameter settings in the call to nag_opt_estimate_deriv will be printed.

print_deriv – Nag_DPrintType Default = Nag_D_Print
Input: controls whether printout is produced by nag_opt_estimate_deriv. The following values are available.

Nag_D_NoPrint No output.
Nag_D_Print Printout for each variable as described in Section 4.1.

Constraint: options.print_deriv = Nag_D_NoPrint or Nag_D_Print.

outfile – char[80] Default = stdout
Input: the name of the file to which results should be printed. If options.outfile[0] = ‘\0’ then the stdout stream is used.
deriv_want – Nag_DWantType

User: specifies which derivatives nag_opt_estimate_deriv should estimate. The following values are available.

Nag_Grad_HessFull  Estimate the gradient and full Hessian, with the user supplying the objective function via objfun.

Nag_Grad_HessDiag Estimate the gradient and the Hessian diagonal values, with the user supplying the objective function via objfun.

Nag_HessFull  Estimate the full Hessian, with the user supplying the objective function and gradients via objfun.

Constraint: options.deriv_want = Nag_Grad_HessFull, Nag_Grad_HessDiag or Nag_HessFull.

use_lfwd_init – Boolean

User: if use_lfwd_init = FALSE, then nag_opt_estimate_deriv ignores any values supplied on entry in h_forward, and computes the initial trial intervals itself. If use_lfwd_init = TRUE, then nag_opt_estimate_deriv uses the forward difference interval provided by the user in h_forward[j−1] as the initial trial interval for computing the appropriate partial derivative to the jth variable, j = 1, 2, ... , n; however, if h_forward[j−1] ≤ 0.0 for some j, the initial trial interval for the jth variable is computed by nag_opt_estimate_deriv.

l_prec – double

User: specifies $\epsilon_{R}$, which is intended to measure the accuracy with which the problem function $F$ can be computed. The value of l_prec should reflect the relative precision of $1 + |F(x)|$, i.e., acts as a relative precision when $|F|$ is large, and as an absolute precision when $|F|$ is small. For example, if $|F(x)|$ is typically of order 1000 and the first six significant figures are known to be correct, an appropriate value of l_prec would be $10^{-6}$. The default value of $\epsilon_{R}$ will be appropriate for most simple functions that are computed with full accuracy.

A discussion of $\epsilon_{R}$ is given in Gill et al (1981), Chapter 8. If the user provides a value of l_prec which nag_opt_estimate_deriv determines to be either too small or too large, the default value will be used instead and a warning will be output if optional parameter print_deriv = Nag_D_Print. The value actually used is returned in options.l_prec_used.

Constraint: options.l_prec > 0.

l_prec_used – double

Output: if fail.code = NE_NOERROR or NW_DERIV_INFO, or if options.nf > 1 and fail.code = NE_USER_STOP, then l_prec_used contains the value of $\epsilon_{R}$ used by nag_opt_estimate_deriv. If the user supplies a value for options.l_prec and nag_opt_estimate_deriv considers that the value supplied is neither too large nor too small, then this value will be returned in l_prec_used; otherwise l_prec_used will contain the default value, $\epsilon_{R}^{0.9}$.

nf – double

Output: the number of times the objective function has been evaluated (i.e., number of calls of objfun).

8. Error Indications and Warnings

NE_USER_STOP

User requested termination, user flag value = ⟨value⟩.

This exit occurs if the user sets comm->flag to a negative value in objfun. If fail is supplied, the value of fail.errnum will be the same as the user’s setting of comm->flag.

NE_INT_ARG_LT

On entry, n must not be less than 1: n = ⟨value⟩.

NE_INT_ARG_LT

On entry, tdh = ⟨value⟩ while n = ⟨value⟩. These parameters must satisfy tdh ≥ n.
NE_OPT_NOT_INIT
Options structure not initialized.

NE_BAD_PARAM
On entry, parameter options.print_deriv had an illegal value.
On entry, parameter options.deriv_want had an illegal value.

NE_INVALID_REAL_RANGE_F
Value (value) given to options.f_prec is not valid.
Correct range is $l_{\text{prec}} > 0.0$.

NE_FORWARD_NULL
options.use_fwd_init = TRUE but argument h_forward is NULL.

NE_ALLOC_FAIL
Memory allocation failed.

NW_DERIV_INFO
On exit, at least one element of the deriv_info array does not contain the value Nag_Deriv_OK.
This does not necessarily represent an unsuccessful exit.

