NAG Library Function Document

nag_opt_one_var_deriv (e04bbc)

1 Purpose

nag_opt_one_var_deriv (e04bbc) searches for a minimum, in a given finite interval, of a continuous function of a single variable, using function and first derivative values. The method (based on cubic interpolation) is intended for functions which have a continuous first derivative (although it will usually work if the derivative has occasional discontinuities).

2 Specification

```c
#include <nag.h>
#include <nageo4.h>
void nag_opt_one_var_deriv (        
    void (*funct)(double xc, double *fc, double *gc, Nag_Comm *comm),
    double e1, double e2, double *a, double *b, Integer max_fun, double *x,
    double *f, double *g, Nag_Comm *comm, NagError *fail)
```

3 Description

nag_opt_one_var_deriv (e04bbc) is applicable to problems of the form:

\[ \text{Minimize } F(x) \text{ subject to } a \leq x \leq b \]

when the first derivative \( dF/dx \) can be calculated. nag_opt_one_var_deriv (e04bbc) normally computes a sequence of \( x \) values which tend in the limit to a minimum of \( F(x) \) subject to the given bounds. It also progressively reduces the interval \( [a, b] \) in which the minimum is known to lie. It uses the safeguarded quadratic-interpolation method described in Gill and Murray (1973).

You must supply a function `funct` to evaluate \( F(x) \) and its first derivative. The arguments \( e1 \) and \( e2 \) together specify the accuracy:

\[ \text{Tol}(x) = e1 \times |x| + e2 \]

to which the position of the minimum is required. Note that `funct` is never called at any point which is closer than \( \text{Tol}(x) \) to a previous point.

If the original interval \( [a, b] \) contains more than one minimum, nag_opt_one_var_deriv (e04bbc) will normally find one of the minima.

4 References

Gill P E and Murray W (1973) Safeguarded steplength algorithms for optimization using descent methods NPL Report NAC 37 National Physical Laboratory

5 Arguments

1: \textbf{funct} – function, supplied by the user

\textit{External Function}

`funct` must calculate the values of \( F(x) \) and \( dF/dx \) at any point \( x \) in \( [a, b] \).

The specification of `funct` is:

```c
void funct (double xc, double *fc, double *gc, Nag_Comm *comm)
```
### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xc</td>
<td>double</td>
<td><em>Input</em>&lt;br&gt;<code>On entry:</code> The point at which the values of $F$ and $dF/dx$ are required.</td>
</tr>
<tr>
<td>fc</td>
<td>double *</td>
<td><em>Output</em>&lt;br&gt;<code>On exit:</code> The value of the function $F$ at the current point $x$.</td>
</tr>
<tr>
<td>gc</td>
<td>double *</td>
<td><em>Output</em>&lt;br&gt;<code>On exit:</code> The value of the first derivative $dF/dx$ at the current point $x$.</td>
</tr>
</tbody>
</table>
| comm | Nag_Comm * | Pointer to structure of type Nag_Comm; the following members are relevant to *funct*.<br>- **first** - Nag_Boolean<br>`On entry:` Will be set to Nag_TRUE on the first call to *funct* and Nag_FALSE for all subsequent calls.<br>- **nf** - Integer<br>`On entry:` The number of calls made to *funct* so far.<br>- **user** - double *
- **iuser** - Integer *
- **p** - Pointer<br>The type Pointer will be void * with a C compiler that defines void * and char *
* otherwise. Before calling nag_opt_one_var_deriv (e04bbc) these pointers may be allocated memory and initialized with various quantities for use by *funct* when called from nag_opt_one_var_deriv (e04bbc). |

### Notes
- *funct* should be tested separately before being used in conjunction with nag_opt_one_var_deriv (e04bbc).
- **e1** - double
  - *Input*<br>`On entry:` The relative accuracy to which the position of a minimum is required. (Note that since e1 is a relative tolerance, the scaling of $x$ is automatically taken into account.)
  - It is recommended that e1 should be no smaller than $2\epsilon$, and preferably not much less than $\sqrt{\epsilon}$, where $\epsilon$ is the *machine precision*.
  - If e1 is set to a value less than $\epsilon$, its value is ignored and the default value of $\sqrt{\epsilon}$ is used instead. In particular, you may set e1 = 0.0 to ensure that the default value is used.
- **e2** - double
  - *Input*<br>`On entry:` The absolute accuracy to which the position of a minimum is required. It is recommended that e2 should be no smaller than $2\epsilon$.
  - If e2 is set to a value less than $\epsilon$, its value is ignored and the default value of $\sqrt{\epsilon}$ is used instead. In particular, you may set e2 = 0.0 to ensure that the default value is used.
- **a** - double *
  - *Input/Output*<br>`On entry:` The lower bound a of the interval containing a minimum.<br>`On exit:` An improved lower bound on the position of the minimum.
- **b** - double *
  - *Input/Output*<br>`On entry:` The upper bound b of the interval containing a minimum.
On exit: an improved upper bound on the position of the minimum.

Constraint: $b > a + e2$.

Note that the value $e2 = \sqrt{c}$ applies here if $e2 < \epsilon$ on entry to nag_opt_one_var_deriv (e04bbc).

6: max_fun – Integer

On entry: the maximum number of calls to funct which you are prepared to allow.

The number of calls to funct actually made by nag_opt_one_var_deriv (e04bbc) may be determined by supplying a non-NULL argument comm (see below) and examining the structure member comm->nf on exit.

Constraint: $\max\text{\_fun} \geq 2$.

(Few problems will require more than 20 function calls.)

7: x – double *

On exit: the estimated position of the minimum.

8: f – double *

On exit: the value of $F$ at the final point $x$.

9: g – double *

On exit: the value of the first derivative $dF/dx$ at the final point $x$.

