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 before calling nag_opt_nlp_option_set_file (e04wec).

2 Specification

```c
#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 before 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 12 in nag_opt_nlp_solve (e04wdc).
4 References


5 Arguments

1: fileid – Nag_FileID  
   Input
   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
   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 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_ALLOC_FAIL
Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

NE_BAD_PARAM
On entry, argument (value) had an illegal value.

NE_E04_OPTION_INVALID
At least one line of the options file is invalid.
Could not read options file on unit fileid = (value).
Could not read options file on unit fileid. This may be due to:
(a) fileid is not a valid unit number;
(b) a file is not associated with unit fileid, or if it is, is unavailable for read access;
(c) one or more lines of the options file is invalid. Check that all keywords are neither ambiguous nor misspelt;
(d) Begin was found, but end-of-file was found before End was found;
(e) end-of-file was found before Begin was found.

NE_E04WCC_NOT_INIT
The 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 contact NAG for assistance.
An unexpected error has been triggered by this function. Please contact NAG.
See Section 3.6.6 in the Essential Introduction for further information.

NE_NO_LICENCE
Your licence key may have expired or may not have been installed correctly.
See Section 3.6.5 in the Essential Introduction for further information.
7 Accuracy
Not applicable.

8 Parallelism and Performance
Not applicable.

9 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).

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

\[ F(x) = x_1x_4(x_1 + x_2 + x_3) + x_3 \]

subject to the bounds

\[ 1 \leq x_1 \leq 5 \]
\[ 1 \leq x_2 \leq 5 \]
\[ 1 \leq x_3 \leq 5 \]
\[ 1 \leq x_4 \leq 5 \]

to the general linear constraint

\[ x_1 + x_2 + x_3 + x_4 \leq 20, \]

and to the nonlinear constraints

\[ x_1^4 + x_2^4 + x_3^4 + x_4^4 \leq 40, \quad x_1x_2x_3x_4 \geq 25. \]

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 12 in nag_opt_nlp_solve (e04wdc).

10.1 Program Text

/* nag_opt_nlp_option_set_file (e04wec) Example Program. *
 * Copyright 2014 Numerical Algorithms Group.
 */
#include <stdio.h>
#include <string.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nage04.h>

Mark 25
```c
#ifdef __cplusplus
extern "C" {
#endif
static void NAG_CALL 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 NAG_CALL objfun(Integer *mode, Integer n, const double x[],
    double *objf, double grad[], Integer nstate,
    Nag_Comm *comm);
#ifdef __cplusplus}
#endif
int main(void)
{
    const char *optionsfile = "e04wece.opt";

    /* Scalars */
    double bndinf, featol, objf;
    Integer elmode, exit_status, i, j, majits, n, nclin, ncnln, nctotal,
        pda;
    Integer pdcj, pdh;

    /* Arrays */
    static double ruser[2] = {-1.0, -1.0};
    double *a = 0, *bl = 0, *bu = 0, *ccon = 0, *cjac = 0, *clamda = 0;
    double *grad = 0, *hess = 0, *x = 0;
    Integer *istate = 0, *iuser = 0;

    /* Nag Types */
    Nag_E04State state;
    NagError fail;
    Nag_Comm comm;
    Nag_FileID fileidin,
    Nag_FileID fileidout;

    #define A(I, J) a[(I-1)*pda + J - 1]
    exit_status = 0;
    INIT_FAIL(fail);

    printf("%s", "nag_opt_nlp_option_set_file (e04wec) Example Program"
            " Results");
    printf("\n");

    /* For communication with user-supplied functions: */
    comm.user = ruser;
    fflush(stdout);

    /* This program demonstrates the use of routines to set and get values of
    * optional parameters associated with nag_opt_nlp_solve (e04wec).
    */
    /* Skip heading in data file */
    #ifdef __WIN32
    scanf_s("%*[\n] ");
    #else
    scanf("%*[\n] ");
    #endif

