NAG Toolbox

nag_opt_check_deriv (e04hc)

1 Purpose

nag_opt_check_deriv (e04hc) checks that a function for evaluating an objective function and its first derivatives produces derivative values which are consistent with the function values calculated.

2 Syntax

```
[f, g, iw, w, ifail] = nag_opt_check_deriv(funct, x, iw, w, 'n', n)
[f, g, iw, w, ifail] = e04hc(funct, x, iw, w, 'n', n)
```

Note: the interface to this routine has changed since earlier releases of the toolbox:

At Mark 22: *liw* and *lw* were removed from the interface.

3 Description

Routines for minimizing a function of several variables may require you to supply a function to evaluate the objective function $F(x_1, x_2, \ldots, x_n)$ and its first derivatives. nag_opt_check_deriv (e04hc) is designed to check the derivatives calculated by such user-supplied functions . As well as the function to be checked (**funct**), you must supply a point $x = (x_1, x_2, \ldots, x_n)^{\mathrm{T}}$ at which the check will be made. Note that nag_opt_check_deriv (e04hc) checks functions of the form required for nag_opt_bounds_mod_deriv_comp (e04kd) and nag_opt_bounds_mod_deriv2_comp (e04lb).

nag_opt_check_deriv (e04hc) first calls **funct** to evaluate F and its first derivatives $g_j = \dfrac{\partial F}{\partial x_j}$, 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 Parameters

5.1 Compulsory Input Parameters

1: **funct** – SUBROUTINE, supplied by the user.

funct must evaluate the function and its first derivatives at a given point. (The minimization functions mentioned in Section 3 gives you the option of resetting arguments of **funct** to cause the minimization process to terminate immediately. nag_opt_check_deriv (e04hc) will also terminate immediately, without finishing the checking process, if the argument in question is reset.)

```
[iflag, fc, gc, iw, w] = funct(iflag, n, xc, iw, w)
```

Input Parameters

1: **iflag** – INTEGER

Will be set to 2.

2: **n** – INTEGER

The number n of variables.

3: **xc**(**n**) – REAL (KIND=nag_wp) array

The point x at which F and its derivatives are required.

4: **iw**(liw) – INTEGER array

5: **w**(lw) – REAL (KIND=nag_wp) array

These arguments are present so that **funct** will be of the form required by the minimization functions mentioned in Section 3. **funct** is called with nag_opt_check_deriv (e04hc)'s arguments **iw**, liw, **w**, lw as these arguments. If the advice given in the minimization function documents is being followed, you will have no reason to examine or change any elements of **iw** or **w**. In any case, **funct must not change** the first $3 \times$ **n** elements of **w**.

Output Parameters

1: **iflag** – INTEGER

If you reset **iflag** to a negative number in **funct** and return control to nag_opt_check_deriv (e04hc), nag_opt_check_deriv (e04hc) will terminate immediately with **ifail** set to your setting of **iflag**.

2: **fc** – REAL (KIND=nag_wp)

Unless **funct** resets **iflag**, **fc** must be set to the value of the function F at the current point x.

3: **gc**(**n**) – REAL (KIND=nag_wp) array

Unless **funct** resets **iflag**, **gc**(j) must be set to the value of the first derivative $\dfrac{\partial F}{\partial x_j}$ at the point x, for $j = 1, 2, \ldots, n$.

4: **iw**(liw) – INTEGER array

5: **w**(lw) – REAL (KIND=nag_wp) array

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

x(j), for $j = 1, 2, \ldots, n$, must be set to the coordinates of a suitable point at which to check the derivatives calculated by **funct**. '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.

3: **iw**(liw) – INTEGER array

liw, the dimension of the array, must satisfy the constraint $liw \geq 1$.

This array is in the argument list so that it can be used by other library functions for passing integer quantities to **funct**. It is not examined or changed by nag_opt_check_deriv (e04hc). Generally, you must provide an array **iw** but are advised not to use it.

4: **w**(lw) – REAL (KIND=nag_wp) array

lw, the dimension of the array, must satisfy the constraint $lw \geq 3 \times \mathbf{n}$.

Constraint: $lw \geq 3 \times \mathbf{n}$.

5.2 Optional Input Parameters

1: **n** – INTEGER

Default: the dimension of the array **x**.

The number n of independent variables in the objective function.

