nag_opt_simplex (e04ccc)

1. Purpose

nag_opt_simplex (e04ccc) minimizes a general function $F(x)$ of n independent variables $x = (x_1, x_2, \ldots, x_n)^T$ by the Simplex method. No derivatives are required.

2. Specification

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

void nag_opt_simplex(Integer n,
            void (*funct)(Integer n, double *xc, double *fc, Nag_Comm *comm),
            double x[], double *fmin, Nag_E04_Opt *options, Nag_Comm *user_comm,
            NagError *fail)
```

3. Description

nag_opt_simplex finds an approximation to a minimum of a function $F(x)$ of n variables. The user must supply a function to calculate the value of $F(x)$ for any set of values of the variables.

The method is iterative. A simplex of $n+1$ points is set up in the dimensional space of the variables (for example, in 2 dimensions the simplex is a triangle) under the assumption that the problem has been scaled so that the values of the independent variables at the minimum are of order unity. The starting point provided by the user is the first vertex of the simplex, the remaining n vertices are generated internally (see Parkinson and Hutchinson (1972)). The vertex of the simplex with the largest function value is reflected in the centre of gravity of the remaining vertices and the function value at this new point is compared with the remaining function values. Depending on the outcome of this test the new point is accepted or rejected, a further expansion move may be made, or a contraction may be carried out. When no further progress can be made the sides of the simplex are reduced in length and the method is repeated.

The method tends to be slow, but it is robust and therefore very useful for functions that are subject to inaccuracies.

4. Parameters

n

Input: n, the number of independent variables.
Constraint: $\mathbf{n} \geq 1$.

funct

The function **funct**, supplied by the user, must calculate the value of $F(x)$ at any point x. (However, if the user does not wish to calculate the value of $F(x)$ at a particular x, there is the option of setting a parameter to cause nag_opt_simplex to terminate immediately.)

The specification of **funct** is:

```
void funct(Integer n, double xc[], double *fc, Nag_Comm *comm)
```

n

Input: n, the number of variables.

xc[n]

Input: x, the point at which the value of $F(x)$ is required.

fc

Output: the value of $F(x)$ at the current point x.

comm

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

flag – Integer

Input: **comm->flag** contains a non-negative number.

Output: if **funct** resets **comm->flag** to some negative number then nag_opt_simplex will terminate immediately with the error indicator **NE_USER_STOP**. If **fail** is supplied to nag_opt_simplex, **fail.errnum** will be set to the user's setting of **comm->flag**.

first – Boolean

Input: will be set to **TRUE** on the first call to **funct** and **FALSE** for all subsequent calls.

nf – Integer

Input: the number of calls made to **funct** so far.

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_simplex these pointers may be allocated memory by the user and initialised with various quantities for use by **funct** when called from nag_opt_simplex.

Note: **funct** should be tested separately before being used in conjunction with nag_opt_simplex. The array **xc** must **not** be changed within **funct**.

x[n]

Input: a guess at the position of the minimum. Note that the problem should be scaled so that the values of the variables $x_1, x_2, \ldots, x_n$ are of order unity.

Output: the value of x corresponding to the function value returned in **fmin**.

fmin

Output: the lowest function value found.

options

Input/Output: a pointer to a structure of type Nag_E04_Opt whose members are optional parameters for nag_opt_simplex. These structure members offer the means of adjusting some of the parameter values of the algorithm and on output will supply further details of the results. A description of the members of **options** is given below in Section 7.

If any of these optional parameters are required then the structure **options** should be declared and initialised by a call to nag_opt_init (e04xxc) and supplied as an argument to nag_opt_simplex. However, if the optional parameters are not required the NAG defined null pointer, `E04_DEFAULT`, can be used in the function call.

comm

Input/Output: structure containing pointers for communication to user-supplied functions; see the above description of **funct** 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_simplex; **comm** will then be declared internally for use in calls to user-supplied functions.

fail

The NAG error parameter, see the Essential Introduction to the NAG C Library.
Users are recommended to declare and initialise **fail** and set **fail.print** = **TRUE** for this function.

4.1. Description of Printed Output

Intermediate and final results are printed out by default. The level of printed output can be controlled by the user with the option **print_level** (see Section 7.2.).The default print level of **Nag_Soln_Iter** provides a single line of output at each iteration and the final result.
The line of results printed at each iteration gives:

`Itn`	the current iteration number k.
`Nfun`	the cumulative number of calls to **lsqfun**.
`Objective`	the current value of the objective function, $F(x^{(k)})$.
`Norm g`	the Euclidean norm of the gradient of $F(x^{(k)})$.
`Norm x`	the Euclidean norm of $x^{(k)}$.
`Norm(x(k-1)-x(k))`	the Euclidean norm of $x^{(k-1)} - x^{(k)}$.
`Step`	the step $\alpha^{(k)}$ taken along the computed search direction $p^{(k)}$.

The printout of the final result consists of:

`x`	the final point x^*.
`Function value`	the value of $F(x^*)$.

5. Comments

A list of possible error exits and warnings from nag_opt_simplex is given in Section 8.

5.1. Accuracy

On a successful exit the accuracy will be as defined by **options.optim_tol** (see Section 7.2.).

6. Example 1

A simple program to locate a minimum of the function:

$$F = e^{x_1}(4x_1^2 + 2x_2^2 + 4x_1x_2 + 2x_2 + 1).$$

The program uses $(-1.0, 1.0)$ as the initial guess at the position of the minimum.

This example shows the simple use of nag_opt_simplex where default values are used for all optional parameters. An example showing the use of optional parameters is given in Section 12. There is however only one example program file, the main program of which calls both examples. The main program and example 1 are given below.

6.1. Example Text