    #ifdef __WIN32
    scanf_s("%"NAG_IFMT" %"NAG_IFMT" %"NAG_IFMT" ", &n, &nclin, &ncnln);
    #else
    scanf("%"NAG_IFMT" %"NAG_IFMT" %"NAG_IFMT" ", &n, &nclin, &ncnln);
    #endif
    #ifdef __WIN32
    ...
    #endif
```
```c
scanf_s("%*[\n ]");
#else
    scanf("%*[\n ]");
#endif

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)) ||
        !(x = NAG_ALLOC(n, double)) ||
        !(istate = NAG_ALLOC(nctotal, Integer)) ||
        !(iuser = NAG_ALLOC(1, Integer)))
    {
        printf("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)
        {
            #ifdef _WIN32
                scanf_s("%lf", &A(i, j));
            #else
                scanf("%lf", &A(i, j));
            #endif
        }
    }
    /* Read A, BL, BU and X from data file */
    for (i = 1; i <= n + nclin + ncnln; ++i)
    {
        #ifdef _WIN32
            scanf_s("%lf", &bl[i - 1]);
        #else
            scanf("%lf", &bl[i - 1]);
        #endif
        #ifdef _WIN32
            scanf_s("%*[\n ]");
        #else
            scanf("%*[\n ]");
        #endif
        for (i = 1; i <= n + nclin + ncnln; ++i)
        {
            #ifdef _WIN32
                scanf_s("%lf", &bu[i - 1]);
            #else
                scanf("%lf", &bu[i - 1]);
            #endif
        }
}
END:
```

---

Mark 25 e04wec
```c
#define _WIN32
scanf_s("%*\[\n\] ");
#else
scanf("%*\[\n\] ");
#endif

for (i = 1; i <= n; ++i)
{
#define _WIN32
scanf_s("%lf", &x[i - 1]);
#else
scanf("%lf", &x[i - 1]);
#endif

#define _WIN32
scanf("%*\[\n\] ");
#else
scanf("%*\[\n\] ");
#endif

/* Call nag_opt_nlp_init (e04wcc) to initialise nag_opt_nlp_solve (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)
{
    printf("Initialisation of nag_opt_nlp_init (e04wcc) failed.\n%s\n", fail.message);
    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, &fileidout, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Fileidout 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", fileidout, &state, &fail);
if (fail.code != NE_NOERROR)
{
    printf("nag_opt_nlp_option_set_integer (e04wgc) failed to set Print" 
" File\n");
    exit_status = 1;
    goto END;
}

