NAG C Library Function Document

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

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

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

3 Description

The function nag_opt_nlp_solve (e04wdc) 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 (e04hcc) is designed to check the derivatives calculated by such a user-supplied function. As well as the function to be checked (**objfun**), the user must supply a point $x = (x_1, x_2, \ldots, x_n)^{\mathrm{T}}$ at which the check is to be made.

nag_opt_check_deriv (e04hcc) first calls the supplied function **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 $g^{\mathrm{T}} p_1$ and $g^{\mathrm{T}} p_2$ 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 $g^{\mathrm{T}} p_1$ or between v_2 and $g^{\mathrm{T}} p_2$ is judged too large, an error indicator is set.

4 References

None.

5 Arguments

1: **n** – Integer *Input*

On entry: the number n of independent variables in the objective function.

Constraint: $\mathbf{n} \geq 1$.

2: **objfun** – function, supplied by the user *External Function*

objfun must evaluate the objective function and its first derivatives at a given point. (The minimization function nag_opt_nlp_solve (e04wdc) gives the user the option of resetting a parameter, **comm** → **flag**, to terminate the minimization process immediately. nag_opt_check_deriv (e04hcc) will also terminate immediately, without finishing the checking process, if the parameter in question is reset to a negative value.)

Its specification is:

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

1: **n** – Integer *Input*

On entry: the number n of variables.

2: **x**[**n**] – const double *Input*

On entry: the point x at which F and its derivatives are required.

3: **objf** – double * *Output*

On exit: **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 (e04hcc) will then terminate.

4: **g**[**n**] – double *Output*

On exit: 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 $j = 1, 2, \ldots, n$

5: **comm** – Nag_Comm *

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

flag – Integer *Input/Output*

On entry: **comm** → **flag** will be set to 2.

On exit: if **objfun** resets **comm** → **flag** to some negative number then nag_opt_check_deriv (e04hcc) will terminate immediately with the error indicator **NE_USER_STOP**. If **fail** is supplied to nag_opt_check_deriv (e04hcc), **fail.errnum** will be set to the user's setting of **comm** → **flag**.

first – Nag_Boolean *Input*

On entry: will be set to **Nag_True** on the first call to **objfun** and **Nag_False** for all subsequent calls.

nf – Integer *Input*

On entry: 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 (e04hcc) these pointers may be allocated memory by the user and initialized with various quantities for use by **objfun** when called from nag_opt_check_deriv (e04hcc).

The array **x** must **not** be changed by **objfun**.

3: **x**[**n**] – const double *Input*

On entry: **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.

4: **objf** – double * *Output*

On exit: 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**.

5: **g**[**n**] – double *Output*

On exit: 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 $j = 1, 2, \ldots, n$.

6: **comm** – Nag_Comm * *Input/Output*

On entry/on exit: structure containing pointers for communication with the user-defined function; 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_check_deriv (e04hcc); **comm** will then be declared internally for use in calls to **objfun**.

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

The NAG error parameter, see the Essential Introduction.

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.

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 **objfun** is calculating them correctly.

NE_INT_ARG_LT

On entry, **n** must not be less than 1: **n** = $\langle value \rangle$.

NE_USER_STOP

User requested termination, user flag value = $\langle value \rangle$.

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**. The check on **objfun** will not have been completed.

7 Accuracy

fail is set to **NE_DERIV_ERRORS** if

$$\left(v_k - g^{\mathrm{T}} p_k\right)^2 \geq h \times \left(\left(g^{\mathrm{T}} p_k\right)^2 + 1\right)$$

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

8 Further Comments

The user-defined function **objfun** is called three times.

Before using nag_opt_check_deriv (e04hcc) to check the calculation of first derivatives, the user should be confident that **objfun** is calculating F correctly. The usual way of checking the calculation of the function is to compare values of $F(x)$ calculated by **objfun** at non-trivial points x with values calculated independently. ('Non-trivial' means that, as when setting x before calling nag_opt_check_deriv (e04hcc), co-ordinates such as 0.0 or 1.0 should be avoided.)

9 Example

Suppose that it is intended to use nag_opt_nlp_solve (e04wdc) 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 **objfun**. (The test of whether **comm** $\rightarrow$ **flag** $\neq 0$ in **objfun** is present for when **objfun** is called by nag_opt_nlp_solve (e04wdc). nag_opt_check_deriv (e04hcc) will always call **objfun** with **comm** $\rightarrow$ **flag** set to 2.)

9.1 Program Text

```
/* nag_opt_check_deriv (e04hcc) Example Program.
 *
 * Copyright 1991 Numerical Algorithms Group.
 *
 * Mark 2, 1991.
 * Mark 7 revised, 2001.
 * Mark 8 revised, 2004.
 *
 */

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

#ifdef __cplusplus
extern "C" {
#endif
  static void objfun(Integer n, double x[], double *f,
                     double g[], Nag_Comm *comm);
#ifdef __cplusplus
}
#endif

#define NMAX 4
static void objfun(Integer n, double x[], double *objf,
                   double g[], Nag_Comm *comm)
{
  /* 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;
    }
}                               /* objfun */

int main(void)
{
  Integer exit_status=0, i, n;
  NagError fail;
  double *g=0, objf, *x=0;

  INIT_FAIL(fail);

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

  n = NMAX;
  if (n>=1)
    {
      if ( !( x = NAG_ALLOC(n, double)) ||
           !( g = NAG_ALLOC(n, double)) )
        {
          Vprintf("Allocation failure\n");
          exit_status = -1;
          goto END;
        }
    }
  else
    {
      Vprintf("Invalid n.\n");
      exit_status = 1;
      return exit_status;
    }
  x[0] = 1.46;
  x[1] = -0.82;
  x[2] = 0.57;
  x[3] = 1.21;

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

  /* Call derivative checker */
  /* nag_opt_check_deriv (e04hcc).
   * Derivative checker for use with nag_opt_bounds_deriv
   * (e04kbc)
   */
  nag_opt_check_deriv(n, objfun, x, &objf, g, NAGCOMM_NULL, &fail);
  if (fail.code != NE_NOERROR)
    {
      Vprintf("Error from nag_opt_check_deriv (e04hcc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }

  Vprintf("\nFirst 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\n");
  for (i = 0; i < n; ++i)
    Vprintf("  %9.3e ", g[i]);
  Vprintf("\n");
 END:
  if (x) NAG_FREE(x);
  if (g) NAG_FREE(g);
  return exit_status;
```

```
}
```

9.2 Program Data

None.

9.3 Program Results

```
nag_opt_check_deriv (e04hcc) Example 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
```