NAG Library Function Document

nag_opt_sparse_nlp_option_set_file (e04vkc)

1 Purpose

nag_opt_sparse_nlp_option_set_file (e04vkc) may be used to supply optional arguments to nag_opt_sparse_nlp_solve (e04vhc) from an external file. The initialization function nag_opt_sparse_nlp_init (e04vgc) **must** have been called before calling nag_opt_sparse_nlp_option_set_file (e04vkc).

2 Specification

```
#include <nag.h>
#include <nage04.h>
void nag_opt_sparse_nlp_option_set_file (Nag_FileID fileid,
     Nag_E04State *state, NagError *fail)
```

3 Description

nag_opt_sparse_nlp_option_set_file (e04vkc) may be used to supply values for optional arguments to nag_opt_sparse_nlp_solve (e04vhc). nag_opt_sparse_nlp_option_set_file (e04vkc) 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_sparse_nlp_solve (e04vhc) 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_sparse_nlp_solve (e04vhc).

A complete list of optional arguments, their abbreviations, synonyms and default values is given in Section 12 in nag_opt_sparse_nlp_solve (e04vhc).

4 References

None.

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_BAD_PARAM

On entry, argument ⟨*value*⟩ had an illegal value.

NE_E04VGC_NOT_INIT

Initialization function nag_opt_sparse_nlp_init (e04vgc) 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.

NE_OPTIONS_FILE_READ_FAILURE

At least one line of the options file is invalid.

Could not read options file on unit **fileid** = ⟨*value*⟩.

7 Accuracy

Not applicable.

8 Parallelism and Performance

Not applicable.

9 Further Comments

nag_opt_sparse_nlp_option_set_string (e04vlc), nag_opt_sparse_nlp_option_set_integer (e04vmc) or nag_opt_sparse_nlp_option_set_double (e04vnc) may also be used to supply optional arguments to nag_opt_sparse_nlp_solve (e04vhc).

10 Example

This example solves the same problem as the example in the document for nag_opt_sparse_nlp_solve (e04vhc), but sets and reads some optional arguments first. See Section 10 in nag_opt_sparse_nlp_solve (e04vhc) for further details.

The example in the document for nag_opt_sparse_nlp_jacobian (e04vjc) also solves the same problem (see Section 10 in nag_opt_sparse_nlp_jacobian (e04vjc)), but it first calls nag_opt_sparse_nlp_jacobian (e04vjc) to determine the sparsity pattern before calling nag_opt_sparse_nlp_option_set_file (e04vkc).

