NAG Library Function Document

nag_opt_one_var_no_deriv (e04abc)

1 Purpose

nag_opt_one_var_no_deriv (e04abc) searches for a minimum, in a given finite interval, of a continuous function of a single variable, using function values only. The method (based on quadratic 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 <nage04.h>

void nag_opt_one_var_no_deriv (  
    void (*funct)(double xc, double *fc, Nag_Comm *comm),  
    double e1, double e2, double *a, double *b, Integer max_fun, double *x,  
    double *f, Nag_Comm *comm, NagError *fail)
```

3 Description

nag_opt_one_var_no_deriv (e04abc) is applicable to problems of the form:

Minimize \( F(x) \) subject to \( a \leq x \leq b \).

It 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 \( F(x) \). The arguments `e1` and `e2` together specify the accuracy

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

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

If the original interval \( [a, b] \) contains more than one minimum, nag_opt_one_var_no_deriv (e04abc) 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: `funct` – function, supplied by the user

`funct` must calculate the value of \( F(x) \) at any point \( x \) in \( [a, b] \).

The specification of `funct` is:

```c
void funct (double xc, double *fc, Nag_Comm *comm)
```
**Input**

1: \( \textbf{xc} \) – double

\( \text{On entry: } x \), the point at which the value of \( F(x) \) is required.

2: \( \textbf{fc} \) – double *

\( \text{On exit: } \) the value of the function \( F \) at the current point \( x \).

3: \( \textbf{comm} \) – Nag_Comm *

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

   - \( \textbf{first} \) – Nag_Boolean

      \( \text{On entry: } \) will be set to Nag_TRUE on the first call to \textbf{funct} and Nag_FALSE for all subsequent calls.

   - \( \textbf{nf} \) – Integer

      \( \text{On entry: } \) the number of calls made to \textbf{funct} so far.

   - \( \textbf{user} \) – double *
   - \( \textbf{iuser} \) – Integer *
   - \( \textbf{p} \) – Pointer

The type Pointer will be \texttt{void *} with a C compiler that defines \texttt{void *} and \texttt{char *} otherwise. Before calling nag_opt_one_var_no_deriv (e04abc) these pointers may be allocated memory and initialized with various quantities for use by \textbf{funct} when called from nag_opt_one_var_no_deriv (e04abc).

**Note:** \textbf{funct} should be tested separately before being used in conjunction with nag_opt_one_var_no_deriv (e04abc).

**Input**

2: \( \textbf{e1} \) – double

\( \text{On entry: } \) the relative accuracy to which the position of a minimum is required. (Note that since \( \textbf{e1} \) is a relative tolerance, the scaling of \( x \) is automatically taken into account.)

It is recommended that \( \textbf{e1} \) should be no smaller than \( 2 \varepsilon \), and preferably not much less than \( \sqrt{\varepsilon} \), where \( \varepsilon \) is the machine precision.

If \( \textbf{e1} \) is set to a value less than \( \varepsilon \), its value is ignored and the default value of \( \sqrt{\varepsilon} \) is used instead. In particular, you may set \( \textbf{e1} = 0.0 \) to ensure that the default value is used.

3: \( \textbf{e2} \) – double

\( \text{On entry: } \) the absolute accuracy to which the position of a minimum is required. It is recommended that \( \textbf{e2} \) should be no smaller than \( 2 \varepsilon \).

If \( \textbf{e2} \) is set to a value less than \( \varepsilon \), its value is ignored and the default value of \( \sqrt{\varepsilon} \) is used instead. In particular, you may set \( \textbf{e2} = 0.0 \) to ensure that the default value is used.

**Input/Output**

4: \( \textbf{a} \) – double *

\( \text{On entry: } \) the lower bound \( a \) of the interval containing a minimum.

\( \text{On exit: } \) an improved lower bound on the position of the minimum.

5: \( \textbf{b} \) – double *

\( \text{On entry: } \) the upper bound \( b \) of the interval containing a minimum.

\( \text{On exit: } \) an improved upper bound on the position of the minimum.

**Constraint:** \( \textbf{b} > \textbf{a} + \textbf{e2} \).
Note that the value $e^2 = \sqrt{e}$ applies here if $e^2 < \epsilon$ on entry to nag_opt_one_var_no_deriv (e04abc).

6: max_fun – Integer  

*Input*

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

The number of evaluations actually performed by nag_opt_one_var_no_deriv (e04abc) may be determined by supplying a non-NULL argument comm (see below) and examining the structure member comm->nf on exit.

*Constraint: max_fun \geq 3.*

(Few problems will require more than 30 function evaluations.)

7: x – double *  

*Output*

On exit: the estimated position of the minimum.

8: f – double *  

*Output*

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

9: comm – Nag_Comm *  

*Input/Output*

*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_no_deriv (e04abc); comm will then be declared internally for use in calls to user-supplied functions.

10: fail – NagError *  

*Input/Output*

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

6  Error Indicators and Warnings

**NE_2_REAL_ARG_GE**

On entry, $a + e^2 = \langle value \rangle$ while $b = \langle value \rangle$. These arguments must satisfy $a + e^2 < b$.

**NE_INT_ARG_LT**

On entry, max_fun must not be less than 3: 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_no_deriv (e04abc) (preferably with the values of $a$ and $b$ given on exit from the previous call to nag_opt_one_var_no_deriv (e04abc)).

7  Accuracy

If $F(x)$ is $\delta$-unimodal for some $\delta < Tol(x)$, where $Tol(x) = e^1 \times |x| + e^2$, 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 \( F(x) \) can be evaluated very quickly, the run time will usually be dominated by the time spent in \( \text{funct} \).

If \( F(x) \) has more than one minimum in the original interval \([a, b]\), \( \text{nag_opt_one_var_no_deriv (e04abc)} \) will determine an approximation \( x \) (and improved bounds \( a \) and \( b \)) for one of the minima.

If \( \text{nag_opt_one_var_no_deriv (e04abc)} \) finds an \( x \) such that \( F(x - \delta_1) > F(x) < F(x + \delta_2) \) for some \( \delta_1, \delta_2 \geq T0l(x) \), 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 \( \text{funct} \) should be programmed to calculate \( F(x) \) as accurately as possible, so that \( \text{nag_opt_one_var_no_deriv (e04abc)} \) will not be liable to find a spurious minimum.

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 \( \text{nag_opt_one_var_no_deriv (e04abc)} \) can be used to obtain a good approximation to the position of a minimum.

10.1 Program Text
/* \text{nag_opt_one_var_no_deriv (e04abc)} Example Program. */
/* * Copyright 2014 Numerical Algorithms Group. *
* Mark 7 revised, 2001.
* Mark 8 revised, 2004. *
*/
#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <math.h>
#include <nage04.h>

#ifdef __cplusplus
extern "C" {
#endif

static void NAG_CALL funct(double xc, double *fc, Nag_Comm *comm);

#ifdef __cplusplus
}
#endif
static void NAG_CALL funct(double xc, double *fc, 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;
}
/* 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, x;

    INIT_FAIL(fail);

    printf(
        "nag_opt_one_var_no_deriv (e04abc) 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_no_deriv (e04abc).
     * Minimizes a function of one variable, using function values only.
     */
    nag_opt_one_var_no_deriv(funct, e1, e2, &a, &b, max_fun, &x, &f, &comm, &fail);
    if (fail.code != NE_NOERROR)
    {
        printf("Error from nag_opt_one_var_no_deriv (e04abc)\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("%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_no_deriv (e04abc) 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.
10 function evaluations were required.