

NAG C Library Function Document

nag_opt_nlp_option_set_file (e04wec)

1 Purpose

nag_opt_nlp_option_set_file (e04wec) may be used to supply optional arguments to nag_opt_nlp_solve (e04wdc) from an external file. The initialization function nag_opt_nlp_init (e04wcc) **must** have been called prior to calling nag_opt_nlp_option_set_file (e04wec).

2 Specification

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

```
void nag_opt_nlp_option_set_file (Nag_FileID fileid, Nag_E04State *state,
    NagError *fail)
```

3 Description

nag_opt_nlp_option_set_file (e04wec) may be used to supply values for optional arguments to nag_opt_nlp_solve (e04wdc). nag_opt_nlp_option_set_file (e04wec) reads an external file whose **fileid** has been returned by a call to nag_open_file (x04acc). nag_open_file (x04acc) must be called to provide **fileid**. Each line of the file defines a single optional argument. It is only necessary to supply values for those arguments whose values are to be different from their default values.

Each optional argument is defined by a single character string, consisting of one or more items. The items associated with a given option must be separated by spaces, or equals signs [=]. Alphabetic characters may be upper or lower case. The string

```
Print Level = 1
```

is an example of a string used to set an optional argument. For each option the string contains one or more of the following items:

- a mandatory keyword;
- a phrase that qualifies the keyword;
- a number that specifies an Integer or double value. Such numbers may be up to 16 contiguous characters which can be read using C's d or g formats, terminated by a space if this is not the last item on the line.

Blank strings and comments are ignored. A comment begins with an asterisk (*) and all subsequent characters in the string are regarded as part of the comment.

The file containing the options must start with **Begin** and must finish with **End**. An example of a valid options file is:

```
Begin * Example options file
    Print level = 5
End
```

Optional argument settings are preserved following a call to nag_opt_nlp_solve (e04wdc) and so the keyword **Defaults** is provided to allow you to reset all the optional arguments to their default values prior to a subsequent call to nag_opt_nlp_solve (e04wdc).

A complete list of optional arguments, their abbreviations, synonyms and default values is given in Section 11 of the document for nag_opt_nlp_solve (e04wdc).

4 References

Hock W and Schittkowski K (1981) *Test Examples for Nonlinear Programming Codes. Lecture Notes in Economics and Mathematical Systems* 187 Springer-Verlag

5 Arguments

- 1: **fileid** – Nag_FileID *Input*
Note: **fileid** is a NAG defined type (see Section 2.2.1.1 of the Essential Introduction).
On entry: the ID of the option file to be read as returned by a call to nag_open_file (x04acc).
- 2: **state** – Nag_E04State * *Communication Structure*
Note: **state** is a NAG defined type (see Section 2.2.1.1 of the Essential Introduction).
state contains internal information required for functions in this suite. It must not be modified in any way.
- 3: **fail** – NagError * *Input/Output*
The NAG error argument (see Section 2.6 of the Essential Introduction).

6 Error Indicators and Warnings

NE_BAD_PARAM

On entry, argument $\langle value \rangle$ had an illegal value.

NE_E04WCC_NOT_INIT

Initialization function nag_opt_nlp_init (e04wcc) has not been called.

NE_INTERNAL_ERROR

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please consult NAG for assistance.

7 Accuracy

Not applicable.

8 Further Comments

nag_opt_nlp_option_set_string (e04wfc), nag_opt_nlp_option_set_integer (e04wgc) or nag_opt_nlp_option_set_double (e04whc) may also be used to supply optional arguments to nag_opt_nlp_solve (e04wdc).