/* Use nag_opt_nlp_option_set_file (e04wec) to read some options from */
/* the options file. First call nag_open_file (x04acc) to set the options file */
/* fileid. */
nag_open_file(optionsfile, 0, &fileidin, &fail);
if (fail.code != NE_NOERROR)
{
    printf("nag_open_file (optionsfile, 0, &fileidin, &fail) failed.\n");
    exit_status = 1;
    goto END;
}
```

The above code snippet is a part of the NAG Library Manual, specifically the `e04wec` module.
printf("Fileidin 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(fileidin, &state, &fail);
if (fail.code != NE_NOERROR)
{
    printf("nag_opt_nlp_option_set_file (e04wec) could not read input\n"
    " File\n");
    exit_status = 1;
    goto END;
}

/* Use nag_opt_nlp_option_get_integer (e04wkc) to find the value of
 * Integer-valued option 'Elastic mode'.
 */

nag_opt_nlp_option_get_integer("Elastic mode", &elmode, &state, &fail);
if (fail.code != NE_NOERROR)
{
    printf("nag_opt_nlp_option_get_integer (e04wkc) failed to find the value\n"
    " of Elastic Mode\n");
    exit_status = 1;
    goto END;
}

printf("Option 'Elastic mode' has the value ");
printf("%3"NAG_IFMT".\n", elmode);

/* 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("Infinite bound size", bndinf, &state, &fail);
if (fail.code != NE_NOERROR)
{
    printf("nag_opt_nlp_option_set_double (e04whc) failed to set Infinite\n" " 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("Feasibility tolerance", &featol, &state, &fail);
if (fail.code != NE_NOERROR)
{
    printf("nag_opt_nlp_option_get_double (e04wlc) failed to find the value\n" " of a real-valued option\n");
    exit_status = 1;
    goto END;
}

printf("Option 'Feasibility tolerance' has the value %14.5e.\n", featol);
/* Use nag_opt_nlp_option_set_string (e04wfc) to set the option 'Major
* iterations limit'.
*/

/* 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)
{
    printf("nag_opt_nlp_option_set_string (e04wfc) failed to set Major"
            " iterations limit\n");
    exit_status = 1;
    goto END;
}

fflush(stdout);

/* Solve the problem. */

nag_opt_nlp_solve(n, nclin, ncnln, pda, pdcj, pdh, a, bl, bu,
                   confun, objfun, &majits, istate, ccon, cjac, clamda, &
                   objf, grad, hess, x, &state, &comm, &fail);

if (fail.code == NE_NOERROR)
{
    printf("Final objective value = %11.3f\n", objf);
    printf("Optimal X = ");
    for (i = 1; i <= n; ++i)
    {
        printf("%9.2f%s", x[i - 1], i%7 == 0 || i == n?"\n":" ");
    }
}
else
{
    printf("Error message from nag_opt_nlp_solve (e04wdc).\n");
    fail.message);
}

END:
NAG_FREE(a);
NAG_FREE(bl);
NAG_FREE(bu);
NAG_FREE(ccon);
NAG_FREE(cjac);
NAG_FREE(clamda);
NAG_FREE(grad);
NAG_FREE(hess);
NAG_FREE(x);
NAG_FREE(istate);
NAG_FREE(iuser);
return exit_status;
}

static void NAG_CALL 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 (comm->user[0] == -1.0)
{

flush(stdout);
printf("(User-supplied callback objfun, first invocation.)\n");
comm->user[0] = 0.0;
flush(stdout);
}

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

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

return;
} /* objfun */

static void NAG_CALL 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 (comm->user[1] == -1.0)
    {
        flush(stdout);
        printf("(User-supplied callback confun, first invocation.)\n");
        comm->user[1] = 0.0;
        flush(stdout);
    }

    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)
        {
        }
        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)
    {
    }
    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 */

10.2 Program Data

Begin nag_opt_nlp_option_set_file (e04wec) 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

nag_opt_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

10.3 Program Results

nag_opt_nlp_option_set_file (e04wec) Example Program Results

OPTIONS file
-----------

Begin nag_opt_nlp_option_set_file (e04wec) 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
-------------
QP solver 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......... 2.04E-11  Partial price......... 1
Crash option........... 3  Elastic weight.......... 1.00E+04  Prt1 price section ( A) 4
                     New superbasics......... 99  Prt1 price section (-I) 3

The SQP Method
--------------
Minimize............... 4  Major optimality tol... 2.00E-06  Function precision..... 1.72E-13
Nonlinear objectiv vars 4  Major feasibility tol... 1.00E-06  Violation limit........ 1.00E+06
Unbounded step size.... 1.00E+10  Superbasics limit...... 4  Central difference int. 5.57E-05
Nonlinear Jacobian vars 4  Reduced Hessian dim..... 4  Derivative level....... 3