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

5.3 Output Parameters

1: **f** – REAL (KIND=nag_wp)

Unless you set **iflag** negative in the first call of **funct**, **f** contains the value of the objective function $F(x)$ at the point given by you in **x**.

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

Unless you set **iflag** negative in the first call of **funct**, **g**(j) contains the value of the derivative $\dfrac{\partial F}{\partial x_j}$ at the point given in **x**, as calculated by **funct**, for $j = 1, 2, \ldots, n$.

3: **iw**(liw) – INTEGER array

4: **w**(lw) – REAL (KIND=nag_wp) array

$lw = 3 \times \mathbf{n}$.

Communication array, used to store information between calls to nag_opt_check_deriv (e04hc).

5: **ifail** – INTEGER

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

6 Error Indicators and Warnings

Note: nag_opt_check_deriv (e04hc) may return useful information for one or more of the following detected errors or warnings.

Errors or warnings detected by the function:

ifail < 0 (*warning*)

A negative value of **ifail** indicates an exit from nag_opt_check_deriv (e04hc) because you have set **iflag** negative in **funct**. The setting of **ifail** will be the same as your setting of **iflag**. The check on **funct** will not have been completed.

ifail $= 1$

On entry, $\mathbf{n} < 1$,
or $liw < 1$,
or $lw < 3 \times \mathbf{n}$.

ifail $= 2$ (*warning*)

You should check carefully the derivation and programming of expressions for the derivatives of $F(x)$, because it is very unlikely that **funct** is calculating them correctly.

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

ifail is set to 2 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 (x02aj).

8 Further Comments

funct is called 3 times.

Before using nag_opt_check_deriv (e04hc) to check the calculation of first derivatives, you should be confident that **funct** is calculating F correctly. The usual way of checking the calculation of the function is to compare values of $F(x)$ calculated by **funct** at nontrivial points x with values calculated independently. ('Non-trivial' means that, as when setting x before calling nag_opt_check_deriv (e04hc), coordinates such as 0.0 or 1.0 should be avoided.)

nag_opt_check_deriv (e04hc) only checks the derivatives calculated when **iflag** $= 2$. So, if **funct** is intended for use in conjunction with a minimization function which may set **iflag** to 1, you must check that, for given settings of the **xc**(j), **funct** produces the same values for the **gc**(j) when **iflag** is set to 1 as when **iflag** is set to 2.

9 Example

Suppose that it is intended to use nag_opt_bounds_mod_deriv_comp (e04kd) 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 **funct**. (The tests of whether **iflag** $= 0$ or 1 in **funct** are present ready for when **funct** is called by nag_opt_bounds_mod_deriv_comp (e04kd). nag_opt_check_deriv (e04hc) will always call **funct** with **iflag** set to 2.)

9.1 Program Text

```
function e04hc_example

fprintf('e04hc example results\n\n');

x  = [1.46; -0.82;  0.57;  1.21];
fprintf('The test point is: %8.2f%8.2f%8.2f%8.2f\n',x);

iw = [nag_int(0)];
w  = zeros(12,1);

wstat = warning;
warning('OFF');
[f, g, iw, w, ifail] = e04hc(@funct, x, iw, w);
if (ifail>=0)
  if (ifail==0)
    fprintf('\n1st derivatives are consistent with function values\n');
```

```
  else
    fprintf('\nProbable error in calculation of 1st derivatives\n');
  end
  fprintf('At test point, f = %7.3f\n',f)
  fprintf('               g = %7.3f %7.3f %7.3f %7.3f\n',g)
end

warning(wstat);

function [iflag, fc, gc, iw, w] = funct(iflag, n, xc, iw, liw, w, lw)
  gc = zeros(n, 1);
  fc = 0;
  a = xc(1) + 10*xc(2);
  b = xc(3) -    xc(4);
  c = xc(2) -  2*xc(3);
  d = xc(1) -    xc(4);
  if (iflag ~= 1)
    fc = a^2 + 5*b^2 + c^4 + 10*d^4;
  end
  if (iflag ~= 0)
    gc(1) =   2*a + 40*d^3;
    gc(2) =  20*a +  4*c^3;
    gc(3) =  10*b -  8*c^3;
    gc(4) = -10*b - 40*d^3;
  end
```

9.2 Program Results

```
     e04hc example results

The test point is:     1.46   -0.82    0.57    1.21

1st derivatives are consistent with function values
At test point, f =  62.273
               g = -12.855 -164.918  53.836   5.775
```