9 Example

This example is based on Problem 71 in Hock and Schittkowski (1981) and involves the minimization of the nonlinear function

$$F(x) = x_1 x_4 (x_1 + x_2 + x_3) + x_3$$

subject to the bounds

$$\begin{aligned} 1 &\leq x_1 \leq 5 \\ 1 &\leq x_2 \leq 5 \\ 1 &\leq x_3 \leq 5 \\ 1 &\leq x_4 \leq 5 \end{aligned}$$

to the general linear constraint

$$x_1 + x_2 + x_3 + x_4 \leq 20,$$

and to the nonlinear constraints

$$\begin{aligned} x_1^2 + x_2^2 + x_3^2 + x_4^2 &\leq 40, \\ x_1 x_2 x_3 x_4 &\geq 25. \end{aligned}$$

The initial point, which is infeasible, is

$$x_0 = (1, 5, 5, 1)^T,$$

and $F(x_0) = 16$.

The optimal solution (to five figures) is

$$x^* = (1.0, 4.7430, 3.8211, 1.3794)^T,$$

and $F(x^*) = 17.014$. One bound constraint and both nonlinear constraints are active at the solution.

The document for `nag_opt_nlp_option_set_file` (e04wec) includes an example program to solve the same problem using some of the optional arguments described in Section 11 of the document for `nag_opt_nlp_solve` (e04wdc).

9.1 Program Text

```

/* nag_opt_nlp_option_set_file (e04wec) Example Program.
 *
 * Copyright 2004 Numerical Algorithms Group.
 *
 * Mark 8, 2004.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nage04.h>
#include <nagx04.h>

static void confun(Integer *mode, Integer ncnln, Integer n,
                  Integer ldcj, const Integer needc[], const double x[],
                  double ccon[], double cjac[], Integer nstate,
                  Nag_Comm *comm);

static void objfun(Integer *mode, Integer n, const double x[],
                  double *objf, double grad[], Integer nstate, Nag_Comm *comm);

int main(void)
{
    /* Scalars */
    double bndinf, featol, objf;
    Integer elmode, exit_status, i, j, majits, n, nclin, ncnln, nctotal, pda;
    Integer pdcj, pdh;

    /* Arrays */
    double *a=0, *bl=0, *bu=0, *ccon=0, *cjac=0, *clamda=0, *grad=0, *hess=0;
    double *ruser=0, *x=0;
    Integer *istate=0, *iuser=0;

    /*Nag Types*/
    Nag_E04State state;
    Nag_Error fail;
    Nag_Comm comm;
    Nag_FileID fileid;

#define A(I,J) a[(I-1)*pda + J - 1]

    exit_status = 0;

```

```

INIT_FAIL(fail);
Vprintf("%s", "nag_opt_sparse_nlp_option_set_file (e04wec) Example Program"
        " Results");
Vprintf("\n");

/* This program demonstrates the use of routines to set and get values of
 * optional parameters associated with nag_opt_nlp_solve (e04wdc).
 */

/* Skip heading in data file */
Vscanf("%*[\n] ");

Vscanf("%ld %ld %ld ", &n, &nclin, &ncnln);
Vscanf("%*[\n] ");

if (n > 0 && nclin >= 0 && ncnln >= 0)
{
    /* Allocate memory */
    nctotal = n + nclin + ncnln;
    if ( !(a = NAG_ALLOC(ncnln*n, double)) ||
          !(bl = NAG_ALLOC(nctotal, double)) ||
          !(bu = NAG_ALLOC(nctotal, double)) ||
          !(ccon = NAG_ALLOC(ncnln, double)) ||
          !(cjac = NAG_ALLOC(ncnln*n, double)) ||
          !(clamda = NAG_ALLOC(nctotal, double)) ||
          !(grad = NAG_ALLOC(n, double)) ||
          !(hess = NAG_ALLOC(n*n, double)) ||
          !(ruser = NAG_ALLOC(1, double)) ||
          !(x = NAG_ALLOC(n, double)) ||
          !(istate = NAG_ALLOC(nctotal, Integer)) ||
          !(iuser = NAG_ALLOC(1, Integer)) )
    {
        Vprintf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }
    pda = n;
    pdcj = n;
    pdh = n;
}
/* Read A, BL, BU and X from data file */
if (nclin > 0)
{
    for (i = 1; i <= nclin; ++i)
    {
        for (j = 1; j <= n; ++j)
        {
            Vscanf("%lf", &A(i,j));
        }
        Vscanf("%*[\n] ");
    }

    for (i = 1; i <= n + nclin + ncnln; ++i)
    {
        Vscanf("%lf", &bl[i - 1]);
    }
    Vscanf("%*[\n] ");

    for (i = 1; i <= n + nclin + ncnln; ++i)
    {
        Vscanf("%lf", &bu[i - 1]);
    }
    Vscanf("%*[\n] ");

    for (i = 1; i <= n; ++i)
    {
        Vscanf("%lf", &x[i - 1]);
    }
}