10.1 Program Text

```
/* nag_opt_sparse_nlp_option_set_file (e04vkc) Example Program.
 *
 * Copyright 2004 Numerical Algorithms Group.
 *
 * Mark 8, 2004.
 */

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

#ifdef __cplusplus
extern "C" {
#endif
static void NAG_CALL usrfun(Integer *status, Integer n, const double x[],
                            Integer needf, Integer nf, double f[],
                            Integer needg, Integer leng, double g[],
                            Nag_Comm *comm);
#ifdef __cplusplus
}
#endif

int main(void)
{
  const char   *optionsfile = "e04vkce.opt";

  /* Scalars */
  double       bndinf, featol, objadd, sinf;
  Integer      elmode, exit_status = 0, i, lena, leng, n, nea, neg, nf, nfname,
               ninf;
  Integer      ns, nxname, objrow;

  /* Arrays */
  static double ruser[1] = {-1.0};
  char         nag_enum_arg[40];
  char         **fnames = 0, *prob = 0, **xnames = 0;
  double       *a = 0, *f = 0, *flow = 0, *fmul = 0, *fupp = 0;
  double       *x = 0, *xlow = 0, *xmul = 0, *xupp = 0;
  Integer      *fstate = 0, *iafun = 0, *igfun = 0, *iuser = 0, *javar = 0;
  Integer      *jgvar = 0, *xstate = 0;

  /* Nag Types */
  Nag_E04State state;
  NagError     fail;
  Nag_Comm     comm;
  Nag_Start    start;
  Nag_FileID   optfileid, outfileid;

  INIT_FAIL(fail);

  printf("%s\n",
         "nag_opt_sparse_nlp_option_set_file (e04vkc) Example Program"
         " Results");

  /* 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_sparse_nlp_solve (e04vhc).
   */

  /* Skip heading in data file */
  scanf("%*[^\n] ");
  scanf("%ld%ld%*[^\n] ", &n, &nf);
  scanf("%ld%ld%ld %39s %*[^\n] ", &nea, &neg,
        &objrow, nag_enum_arg);

  /* nag_enum_name_to_value (x04nac).
   * Converts NAG enum member name to value
   */
  start = (Nag_Start) nag_enum_name_to_value(nag_enum_arg);

  if (n > 0 && nf > 0 && nea > 0 && neg > 0)
    {
      nxname = n;
      nfname = nf;

      /* Allocate memory */
      if (!(fnames = NAG_ALLOC(nfname, char *)) ||
          !(prob = NAG_ALLOC(9, char)) ||
          !(xnames = NAG_ALLOC(nxname, char *)) ||
          !(a = NAG_ALLOC(300, double)) ||
          !(f = NAG_ALLOC(100, double)) ||
          !(flow = NAG_ALLOC(100, double)) ||
          !(fmul = NAG_ALLOC(100, double)) ||
          !(fupp = NAG_ALLOC(100, double)) ||
          !(x = NAG_ALLOC(100, double)) ||
          !(xlow = NAG_ALLOC(100, double)) ||
          !(xmul = NAG_ALLOC(100, double)) ||
          !(xupp = NAG_ALLOC(100, double)) ||
          !(fstate = NAG_ALLOC(100, Integer)) ||
          !(iafun = NAG_ALLOC(300, Integer)) ||
          !(igfun = NAG_ALLOC(300, Integer)) ||
          !(iuser = NAG_ALLOC(1, Integer)) ||
          !(javar = NAG_ALLOC(300, Integer)) ||
          !(jgvar = NAG_ALLOC(300, Integer)) ||
          !(xstate = NAG_ALLOC(100, Integer)))
        {
          printf("Allocation failure\n");
          exit_status = -1;
          goto END;
        }
    }
  else
    {
      printf("Invalid n or nf or nea or neg\n");
      exit_status = 1;
      return exit_status;
    }
  lena = MAX(1, nea);
  leng = MAX(1, neg);
  objadd = 0.;
  strcpy(prob, "        ");

  /* Read the variable names xnames */

  for (i = 0; i < nxname; ++i) {
    xnames[i] = NAG_ALLOC(9, char);
    scanf(" ' %8s '", xnames[i]);
  }
  scanf("%*[^\n] ");

  /* Read the function names fnames */
  for (i = 0; i < nfname; ++i) {
    fnames[i] = NAG_ALLOC(9, char);
    scanf(" '%8s'", fnames[i]);
  }
  scanf("%*[^\n] ");
```

```
  /* Read the sparse matrix A, the linear part of F */
  for (i = 0; i < nea; ++i) {
    /* For each element read row, column, A(row,column) */
    scanf("%ld%ld%lf%*[^\n] ", &iafun[i], &javar[i], &a[i]);
  }
  /* Read the structure of sparse matrix g, the nonlinear part of f */
  for (i = 0; i < neg; ++i) {
    /* For each element read row, column */
    scanf("%ld%ld%*[^\n] ", &igfun[i], &jgvar[i]);
  }

  /* Read the lower and upper bounds on the variables */
  for (i = 0; i < n; ++i) {
    scanf("%lf%lf%*[^\n] ", &xlow[i], &xupp[i]);
  }

  /* Read the lower and upper bounds on the functions */
  for (i = 0; i < nf; ++i) {
    scanf("%lf%lf%*[^\n] ", &flow[i], &fupp[i]);
  }

  /* Initialise x, xstate, xmul, f, fstate, fmul */
  for (i = 0; i < n; ++i) {
    scanf("%lf", &x[i]);
  }
  scanf("%*[^\n] ");

  for (i = 0; i < n; ++i) {
    scanf("%ld", &xstate[i]);
  }
  scanf("%*[^\n] ");

  for (i = 0; i < n; ++i) {
    scanf("%lf", &xmul[i]);
  }
  scanf("%*[^\n] ");

  for (i = 0; i < nf; ++i) {
    scanf("%lf", &f[i]);
  }
  scanf("%*[^\n] ");

  for (i = 0; i < nf; ++i) {
    scanf("%ld", &fstate[i]);
  }
  scanf("%*[^\n] ");

  for (i = 0; i < nf; ++i) {
      scanf("%lf", &fmul[i]);
  }
  scanf("%*[^\n] ");

  /* Initialise e04vhc using nag_opt_sparse_nlp_init (e04vgc):