Major step limit....... 2.00E+00  Derivative linesearch  Derivative level....... 4
Major iterations limit. 50  Linesearch tolerance. 0.900000  Verify level........... 0
Minor iterations limit. 500  Penalty parameter...... 0.00E+00  Major Print Level...... 1

Hessian Approximation
---------------------
Full-Memory Hessian... 99999999  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..... 2.04E-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

Matrix statistics
-----------------
<table>
<thead>
<tr>
<th></th>
<th>Total</th>
<th>Normal</th>
<th>Free</th>
<th>Fixed</th>
<th>Bounded</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rows</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Columns</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>No. of matrix elements</td>
<td>12</td>
<td>Density 100.000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Biggest</td>
<td>1.0000E+00</td>
<td>(excluding fixed columns,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Smallest</td>
<td>0.0000E+00</td>
<td>free rows, and RHS)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No. of objective coefficients</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nonlinear constraints</td>
<td>2</td>
<td>Linear constraints</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nonlinear variables</td>
<td>4</td>
<td>Linear variables</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jacobian variables</td>
<td>4</td>
<td>Objective variables</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total constraints</td>
<td>3</td>
<td>Total variables</td>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(User-supplied callback confun, first invocation.)
(User-supplied callback objfun, first invocation.)
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 $4.99993303560E+00$

<table>
<thead>
<tr>
<th>Itns</th>
<th>Major Minors</th>
<th>Step</th>
<th>nCon</th>
<th>Feasible</th>
<th>Optimal</th>
<th>MeritFunction</th>
<th>L+U BS</th>
<th>Penalty</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>$1.7E+00$</td>
<td>$2.8E+00$</td>
<td>$1.60000000E+01$</td>
<td>7</td>
<td>2 $1.0E+00$</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>$1.3E-01$</td>
<td>$3.2E-01$</td>
<td>$1.7726188E+01$</td>
<td>8</td>
<td>1 $6.2E+00$ $8.3E-02$</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>$1.0E+00$</td>
<td>$3.7E-02$</td>
<td>$1.7099571E+01$</td>
<td>7</td>
<td>1 $2.0E+00$ $8.3E-02$</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>$1.0E+00$</td>
<td>$2.2E-02$</td>
<td>$1.714005E+01$</td>
<td>7</td>
<td>1 $1.8E+00$ $8.3E-02$</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>$1.0E+00$</td>
<td>$1.5E-04$</td>
<td>$6.0E-04$</td>
<td>$1.7014018E+01$</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>$1.0E+00$</td>
<td>$4 (3.3E-07)$</td>
<td>$2.3E-07$</td>
<td>$1.7014017E+01$</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>$1.0E+00$</td>
<td>$7 (4.2E-10)(2.4E-08)$</td>
<td>$1.7014017E+01$</td>
<td>7</td>
<td>1 $1.9E+00$ $3.6E-01$</td>
</tr>
</tbody>
</table>

E04WDM EXIT 0 -- finished successfully
E04WDM INFO 1 -- optimality conditions satisfied

<table>
<thead>
<tr>
<th>Variable</th>
<th>State</th>
<th>Value</th>
<th>Lower bound</th>
<th>Upper bound</th>
<th>Lagr multiplier</th>
<th>Slack</th>
</tr>
</thead>
<tbody>
<tr>
<td>variable</td>
<td>LL</td>
<td>1.000000</td>
<td>1.000000</td>
<td>5.000000</td>
<td>1.087871</td>
<td>.</td>
</tr>
<tr>
<td>variable</td>
<td>FR</td>
<td>4.743000</td>
<td>1.000000</td>
<td>5.000000</td>
<td>.</td>
<td>0.2570</td>
</tr>
<tr>
<td>variable</td>
<td>FR</td>
<td>3.821150</td>
<td>1.000000</td>
<td>5.000000</td>
<td>.</td>
<td>1.179</td>
</tr>
<tr>
<td>variable</td>
<td>FR</td>
<td>1.379408</td>
<td>1.000000</td>
<td>5.000000</td>
<td>.</td>
<td>0.3794</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Linear constr</th>
<th>State</th>
<th>Value</th>
<th>Lower bound</th>
<th>Upper bound</th>
<th>Lagr multiplier</th>
<th>Slack</th>
</tr>
</thead>
<tbody>
<tr>
<td>lincon</td>
<td>FR</td>
<td>10.94356</td>
<td>None</td>
<td>20.000000</td>
<td>.</td>
<td>9.056</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Nonlinear constr</th>
<th>State</th>
<th>Value</th>
<th>Lower bound</th>
<th>Upper bound</th>
<th>Lagr multiplier</th>
<th>Slack</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlncon</td>
<td>LL</td>
<td>40.000000</td>
<td>None</td>
<td>40.000000</td>
<td>$-0.1614686$</td>
<td>$-0.2700E-08$</td>
</tr>
<tr>
<td>nlncon</td>
<td>LL</td>
<td>25.000000</td>
<td>25.000000</td>
<td>None</td>
<td>0.5552937</td>
<td>$-0.2215E-08$</td>
</tr>
</tbody>
</table>

Final objective value = $17.014$
Optimal X = 1.00 4.74 3.82 1.38