```

```

Vscanf("%*[^\\n] ");

/* Call nag_opt_nlp_init (e04wcc) to initialise e04wdc. */
/* nag_opt_nlp_init (e04wcc).
 * Initialization function for nag_opt_nlp_solve (e04wdc)
 */
nag_opt_nlp_init(&state, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Initialisation of nag_opt_nlp_init (e04wcc) failed.\\n");
    exit_status = 1;
    goto END;
}
/* By default nag_opt_nlp_solve (e04wdc) does not print monitoring
 * information. Call nag_open_file (x04acc) to set the print file fileid.
 */
/* nag_open_file (x04acc).
 * Open unit number for reading, writing or appending, and
 * associate unit with named file
 */
nag_open_file("", 2, &fileid, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Fileid could not be obtained.\\n");
    exit_status = 1;
    goto END;
}

/* Use nag_opt_nlp_option_set_integer (e04wgc) to set the Integer-valued
 * option 'Print file' */
/* nag_opt_nlp_option_set_integer (e04wgc).
 * Set a single option for nag_opt_nlp_solve (e04wdc) from
 * an integer argument
 */
nag_opt_nlp_option_set_integer("Print file", fileid, &state, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("nag_opt_nlp_option_set_integer (e04wgc) failed to set Print"
           " File\\n");
    exit_status = 1;
    goto END;
}

/* Use nag_opt_sparse_nlp_option_set_file (e04wec) to read some options from
 * the end of the input data file. Call nag_open_file (x04acc) to set the
 * stdin fileid.
 */
/* nag_open_file (x04acc), see above. */
nag_open_file("", 0, &fileid, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Fileid could not be obtained.\\n");
    exit_status = 1;
    goto END;
}
/* nag_opt_nlp_option_set_file (e04wec).
 * Supply optional parameter values for nag_opt_nlp_solve
 * (e04wdc) from external file
 */
nag_opt_nlp_option_set_file(fileid, &state, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("nag_opt_sparse_nlp_option_set_file (e04wec) could not read input"
           " File\\n");
    exit_status = 1;
    goto END;
}

Vprintf("\\n");

/* Use nag_opt_nlp_option_get_integer (e04wkc) to find the value of

```

```

* Integer-valued option 'Elastic mode'.
*/
/* nag_opt_nlp_option_get_integer (e04wkc).
* Get the setting of an integer valued option of
* nag_opt_nlp_solve (e04wdc)
*/
nag_opt_nlp_option_get_integer("Elastic mode", &elmode, &state, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("nag_opt_nlp_option_get_integer (e04wkc) failed to find the value"
           " of Elastic Mode\n");
    exit_status = 1;
    goto END;
}
Vprintf("%1s", "");
Vprintf("Option 'Elastic mode' has the value ");

Vprintf("%3ld", elmode);

Vprintf(".");

Vprintf("\n");

/* Use nag_opt_nlp_option_set_double (e04whc) to set the value of real-valued
* option 'Infinite bound size'.
*/
bndinf = 1e10;
/* nag_opt_nlp_option_set_double (e04whc).
* Set a single option for nag_opt_nlp_solve (e04wdc) from a
* double argument
*/
nag_opt_nlp_option_set_double("Infinite bound size", bndinf, &state, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("nag_opt_nlp_option_set_double (e04whc) failed to set Infinite"
           " bound size\n");
    exit_status = 1;
    goto END;
}

/* Use nag_opt_nlp_option_get_double (e04wlc) to find the value of real-valued
* option 'Feasibility tolerance'.
*/
/* nag_opt_nlp_option_get_double (e04wlc).
* Get the setting of a double valued option of
* nag_opt_nlp_solve (e04wdc)
*/
nag_opt_nlp_option_get_double("Feasibility tolerance", &featol, &state,
                             &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("nag_opt_nlp_option_get_double (e04wlc) failed to find the value"
           " of a real-valued option\n");
    exit_status = 1;
    goto END;
}
Vprintf("Option 'Feasibility tolerance' has the value %13.5e.\n", featol);

/* Use nag_opt_nlp_option_set_string (e04wfc) to set the option 'Major
* iterations limit'.
*/
/* nag_opt_nlp_option_set_string (e04wfc).
* Set a single option for nag_opt_nlp_solve (e04wdc) from a
* character string
*/
nag_opt_nlp_option_set_string("Major iterations limit 50", &state, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("nag_opt_nlp_option_set_string (e04wfc) failed to set Major"
           " iterations limit\n");
    exit_status = 1;
}