See Section 9.2 for information about the possible values which may be returned in deriv_info.

NE_NOT_APPEND_FILE
Cannot open file (string) for appending.

NE_WRITE_ERROR
Error occurred when writing to file (string).

NE_NOT_CLOSE_FILE
Cannot close file (string).

9. Further Comments

9.1. Accuracy
The function exits with fail.code = NE_NOERROR if the algorithm terminated successfully, i.e., the forward-difference estimates of the appropriate first derivatives (computed with the final estimate of the ‘optimal’ forward-difference interval $h_F$) and the central-difference estimates (computed with the interval $h_o$ used to compute the final estimate of the second derivative) agree to at least half a decimal place.

9.2. Diagnostic Information
Diagnostic information is returned via the array parameter deriv_info. If fail.code = NE_NOERROR on exit then deriv_info[$j - 1$] = Nag_Deriv_OK, for $j = 1, 2, \ldots, n$. If fail.code = NW_DERIV_INFO on exit, then, for at least one $j$, deriv_info[$j - 1$] contains one of the following values:

Nag_Fun_Constant
The appropriate function appears to be constant. On exit, $h_{\text{forward}}[j - 1]$ is set to the initial trial interval corresponding to a well scaled problem, and Error est in the printed output is set to zero. This value occurs when the estimated relative condition error in the first derivative approximation is unacceptably large for every value of the finite-difference interval. If this happens when the function is not constant the initial interval may be too small; in this case, it may be worthwhile to rerun nag_opt_estimate_deriv with larger initial trial interval values supplied in h_forward and with the optional parameter use_fwd_init set to TRUE. This error may also occur if the function evaluation includes an inordinately large constant term or if optional parameter f_prec is too large.

Nag_Fun_LinearOdd
The appropriate function appears to be linear or odd. On exit, $h_{\text{forward}}[j - 1]$ is set to the smallest interval with acceptable bounds on the relative condition error in the forward- and backward-difference estimates. In this case, the estimated relative condition error in the second derivative approximation remained large for every trial interval, but the estimated error in the first derivative approximation was acceptable for at least one interval. If the function is not linear or odd the relative condition error in the second derivative may be decreasing very slowly. It may be worthwhile to rerun nag_opt_estimate_deriv with larger initial trial interval values supplied in h_forward and with use_fwd_init set to TRUE.
Nag_2ndDeriv_Large
The second derivative of the appropriate function appears to be so large that it cannot be reliably estimated (e.g., near a singularity). On exit, h\textsubscript{forward}[j - 1] is set to the smallest trial interval.

This value occurs when the relative condition error estimate in the second derivative remained very small for every trial interval.

If the second derivative is not large the relative condition error in the second derivative may be increasing very slowly. It may be worthwhile to rerun nag_opt_estimate_deriv with smaller initial trial interval values supplied in h\textsubscript{forward} and with use\textsubscript{hfwd_init} set to TRUE. This error may also occur when the given value of the optional parameter \textit{f_prec} is not a good estimate of a bound on the absolute error in the appropriate function (i.e., \textit{options.f_prec} is too small).

Nag_1stDeriv_Small
The algorithm terminated with an apparently acceptable estimate of the second derivative. However the forward-difference estimates of the appropriate first derivatives (computed with the final estimate of the ‘optimal’ forward-difference interval) and the central difference estimates (computed with the interval used to compute the final estimate of the second derivative) do not agree to half a decimal place. The usual reason that the forward- and central-difference estimates fail to agree is that the first derivative is small.

If the first derivative is not small, it may be helpful to run nag\_opt\_estimate\_deriv at a different point.

9.3. Timing

Unless the objective function can be evaluated very quickly, the run time will usually be dominated by the time spent in \textit{objfun}.

To evaluate an acceptable set of finite-difference intervals for a well-scaled problem nag\_opt\_estimate\_deriv will use around 2 function evaluations per variable; in a badly scaled problem 6 function evaluations per variable may be needed.

In the default case where gradients and the full Hessian matrix are required (i.e., optional parameter deriv\_want = Nag_Grad_HessFull), nag\_opt\_estimate\_deriv performs a further $3n(n+1)/2$ function evaluations. If the full Hessian matrix is required, with the user supplying both function and gradients (i.e., deriv\_want = Nag_HessFull), a further $n$ function evaluations are performed.