```
/* nag_opt_simplex(e04ccc) Example Program
 *
 * Copyright 1996 Numerical Algorithms Group.
 *
 * Mark 4, 1996.
 */

#include <nag.h>
#include <math.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nage04.h>
#include <nagx02.h>

#ifdef NAG_PROTO
```

```
static void funct(Integer n, double *xc, double *fc, Nag_Comm *comm);
static void monit(const Nag_Search_State *st, Nag_Comm *comm);
static void ex1(void);
static void ex2(void);
#else
static void funct();
static void monit();
static void ex1();
static void ex2();
#endif

/* Table of constant values */
static Integer c__3 = 3;

main()
{
  /* Two examples are called, ex1() which uses the
   * default settings to solve the problem and
   * ex2() which solves the same problem with
   * some optional parameters set by the user.
   */

  Vprintf("e04ccc Example Program Results.\n");
  ex1();
  ex2();
  exit(EXIT_SUCCESS);
}

static void ex1()
{
  double objf;
  double x[2];

  Integer n;

  Vprintf("\ne04ccc example 1: no option setting.\n");

  n = 2;
  /* Set up the starting point */
  x[0] = 0.4;
  x[1] = -0.8;

  e04ccc(n, funct, x, &objf, E04_DEFAULT, NAGCOMM_NULL, NAGERR_DEFAULT);

}


#ifdef NAG_PROTO
static void funct(Integer n, double *xc, double *objf, Nag_Comm *comm)
#else
     static void funct(n, xc, objf, comm)
     Integer n;
     double *xc;
     double *objf;
     Nag_Comm *comm;
#endif
{
  *objf = exp(xc[0]) * (xc[0] * 4.0 * (xc[0] + xc[1]) +
                        xc[1] * 2.0 * (xc[1] + 1.0) + 1.0);
}
```

6.2 Program Data

None, but there is an example data file which contains the optional parameter values for example 2 below.

6.3 Program Results

```
e04ccc Example Program Results.

e04ccc example 1: no option setting.

Parameters to e04ccc
--------------------

Number of variables...........  2

optim_tol............... 1.05e-08    max_iter................      1500
print_level.........Nag_Soln_Iter    machine precision....... 1.11e-16
outfile.................   stdout

Results from e04ccc:
-------------------

Iteration results:

   Itn   Nfun    Fmin         Fmax
    1     8   2.9836e-02   1.4017e+00
    2    10   2.9836e-02   2.8134e-01
    3    12   2.9836e-02   1.1427e-01
    4    14   2.9836e-02   5.9673e-02
    5    16   8.4227e-03   4.2612e-02
    6    18   8.4227e-03   2.9836e-02
    7    19   8.4227e-03   2.1408e-02
    8    21   3.1325e-04   1.1706e-02
    9    23   3.1325e-04   8.4227e-03
   10    25   3.0314e-04   4.6988e-03
   11    27   3.0314e-04   8.3804e-04
   12    29   1.5662e-04   3.1325e-04
   13    31   3.9493e-05   3.0314e-04
   14    33   2.4900e-05   1.5662e-04
   15    35   2.4900e-05   3.9493e-05
   16    37   8.6529e-06   3.8842e-05
   17    39   8.6026e-06   2.4900e-05
   18    41   2.5852e-06   8.6529e-06
   19    43   2.5852e-06   8.6026e-06
   20    45   2.4239e-06   3.4565e-06
   21    47   3.9390e-07   2.5852e-06
   22    49   2.6414e-07   2.4239e-06
   23    51   2.6414e-07   4.5291e-07
   24    53   1.1839e-07   3.9390e-07
   25    55   9.3983e-08   2.6414e-07
   26    57   2.6672e-08   1.1839e-07
   27    59   2.6672e-08   9.3983e-08
   28    61   1.9199e-08   5.1588e-08
   29    63   9.7190e-09   2.6672e-08