```

```

        goto END;
    }

    /* Solve the problem. */
    /* nag_opt_nlp_solve (e04wdc).
    * Solves the nonlinear programming (NP) problem
    */
    nag_opt_nlp_solve(n, nclin, ncnln, n, n, n, a, bl, bu,
                     confun, objfun, &majits, istate, ccon, cjac, clamda, &
                     objf, grad, hess, x, &state, &comm, &fail);

    if (fail.code == NE_NOERROR)
    {
        Vprintf("\nFinal objective value = %11.3f\n", objf);

        Vprintf("Optimal X = ");
        for (i = 1; i <= n; ++i)
        {
            Vprintf("%9.2f%s", x[i - 1], i%7 == 0 || i == n ? "\n":" ");
        }
    }
    else
    {
        Vprintf ("Error message from nag_opt_nlp_solve (e04wdc) %s\n",
                fail.message);
    }
    END:
    if (a) NAG_FREE(a);
    if (bl) NAG_FREE(bl);
    if (bu) NAG_FREE(bu);
    if (ccon) NAG_FREE(ccon);
    if (cjac) NAG_FREE(cjac);
    if (clamda) NAG_FREE(clamda);
    if (grad) NAG_FREE(grad);
    if (hess) NAG_FREE(hess);
    if (ruser) NAG_FREE(ruser);
    if (x) NAG_FREE(x);
    if (istate) NAG_FREE(istate);
    if (iuser) NAG_FREE(iuser);

    return exit_status;
}

static void objfun(Integer *mode, Integer n, const double x[],
                  double *objf, double grad[], Integer nstate, Nag_Comm *comm)
{
    /* Routine to evaluate objective function and its 1st derivatives. */

    /* Function Body */
    if (*mode == 0 || *mode == 2)
    {
        *objf = x[0] * x[3] * (x[0] + x[1] + x[2]) + x[2];
    }

    if (*mode == 1 || *mode == 2)
    {
        grad[0] = x[3] * (x[0] * 2. + x[1] + x[2]);
        grad[1] = x[0] * x[3];
        grad[2] = x[0] * x[3] + 1.;
        grad[3] = x[0] * (x[0] + x[1] + x[2]);
    }

    return;
} /* objfun */

static void confun(Integer *mode, Integer ncnln, Integer n,
                  Integer ldcj, const Integer needc[], const double x[],
                  double ccon[], double cjac[], Integer nstate, Nag_Comm *comm)
{
    /* Scalars */
    Integer i, j;