10. References


11. See Also

nag\_opt\_nlp \hspace{1em} (e04ucc)
nag\_opt\_init \hspace{1em} (e04xcc)
nag\_opt\_read \hspace{1em} (e04ycx)

12. Example 2

This example shows the use of certain optional parameters. The same \textit{objfun} is used as in Example 1 and the derivatives are estimated at the same point. The options structure is declared and initialized by nag\_opt\_init \hspace{1em} (e04xcc). Two options are set to suppress all printout from nag\_opt\_estimate\_deriv: options\_list is set to FALSE and options\_print\_deriv is set to Nag_D_NoPrint. options\_deriv\_want is set to Nag_Grad_HessDiag and nag\_opt\_estimate\_deriv is called. The returned function value and estimated derivative values are printed out and options\_deriv\_want is reset to Nag_HessFull before...
nag_opt_estimate_deriv is called again. On return, the computed function value and gradient, and estimated Hessian, are printed out.

12.1. Program Text

```c
#include NAG_PROTO

static void ex2(void)
{
    /* Local variables */
    Integer i, j;
    Integer n, tdh;
    double objf;
    double x[MAXN];
    double h_central[MAXN];
    double h_forward[MAXN];
    double g[MAXN], h[MAXN][MAXN], hess_diag[MAXN];

    Nag_DerivInfo deriv_info[MAXN];
    Nag_E04_Opt options;
    static NagError fail;
    n = MAXN;
    tdh = MAXN;
    x[0] = 3.0;
    x[1] = -1.0;
    x[2] = 0.0;
    x[3] = 1.0;
    fail.print = TRUE;

    Vprintf("Example 2: some options are set
");
    e04xxc(&options);
    options.list = FALSE;
    options.print_deriv = Nag_D_NoPrint;
    options.deriv_want = Nag_Grad_HessDiag;

    Vprintf("Estimate gradient and Hessian diagonals given function only
");
    /* Note: it is acceptable to pass an array of length n (hess_diag)
     * as the Hessian parameter in this case.
     */
    e04xac(n, x, objfun, &objf, g, h_forward, h_central,
           hess_diag, tdh, deriv_info, &options, NAGCOMM_NULL, &fail);

    Vprintf("Function value: %12.4e
", objf);
    Vprintf("Estimated gradient vector
");
    for (i = 0; i < n; ++i)
    Vprintf("%12.4e ", g[i]);
    Vprintf("Estimated Hessian matrix diagonal
");
    for (i = 0; i < n; ++i)
    Vprintf("%12.4e ", hess_diag[i]);
    Vprintf("\n");

    options.deriv_want = Nag_HessFull;

    Vprintf("Estimate full Hessian given function and gradients
");
    e04xac(n, x, objfun, &objf, g, h_forward, h_central,
           (double*)h, tdh, deriv_info, &options, NAGCOMM_NULL, &fail);

    Vprintf("Function value: %12.4e
", objf);
    Vprintf("Computed gradient vector
");
    for (i = 0; i < n; ++i)
    Vprintf("%12.4e ", g[i]);
    Vprintf("Estimated Hessian matrix\n");
    for (i = 0; i < n; ++i)
```


```c
{ 
    for (j = 0; j < n; ++j) 
        Vprintf("%12.4e ", h[i][j]); 
    Vprintf("\n"); 
} 
exit(EXIT_SUCCESS); /* ex2 */
```

12.2. Program Data

None.

12.3. Program Results

Example 2: some options are set

Estimate gradient and Hessian diagonals given function only

Function value: 2.1500e+02
Estimated gradient vector
  3.0600e+02  -1.4400e+02  -2.0000e+00  -3.1000e+02
Estimated Hessian matrix diagonal
  4.8200e+02  2.1200e+02  5.7995e+01  4.9000e+02

Estimate full Hessian given function and gradients

Function value: 2.1500e+02
Computed gradient vector
  3.0600e+02  -1.4400e+02  -2.0000e+00  -3.1000e+02
Estimated Hessian matrix
  4.8200e+02  2.0000e+01  0.0000e+00  -4.8000e+02
  2.0000e+01  2.1200e+02  -2.4000e+01  0.0000e+00
  0.0000e+00  -2.4000e+01  5.8000e+01  -1.0000e+01
  -4.8000e+02  0.0000e+00  -1.0000e+01  4.9000e+02