10: comm – Nag_Comm *

Note: comm is a NAG defined type (see Section 3.2.1.1 in the Essential Introduction).

On entry/exit: structure containing pointers for communication to user-supplied functions; see the above description of funct for details. The number of times the function funct was called is returned in the member comm->nf.

If you do not need to make use of this communication feature, the null pointer NAGCOMM_NULL may be used in the call to nag_opt_one_var_deriv (e04bbc); comm will then be declared internally for use in calls to user-supplied functions.

11: fail – NagError *

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_2_REAL_ARG_GE

On entry, $a + e2 = \langle value \rangle$ while $b = \langle value \rangle$. These arguments must satisfy $a + e2 < b$.

NE_INT_ARG_LT

On entry, max_fun must not be less than 2: max_fun = $\langle value \rangle$.

NW_MAX_FUN

The maximum number of function calls, $\langle value \rangle$, have been performed.

This may have happened simply because max_fun was set too small for a particular problem, or may be due to a mistake in the user-supplied function, funct. If no mistake can be found in funct, restart nag_opt_one_var_deriv (e04bbc) (preferably with the values of $a$ and $b$ given on exit from the previous call to nag_opt_one_var_deriv (e04bbc)).
7 Accuracy

If $F(x)$ is $\delta$-unimodal for some $\delta < Tol(x)$, where $Tol(x) = e1 \times |x| + e2$, then, on exit, $x$ approximates the minimum of $F(x)$ in the original interval $[a, b]$ with an error less than $3 \times Tol(x)$.

8 Parallelism and Performance

Not applicable.

9 Further Comments

Timing depends on the behaviour of $F(x)$, the accuracy demanded, and the length of the interval $[a, b]$. Unless $F(x)$ and $dF/dx$ can be evaluated very quickly, the run time will usually be dominated by the time spent in funct.

If $F(x)$ has more than one minimum in the original interval $[a, b]$, nag_opt_one_var_deriv (e04bbc) will determine an approximation $x$ (and improved bounds $a$ and $b$) for one of the minima.

If nag_opt_one_var_deriv (e04bbc) finds an $x$ such that $F(x - \delta_1) > F(x) < F(x + \delta_2)$ for some $\delta_1, \delta_2 \geq Tol(x)$, the interval $[x - \delta_1, x + \delta_2]$ will be regarded as containing a minimum, even if $F(x)$ is less than $F(x - \delta_1)$ and $F(x + \delta_2)$ only due to rounding errors in the user-supplied function. Therefore funct should be programmed to calculate $F(x)$ as accurately as possible, so that nag_opt_one_var_deriv (e04bbc) will not be liable to find a spurious minimum. (For similar reasons, $dF/dx$ should be evaluated as accurately as possible.)

10 Example

A sketch of the function

$$ F(x) = \frac{\sin x}{x} $$

shows that it has a minimum somewhere in the range $[3.5, 5.0]$. The example program below shows how nag_opt_one_var_deriv (e04bbc) can be used to obtain a good approximation to the position of a minimum.

10.1 Program Text

/* nag_opt_one_var_deriv (e04bbc) Example Program. *
* Copyright 2014 Numerical Algorithms Group.
* Mark 7 revised, 2001.
*/
#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <math.h>
#include <nagf04.h>

#ifdef __cplusplus
extern "C" {
#endif
static void NAG_CALL funct(double xc, double *fc, double *gc, Nag_Comm *comm);
#ifdef __cplusplus
}
#endif

static void NAG_CALL funct(double xc, double *fc, double *gc, Nag_Comm *comm)
{
    if (comm->user[0] == -1.0)
printf("(User-supplied callback funct, first invocation.)\n");
comm->user[0] = 0.0;
}
*fc = \sin(xc) / xc;
*gc = (\cos(xc) - *fc) / xc;
} /* funct */

int main(void)
{
    static double ruser[1] = {-1.0};
    Integer exit_status = 0, max_fun;
    NagError fail;
    Nag_Comm comm;
    double a, b, e1, e2, f, g, x;
    INIT_FAIL(fail);
    printf("nag_opt_one_var_deriv (e04bbc) Example Program Results\n\n");
    /* For communication with user-supplied functions: */
    comm.user = ruser;
    /* e1 and e2 are set to zero so that nag_opt_one_var_no_deriv (e04abc) will
     * reset them to their default values */
    e1 = 0.0;
    e2 = 0.0;
    /* The minimum is known to lie in the range (3.5, 5.0) */
    a = 3.5;
    b = 5.0;
    /* Allow 30 calls of funct */
    max_fun = 30;
    /* nag_opt_one_var_deriv (e04bbc).
     * Minimizes a function of one variable, requires first
     * derivatives */
    nag_opt_one_var_deriv(funct, e1, e2, &a, &b, max_fun, &x, &f, &g, &comm,
                         &fail);
    if (fail.code != NE_NOERROR)
    {
        printf("Error from nag_opt_one_var_deriv (e04bbc).\n%s\n",
               fail.message);
        exit_status = 1;
        goto END;
    }
    printf("The minimum lies in the interval %7.5f to %7.5f.\n", a, b);
    printf("Its estimated position is %7.5f,\n", x);
    printf("where the function value is %13.4e\n", f);
    printf("and the gradient is %13.4e.\n", g);
    printf("%1"NAG_IFMT" function evaluations were required.\n", comm.nf);
END:
    return exit_status;
}

10.2 Program Data
None.
10.3 Program Results

nag_opt_one_var_deriv (e04bbc) Example Program Results

(User-supplied callback funct, first invocation.)
The minimum lies in the interval 4.49341 to 4.49341.
Its estimated position is 4.49341,
where the function value is -2.1723e-01
and the gradient is -3.7679e-16.
6 function evaluations were required.