```

```

#define CJAC(I,J) cjac[(I-1)*ldcj + J-1]

/* Routine to evaluate the nonlinear constraints and their 1st */
/* derivatives. */

/* Function Body */
if (nstate == 1)
{
  /* First call to CONFUN. Set all Jacobian elements to zero. */
  /* Note that this will only work when 'Derivative Level = 3' */
  /* (the default; see Section 11.2). */
  for (j = 1; j <= n; ++j)
  {
    for (i = 1; i <= ncnln; ++i)
    {
      CJAC(i, j) = 0.;
    }
  }
}

if (needc[0] > 0)
{
  if (*mode == 0 || *mode == 2)
  {
    ccon[0] = x[0] * x[0] + x[1] * x[1] + x[2] * x[2] + x[3] * x[3];
  }
  if (*mode == 1 || *mode == 2)
  {
    CJAC(1, 1) = x[0] * 2.;
    CJAC(1, 2) = x[1] * 2.;
    CJAC(1, 3) = x[2] * 2.;
    CJAC(1, 4) = x[3] * 2.;
  }
}

if (needc[1] > 0)
{
  if (*mode == 0 || *mode == 2)
  {
    ccon[1] = x[0] * x[1] * x[2] * x[3];
  }
  if (*mode == 1 || *mode == 2)
  {
    CJAC(2, 1) = x[1] * x[2] * x[3];
    CJAC(2, 2) = x[0] * x[2] * x[3];
    CJAC(2, 3) = x[0] * x[1] * x[3];
    CJAC(2, 4) = x[0] * x[1] * x[2];
  }
}

return;
} /* confun */

```


9.2 Program Data

```
nag_opt_sparse_nlp_option_set_file (e04wec) Example Program Data
  4   1   2                                     : N, NCLIN and NCNLN
  1.0  1.0  1.0  1.0                           : Matrix A
  1.0  1.0  1.0  1.0 -1.0E+25 -1.0E+25  25.0    : Lower bounds BL
  5.0  5.0  5.0  5.0  20.0   40.0   1.0E+25    : Upper bounds BU
  1.0  5.0  5.0  1.0                           : Initial vector X
Begin example options file
* Comment lines like this begin with an asterisk.
* Switch off output of timing information:
Timing level 0
* Allow elastic variables:
Elastic mode 1
* Set the feasibility tolerance:
Feasibility tolerance 1.0E-4
End
```

9.3 Program Results

nag_opt_sparse_nlp_option_set_file (e04wec) Example Program Results

```
OPTIONS file
-----

Begin example options file
* Comment lines like this begin with an asterisk.
* Switch off output of timing information:
Timing level 0
* Allow elastic variables:
Elastic mode 1
* Set the feasibility tolerance:
Feasibility tolerance 1.0E-4
End

E04WEZ EXIT 100 -- finished successfully
E04WEZ INFO 101 -- OPTIONS file read

Option 'Elastic mode' has the value 1.
Option 'Feasibility tolerance' has the value 1.00000e-04.

Parameters
=====

Files
-----
Solution file..... 0      Old basis file ..... 0      (Print file)..... 6
Insert file..... 0      New basis file ..... 0      (Summary file)..... 0
Punch file..... 0      Backup basis file..... 0
Load file..... 0      Dump file..... 0

Frequencies
-----
Print frequency..... 100    Check frequency..... 60    Save new basis map..... 100
Summary frequency..... 100  Factorization frequency 50    Expand frequency..... 10000

QP subproblems
-----
QPsolver Cholesky.....
Scale tolerance..... 0.900  Minor feasibility tol.. 1.00E-04  Iteration limit..... 10000
Scale option..... 0      Minor optimality tol.. 1.00E-06  Minor print level..... 1
Crash tolerance..... 0.100  Pivot tolerance..... 2.05E-11  Partial price..... 1
Crash option..... 3      Elastic weight..... 1.00E+04  Prtl price section ( A) 4
New superbasics..... 99    Prtl price section (-I) 3

The SQP Method
-----
Minimize.....          Cold start.....          Proximal Point method.. 1
Nonlinear objectiv vars 4      Major optimality tol... 2.00E-06  Function precision..... 1.72E-13
```

Unbounded step size....	1.00E+10	Superbasics limit.....	4	Difference interval....	4.15E-07
Unbounded objective....	1.00E+15	Reduced Hessian dim....	4	Central difference int.	5.57E-05
Major step limit.....	2.00E+00	Derivative linesearch..		Derivative level.....	3
Major iterations limit.	50	Linesearch tolerance...	0.90000	Verify level.....	0
Minor iterations limit.	500	Penalty parameter.....	0.00E+00	Major Print Level.....	1