Final solution:

Vector x
       5.0005e-01
      -1.0001e+00

Final Function value is   9.7190e-09
```

7. Optional Parameters

A number of optional input and output parameters to nag_opt_simplex are available through the structure argument **options**, type Nag_E04_Opt. A parameter may be selected by assigning an appropriate value to the relevant structure member; those parameters not selected will be assigned default values. If no use is to be made of any of the optional parameters the user should use the NAG defined null pointer, `E04_DEFAULT`, in place of **options** when calling nag_opt_simplex; the default settings will then be used for all parameters.

Before assigning values to **options** directly the structure **must** be initialised by a call to the function nag_opt_init (e04xxc). Values may then be assigned to the structure members in the normal C manner.

Option settings may also be read from a text file using the function nag_opt_read (e04xyc) in which case initialisation of the **options** structure will be performed automatically if not already done. Any subsequent direct assignment to the **options** structure must **not** be preceded by initialisation.

If an assignment of a function pointer in the **options** structure is required, this must be done directly in the calling program, it cannot be assigned using nag_opt_read (e04xyc).

7.1. Optional Parameter Checklist and Default Values

For easy reference, the following list shows the members of **options** which are valid for nag_opt_simplex together with their default values where relevant. The number ϵ is a generic notation for ***machine precision*** (see nag_machine_precision (X02AJC)).

`Boolean list`	**TRUE**
`Nag_PrintType print_level`	**Nag_Soln_Iter**
`char outfile[80]`	`stdout`
`void (*print_fun)()`	NULL
`Integer max_iter`	1500
`double optim_tol`	ϵ
`Integer iter`	
`Integer nf`	

7.2. Description of Optional Parameters

list – Boolean Default = **TRUE**

Input: if **options.list** = **TRUE** the parameter settings in the call to nag_opt_simplex will be printed.

print_level – **Nag_PrintType** Default = **Nag_Soln_Iter**

Input: the level of results printout produced by nag_opt_simplex. The following values are available.

Nag_NoPrint	No output.
Nag_Soln	The final solution only.
Nag_Iter	One line of output for each iteration.
Nag_Soln_Iter	The final solution and one line of output for each iteration.
Nag_Soln_Iter_Full	The final solution and detailed printout at each iteration.

Details of each level of results printout are described in Section 7.3.
Constraint: **options.print_level** = **Nag_NoPrint** or **Nag_Soln** or **Nag_Iter** or **Nag_Soln_Iter** or **Nag_Soln_Iter_Full**.

outfile – char[80] Default = `stdout`

Input: the name of the file to which results should be printed. If **options.outfile**[0] = '\0' then the `stdout` stream is used.

print_fun – pointer to function Default = NULL

Input: printing function defined by the user; the prototype of **print_fun** is

