nag_opt_check_deriv (e04hcc)

1. Purpose

nag_opt_check_deriv (e04hcc) checks that a user-defined C function for evaluating an objective function and its first derivatives produces derivative values which are consistent with the function values calculated.

2. Specification

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

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

3. Description

The function nag_opt_bounds_deriv (e04kbc) for minimizing a function of several variables requires the user to supply a C function to evaluate the objective function \( F(x_1, x_2, \ldots, x_n) \) and its first derivatives. nag_opt_check_deriv is designed to check the derivatives calculated by such a user-supplied function. As well as the function to be checked (\texttt{objfun}), the user must supply a point \( x = (x_1, x_2, \ldots, x_n)^T \) at which the check is to be made.

nag_opt_check_deriv first calls the supplied function \texttt{objfun} to evaluate \( F \) and its first derivatives \( g_j = \frac{\partial F}{\partial x_j} \), for \( j = 1, 2, \ldots, n \) at \( x \). The components of the user-supplied derivatives along two orthogonal directions (defined by unit vectors \( p_1 \) and \( p_2 \), say) are then calculated; these will be \( g^T p_1 \) and \( g^T p_2 \) respectively. The same components are also estimated by finite differences, giving quantities

\[
v_k = \frac{F(x + hp_k) - F(x)}{h}, \quad k = 1, 2
\]

where \( h \) is a small positive scalar. If the relative difference between \( v_1 \) and \( g^T p_1 \) or between \( v_2 \) and \( g^T p_2 \) is judged too large, an error indicator is set.

4. Parameters

\( n \)

Input: the number \( n \) of independent variables in the objective function.

Constraint: \( n \geq 1 \).

\texttt{objfun}

\texttt{objfun} must evaluate the objective function and its first derivatives at a given point. (The minimization function nag_opt_bounds_deriv (e04kbc) gives the user the option of resetting a parameter, \texttt{comm->flag}, to terminate the minimization process immediately. nag_opt_check_deriv will also terminate immediately, without finishing the checking process, if the parameter in question is reset to a negative value.)

The specification of \texttt{objfun} is:
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 F and its derivatives are required.

objf
Output: objfun must set objf to the value of the objective function F at the current point x. If it is not possible to evaluate F then objfun should assign a negative value to comm->flag; nag_opt_check_deriv will then terminate.

g[n]
Output: unless comm->flag is reset to a negative number, objfun must set g[j - 1] to the value of the first derivative \( \frac{\partial F}{\partial x_j} \) at the current point x for \( j = 1, 2, \ldots, n \).

comm
Pointer to structure of type Nag_Comm; the following members are relevant to objfun.

flag – Integer
Input: comm->flag will be set to 2.
Output: if objfun resets comm->flag to some negative number then nag_opt_check_deriv will terminate immediately with the error indicator \text{NE_USER_STOP}. If fail is supplied to nag_opt_check_deriv fail.errnum will be set to the user’s setting of comm->flag.

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

nf – Integer
Input: the number of calculations of the objective function; this value will be equal to the number of calls made to objfun including the current one.

user – double *

iuser – Integer *
p – Pointer

The type Pointer will be void * with a C compiler that defines void * and char * otherwise. Before calling nag_opt_check_deriv these pointers may be allocated memory by the user and initialised with various quantities for use by objfun when called from nag_opt_check_deriv.

The array x must not be changed by objfun.

x[n]
Input: \( x[j - 1] \), for \( j = 1, 2, \ldots, n \) must be set to the co-ordinates of a suitable point at which to check the derivatives calculated by objfun. ‘Obvious’ settings, such as 0.0 or 1.0, should not be used since, at such particular points, incorrect terms may take correct values (particularly zero), so that errors could go undetected. Similarly, it is preferable that no two elements of x should be the same.

objf
Output: unless the user sets comm->flag negative in the first call of objfun, objf contains the value of the objective function \( F(x) \) at the point given by the user in x.

g[n]
Output: unless the user sets comm->flag negative in the first call of objfun, g[j - 1] contains the value of the derivative \( \frac{\partial F}{\partial x_j} \) at the point given in x, as calculated by objfun, for \( j = 1, 2, \ldots, n \).

comm
Input/Output: structure containing pointers for communication with the user defined
function; see the above description of \texttt{objfun} for details. If the user does not need to make use of this communication feature the null pointer \texttt{NAGCOMM=NULL} may be used in the call to \texttt{nag\_opt\_check\_deriv}; \texttt{comm} will then be declared internally for use in calls to \texttt{objfun}.

\textbf{fail}

The NAG error parameter, see the Essential Introduction to the NAG C Library.

5. Error Indications and Warnings

\textbf{NE\_USER\_STOP}

User requested termination, user flag value = \{value\}.

This exit occurs if the user sets \texttt{comm->flag} to a negative value in \texttt{objfun}. If \texttt{fail} is supplied the value of \texttt{fail.errnum} will be the same as the user’s setting of \texttt{comm->flag}. The check on \texttt{objfun} will not have been completed.