Hessian Approximation

Full-Memory Hessian....		Hessian updates.....	99999999	Hessian frequency.....	99999999
				Hessian flush.....	99999999

Nonlinear constraints

Nonlinear constraints..	2	Major feasibility tol..	1.00E-06	Violation limit.....	1.00E+06
Nonlinear Jacobian vars	4				

Miscellaneous

LU factor tolerance....	1.10	LU singularity tol.....	2.05E-11	Timing level.....	0
LU update tolerance....	1.10	LU swap tolerance.....	1.03E-04	Debug level.....	0
LU partial pivoting...		eps (machine precision)	1.11E-16	System information....	No

Nonlinear constraints	2	Linear constraints	1
Nonlinear variables	4	Linear variables	0
Jacobian variables	4	Objective variables	4
Total constraints	3	Total variables	4

The user has defined 8 out of 8 constraint gradients.
 The user has defined 4 out of 4 objective gradients.

Cheap test of user-supplied problem derivatives...

The constraint gradients seem to be OK.

--> The largest discrepancy was 1.84E-07 in constraint 6

The objective gradients seem to be OK.

Gradient projected in one direction 4.99993000077E+00
 Difference approximation 4.99993303727E+00

Itns	Major	Minors	Step	nCon	Feasible	Optimal	MeritFunction	L+U	BSwap	nS	condHz	Penalty
2	0	2		1	1.7E+00	2.8E+00	1.6000000E+01	7		2	1.0E+00	_ r
4	1	2	1.0E+00	2	1.3E-01	3.2E-01	1.7726188E+01	8		1	6.2E+00	8.3E-02 _n r
5	2	1	1.0E+00	3	3.7E-02	1.7E-01	1.7099571E+01	7		1	2.0E+00	8.3E-02 _s
6	3	1	1.0E+00	4	2.2E-02	1.1E-02	1.7014005E+01	7		1	1.8E+00	8.3E-02 _
7	4	1	1.0E+00	5	1.5E-04	6.0E-04	1.7014018E+01	7		1	1.8E+00	9.2E-02 _
8	5	1	1.0E+00	6	(3.3E-07)	2.3E-05	1.7014017E+01	7		1	1.9E+00	3.6E-01 _
9	6	1	1.0E+00	7	(4.2E-10)	(2.4E-08)	1.7014017E+01	7		1	1.9E+00	3.6E-01 _

E04WDF EXIT 0 -- finished successfully
 E04WDF INFO 1 -- optimality conditions satisfied

Problem name	NLP		
No. of iterations	9	Objective value	1.7014017287E+01
No. of major iterations	6	Linear objective	0.0000000000E+00
Penalty parameter	3.599E-01	Nonlinear objective	1.7014017287E+01
No. of calls to funobj	8	No. of calls to funcon	8
No. of superbasics	1	No. of basic nonlinears	2
No. of degenerate steps	0	Percentage	0.00
Max x	2 4.7E+00	Max pi	2 5.5E-01
Max Primal infeas	0 0.0E+00	Max Dual infeas	3 4.8E-08
Nonlinear constraint violn	2.7E-09		

Variable	State	Value	Lower bound	Upper bound	Lagr multiplier	Slack
variable	1	LL	1.000000	5.000000	1.087871	.
variable	2	FR	4.743000	5.000000	.	0.2570
variable	3	FR	3.821150	5.000000	.	1.179
variable	4	FR	1.379408	5.000000	.	0.3794

Linear constrnt	State	Value	Lower bound	Upper bound	Lagr multiplier	Slack
lincon	1	FR	10.94356	20.00000	.	9.056

Nonlin constrnt	State	Value	Lower bound	Upper bound	Lagr multiplier	Slack
nlncn	1	UL	40.00000	40.00000	-0.1614686	-0.2700E-08
nlncn	2	LL	25.00000	None	0.5522937	-0.2215E-08

Final objective value = 17.014

Optimal X = 1.00 4.74 3.82 1.38