```
void (*print_fun)(const Nag_Search_State *st, Nag_Comm *comm);
```

See Section 7.3.1. below for further details.

max_iter – Integer Default = 1500

Input: the maximum number of iterations allowed before termination.
Constraint: **options.max_iter** > 0.

optim_tol – double Default = ϵ

Input: the accuracy in x to which the solution is required. If f_i, for $i = 1, 2, \ldots, n+1$, are the individual function values at the vertices of a simplex and f_m is the mean of these values, then termination will occur when

$$\sqrt{\frac{1}{n+1}\sum_{i=1}^{n+1}(f_i - f_m)^2} < \textbf{options.optim\_tol}.$$

Constraint: **options.optim_tol** $\geq \epsilon$.

iter – Integer

Output: the number of iterations which have been performed by nag_opt_simplex.

nf – Integer

Output: the number of times that **funct** has been called.

7.3. Description of Printed Output

The level of printed output can be controlled with the structure members **options.list** and **options.print_level** (see Section 7.2.). If **list** = **TRUE** then the parameter values to nag_opt_simplex are listed, whereas the printout of results is governed by the value of **print_level**. The default of **print_level** = **Nag_Soln_Iter** provides a single line of output at each iteration and the final result. This section describes all of the possible levels of results printout available from nag_opt_simplex.

When **options.print_level** = **Nag_Iter** or **Nag_Soln_Iter** a single line of output is produced on completion of each iteration, this gives the following values:

`Itn`	the current iteration number k.
`Nfun`	the cumulative number of calls made to **funct**.
`Fmin`	the smallest function value in the current simplex.
`Fmax`	the largest function value in the current simplex.

When **options.print_level** = **Nag_Soln_Iter_Full** more detailed results are given at each iteration. Additional values output are

`x`	the current point $x^{(k)}$.
`Simplex`	Vertices of the simplex with their corresponding vectors containing the positions of the current simplex.

If **options.print_level** = **Nag_Soln** or **Nag_Soln_Iter** or **Nag_Soln_Iter_Full** the final result is printed out. This consists of:

`x`	the final point x^*.
`Function value`	the value of $F(x^*)$.

If **options.print_level** = **Nag_NoPrint**, printout will be suppressed; the user can then print the final solution when nag_opt_simplex returns to the calling program.

7.3.1. Output of results via a user defined printing function

Users may also specify their own print function for output of iteration results and the final solution by use of the **options.print_fun** function pointer, which has prototype

```
void (*print_fun)(const Nag_Search_State *st, Nag_Comm *comm);
```

The rest of this section can be skipped if the default printing facilities provide the required functionality.

When a user defined function is assigned to **options.print_fun** this will be called in preference to the internal print function of nag_opt_simplex. Calls to the user defined function are again controlled by means of the **options.print_level** member. Information is provided through **st** and **comm**, the two structure arguments to **print_fun**. If **comm->it_prt = TRUE** then the results from the last iteration of nag_opt_simplex are in the following members of **st**:

n – Integer

the number of variables.

x – double *

points to the **n** memory locations holding the current point $x^{(k)}$.

fmin – double

holds the smallest function value in the current simplex.

fmax – double

holds the largest function value in the current simplex.

simplex– double *

points to the **(n+1)*n** memory locations. If we regard this pointer as pointing to a notional 2-D array then its **n**+1 rows contain the **n** position vectors of the vertices of the current simplex.

iter – Integer
k, the number of iterations performed by nag_opt_simplex.

nf – Integer
the cumulative number of calls made to **funct**.

The relevant members of the structure **comm** are:

it_prt – Boolean
will be **TRUE** when the print function is called with the result of the current iteration.

sol_prt – Boolean
will be **TRUE** when the print function is called with the final result.

user – double *
iuser – Integer *
p – Pointer
pointers for communication of user information. If used they must be allocated memory by the user either before entry to nag_opt_simplex or during a call to **funct** or **print_fun**. The type Pointer will be `void *` with a C compiler that defines `void *` and `char *` otherwise.

8. Error Indications and Warnings

NE_INT_ARG_LT
On entry, **n** must not be less than 1: **n** = ⟨*value*⟩.

NE_OPT_NOT_INIT
Options structure not initialised.

NE_BAD_PARAM
On entry, parameter **options.print_level** had an illegal value.

NE_INVALID_INT_RANGE_1
Value ⟨*value*⟩ given to **options.max_iter** is not valid. Correct range is **options.max_iter** > 0.

NE_INVALID_REAL_RANGE_E
Value ⟨*value*⟩ given to **options.optim_tol** is not valid. Correct range is **options.optim_tol** $\geq \epsilon$.

NE_NOT_APPEND_FILE
Cannot open file ⟨*string*⟩ for appending.

NE_NOT_CLOSE_FILE
Cannot close file ⟨*string*⟩.

NE_ALLOC_FAIL
Memory allocation failed.

NW_TOO_MANY_ITER
The maximum number of iterations, ⟨*value*⟩, have been performed.
options.max_iter evaluations of $F(x)$ have been completed, nag_opt_simplex has been terminated prematurely. Check the coding of the function **funct** before increasing the value of **options.max_iter**.

NE_USER_STOP
User requested termination, user flag value = ⟨*value*⟩

This exit occurs if the user sets **comm->flag** to a negative value in **funct**. If **fail** is supplied the value of **fail.errnum** will be the same as the user's setting of **comm->flag**.

9. Further Comments

The time taken depends on the number of variables, the behaviour of $F(x)$ and the distance of the starting point from the minimum. Each iteration consists of 1 or 2 evaluations of $F(x)$ unless the size of the simplex is reduced, in which case $n + 1$ evaluations are required.

10. References

Nelder J A and Mead R (1965) A Simplex Method for Function Minimization *Comput. J.* **7** 308–313.

Parkinson J M and Hutchinson D (1972) An Investigation Into the Efficiency of Variants of the Simplex Method *Numerical Methods for Nonlinear Optimization.* (ed. F A Lootsma) Academic Press.

11. See Also

nag_opt_init (e04xxc)
nag_opt_read (e04xyc)

12. Example 2

Example 2 solves the same problem as Example 1 but shows the use of certain optional parameters. This example shows option values being assigned directly within the program text and by reading values from a data file. The **options** structure is declared and initialised by nag_opt_init (e04xxc), a value is then assigned directly to the **print_fun** option. One further option, **optim_tol** is read from the data file by the use of nag_opt_read (e04xyc).

12.1. Program Text