\textbf{NE\_INT\_ARG\_LT}

On entry, \texttt{n} must not be less than 1: \texttt{n = \{value\}}.

\textbf{NE\_ALLOC\_FAIL}

Memory allocation failed.

\textbf{NE\_DERIV\_ERRORS}

Large errors were found in the derivatives of the objective function.

The user should check carefully the derivation and programming of expressions for the derivatives of $F(x)$, because it is very unlikely that \texttt{objfun} is calculating them correctly.

6. Further Comments

The user-defined function \texttt{objfun} is called three times.

Before using \texttt{nag\_opt\_check\_deriv} to check the calculation of first derivatives, the user should be confident that \texttt{objfun} is calculating $F$ correctly. The usual way of checking the calculation of the function is to compare values of $F(x)$ calculated by \texttt{objfun} at non-trivial points $x$ with values calculated independently. (‘Non-trivial’ means that, as when setting $x$ before calling \texttt{nag\_opt\_check\_deriv}, co-ordinates such as 0.0 or 1.0 should be avoided.)

6.1. Accuracy

\texttt{fail.code} is set to \textbf{NE\_DERIV\_ERRORS} if

$$(v_k - g^T p_k)^2 \geq h \times ((g^T p_k)^2 + 1)$$

for $k = 1$ or 2. (See Section 3 for definitions of the quantities involved.) The scalar $h$ is set equal to $\sqrt{\epsilon}$, where $\epsilon$ is the \textit{machine precision} as given by \texttt{nag\_machine\_precision} (X02AJC).

7. See Also

\texttt{nag\_opt\_bounds\_deriv (e04kbc)}

8. Example

Suppose that it is intended to use \texttt{nag\_opt\_bounds\_deriv (e04kbc)} to minimize

$$F = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4.$$ 

The following program could be used to check the first derivatives calculated by the required function \texttt{objfun}. (The test of whether \texttt{comm->flag} $\neq 0$ in \texttt{objfun} is present for when \texttt{objfun} is called by \texttt{nag\_opt\_bounds\_deriv (e04kbc)}. \texttt{nag\_opt\_check\_deriv} will always call \texttt{objfun} with \texttt{comm->flag} set to 2.)
8.1. Program Text

```c
/* nag_opt_check_deriv (e04hcc) Example Program
*/

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

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

#define NMAX 4

#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)
  Integer n;
  double x[];
  double *objf;
  double g[];
  Nag_Comm *comm;
#endif
{
  /* objfun evaluates the objective function and its derivatives. */

  double x1, x2, x3, x4;
  double tmp, tmp1, tmp2, tmp3, tmp4;
  x1 = x[0];
  x2 = x[1];
  x3 = x[2];
  x4 = x[3];

  /* Supply a single function value */
  tmp1 = x1 + 10.0*x2;
  tmp2 = x3 - x4;
  tmp3 = x2 - 2.0*x3, tmp3 *= tmp3;
  tmp4 = x1 - x4, tmp4 *= tmp4;
  *objf = tmp1*tmp1 + 5.0*tmp2*tmp2 + tmp3*tmp3 + 10.0*tmp4*tmp4;

  if (comm->flag != 0)
  {
    /* Calculate the derivatives */
    tmp = x1 - x4;
    g[0] = 2.0*(x1 + 10.0*x2) + 40.0*tmp*tmp*tmp;
    tmp = x2 - 2.0*x3;
    g[1] = 20.0*(x1 + 10.0*x2) + 4.0*tmp*tmp*tmp;
    tmp = x2 - 2.0*x3;
    g[2] = 10.0*(x3 - x4) - 8.0*tmp*tmp*tmp;
    tmp = x1 - x4;
    g[3] = 10.0*(x4 - x3) - 40.0*tmp*tmp*tmp;
  }
}

main()
{
  double x[NMAX], g[NMAX];
```
double objf;
Integer i, n;
static NagError fail;

fail.print = TRUE;

Vprintf("e04hcc Example Program Results.\n");

n = NMAX;
x[0] = 1.46;
x[1] = -0.82;
x[2] = 0.57;
x[3] = 1.21;

Vprintf("The test point is:\n");
for (i = 0; i < n; ++i)
  Vprintf(" %8.4f", x[i]);
Vprintf("\n");

/* Call derivative checker */
e04hcc(n, objfun, x, &objf, g, NAGCOMM_NULL, &fail);

if (fail.code != NE_NOERROR) exit(EXIT_FAILURE);

Vprintf("First derivatives are consistent with function values.\n\n");
Vprintf("At the test point, objfun gives the function value %11.4e\n", objf);
Vprintf("and the 1st derivatives\n");
for (i = 0; i < n; ++i)
  Vprintf(" %9.3e ", g[i]);
Vprintf("\n");
exit(EXIT_SUCCESS);
} /* main */

8.2. Program Data
None.

8.3. Program Results

The test point is:
  1.4600  -0.8200  0.5700  1.2100

First derivatives are consistent with function values.

At the test point, objfun gives the function value  6.2273e+01
and the 1st derivatives

  -1.285e+01  -1.649e+02  5.384e+01  5.775e+00