```
static void ex2()
{

  double objf;
  double x[2];

  Integer i, n;
  Integer monit_freq;

  Boolean print;

  Nag_Comm comm;
  Nag_E04_Opt options;

  static NagError fail;

  Vprintf("\n\ne04ccc example 2: using option setting.\n");

  n = 2;
  monit_freq = 20;
  e04xxc(&options);
  options.print_fun = monit;
  /* Read remaining option value from file */
  fail.print = TRUE;
  print = TRUE;
  e04xyc("e04ccc", "stdin", &options, print, "stdout", NAGERR_DEFAULT);

  comm.p = (Pointer)&monit_freq;

  /* Starting values */
  x[0] = -1.0;
  x[1] = 1.0;

  e04ccc(n, funct, x, &objf, &options, &comm, &fail);

}

#ifdef NAG_PROTO
static void monit(const Nag_Search_State *st, Nag_Comm *comm)
#else
     static void monit(st, comm)
     Nag_Search_State *st;
     Nag_Comm *comm;
#endif
{
#define SIM(I,J) sim[((I)-1)*n + (J)-1]
```

```
  double *sim;
  Integer i, j;
  Integer n, ncall, iter;
  double fmin, fmax;

  Integer *monit_freq=(Integer *)comm->p;

  fmin = st->fmin;
  fmax = st->fmax;
  sim=st->simplex;
  ncall = st->nf;
  iter = st->iter;
  n = st->n;

  if (iter % *monit_freq == 0)
    {
      Vprintf("\nAfter %1ld iteration and %1ld function calls,\
 the function value is %10.4e\n", iter, ncall, fmin);
      Vprintf("The simplex is\n");
      for (i = 1; i <= n+1; ++i)
        {
          for (j = 1; j <= n; ++j)
            {
              Vprintf(" %12.4e", SIM(i,j));
            }
          Vprintf("\n");
        }
    }
  if (comm->sol_prt)
    {
      Vprintf("The final solution is\n");
      for (i = 0; i <n; i++)
        Vprintf("%12.4e\n", st->x[i]);
      Vprintf("After %1ld iterations and %1ld function calls the function \n\
value at the current solution point is %12.4e.\n", iter, ncall, fmin);
    }
} /* monit */
```

12.2. Program Data

```
e04ccc Example Program Data

Example data for ex2: using option setting

Following optional parameter settings are read by e04xyc

begin e04ccc

/* Error tolerance */

optim_tol = 1.0e-14
end
```

12.3. Program Results

```
e04ccc example 2: using option setting.

Optional parameter setting for e04ccc.
--------------------------------------

Option file: stdin

optim_tol set to 1.00e-14
```

```
Parameters to e04ccc
--------------------

Number of variables...........  2

optim_tol............... 1.00e-14    max_iter................     1500
print_level.........Nag_Soln_Iter    machine precision....... 1.11e-16
outfile.................   stdout

After 20 iteration and 44 function calls, the function value is 2.0075e-04
The simplex is
   5.0050e-01  -1.0083e+00
   5.1487e-01  -1.0182e+00
   5.1090e-01  -1.0008e+00

After 40 iteration and 83 function calls, the function value is 6.6865e-10
The simplex is
   5.0001e-01  -1.0000e+00
   5.0001e-01  -1.0000e+00
   4.9999e-01  -1.0000e+00
The final solution is
  5.0000e-01
 -1.0000e+00
After 58 iterations and 119 function calls the function
value at the current solution point is   2.9287e-15.
```