   *  Initialization function for nag_opt_sparse_nlp_solve (e04vhc).
   */
  nag_opt_sparse_nlp_init(&state, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Initialisation of nag_opt_sparse_nlp_init (e04vgc) failed.\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
  }

  /* By default nag_opt_sparse_nlp_solve (e04vhc) does not print monitoring
   * information. Call nag_open_file (x04acc) to set the print file outfileid
   */
  /* nag_open_file (x04acc).
   * Open unit number for reading, writing or appending, and
   * associate unit with named file
   */
  nag_open_file("", 2, &outfileid, &fail);
```

```
  if (fail.code != NE_NOERROR) {
    exit_status = 2;
    goto END;
  }
  /* nag_opt_sparse_nlp_option_set_integer (e04vmc).
   * Set a single option for nag_opt_sparse_nlp_solve (e04vhc)
   * from an integer argument
   */
  nag_opt_sparse_nlp_option_set_integer("Print file", outfileid, &state, &fail);

  if (fail.code != NE_NOERROR) {
    exit_status = 1;
    goto END;
  }

  /* Use nag_opt_sparse_nlp_option_set_file (e04vkc) to read some options from
   * the options file. Call nag_open_file (x04acc) to set the
   * options file optfileid.
   */
  nag_open_file(optionsfile, 0, &optfileid, &fail);
  if (fail.code != NE_NOERROR) {
    nag_close_file(optfileid, &fail);
    exit_status = 1;
    goto END;
  }
  /* nag_opt_sparse_nlp_option_set_file (e04vkc).
   * Supply optional parameter values for
   * nag_opt_sparse_nlp_solve (e04vhc) from external file
   */
  nag_opt_sparse_nlp_option_set_file(optfileid, &state, &fail);
  if (fail.code != NE_NOERROR) {
    nag_close_file(optfileid, &fail);
    exit_status = 1;
    goto END;
  }
  printf("\n");

  /* Find the value of Integer-valued option 'Elastic mode' using
   * nag_opt_sparse_nlp_option_get_integer (e04vrc):
   *  Get the setting of an integer valued option of
   *  nag_opt_sparse_nlp_solve (e04vhc)
   */
  nag_opt_sparse_nlp_option_get_integer("Elastic mode", &elmode, &state, &fail);
  if (fail.code != NE_NOERROR) {
    nag_close_file(optfileid, &fail);
    exit_status = 1;
    goto END;
  }
  printf("Option 'Elastic mode' has the value %3ld.\n", elmode);

  /* Use nag_opt_sparse_nlp_option_set_double (e04vnc) to set the value of
   *  real-valued option 'Infinite bound size'.
   */
  bndinf = 1e10;
  /* nag_opt_sparse_nlp_option_set_double (e04vnc).
   * Set a single option for nag_opt_sparse_nlp_solve (e04vhc)
   * from a double argument
   */
  nag_opt_sparse_nlp_option_set_double("Infinite bound size", bndinf, &state,
                                       &fail);
  if (fail.code != NE_NOERROR) {
    nag_close_file(optfileid, &fail);
    exit_status = 1;
    goto END;
  }

  /* Find the value of real-valued option 'Feasibility tolerance' using
   * nag_opt_sparse_nlp_option_get_double (e04vsc):
   *  Get the setting of a double valued option of
   *  nag_opt_sparse_nlp_solve (e04vhc)
   */
```

```
  nag_opt_sparse_nlp_option_get_double("Feasibility tolerance", &featol,
                                       &state,
                                       &fail);
  if (fail.code != NE_NOERROR) {
    nag_close_file(optfileid, &fail);
    exit_status = 1;
    goto END;
  }
  printf("Option 'Feasibility tolerance' has the value %14.5e.\n",featol);

  /* Set the option 'Major iterations limit' using
   * nag_opt_sparse_nlp_option_set_string (e04vlc):
   *  Set a single option for nag_opt_sparse_nlp_solve (e04vhc)
   *  from a character string
   */
  nag_opt_sparse_nlp_option_set_string("Major iterations limit 50", &state,
                                       &fail);
  if (fail.code != NE_NOERROR) {
    nag_close_file(optfileid, &fail);
    exit_status = 1;
    goto END;
  }
  fflush(stdout);

  /* Solve the problem. */
  /* nag_opt_sparse_nlp_solve (e04vhc).
   * General sparse nonlinear optimizer
   */
  fflush(stdout);
  nag_opt_sparse_nlp_solve(start, nf, n, nxname, nfname, objadd, objrow, prob,
                           usrfun, iafun, javar, a, lena, nea, igfun, jgvar,
                           leng, neg, xlow, xupp, (const char **) xnames, flow,
                           fupp, (const char **) fnames, x, xstate, xmul, f,
                           fstate, fmul, &ns, &ninf, &sinf, &state, &comm,
                           &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_opt_sparse_nlp_solve (e04vhc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }
  nag_close_file(optfileid, &fail);

  printf("Final objective value = %11.1f\n", f[objrow - 1]);
  printf("Optimal X = ");

  for (i = 0; i < n; ++i)
    printf("%9.2f%s", x[i], i%7 == 6 || i == n-1 ? "\n" : " ");

END:
  for (i = 0; i < nxname; i++)
    NAG_FREE(xnames[i]);
  for (i = 0; i < nfname; i++)
    NAG_FREE(fnames[i]);
  NAG_FREE(fnames);
  NAG_FREE(xnames);
  NAG_FREE(prob);
  NAG_FREE(a);
  NAG_FREE(f);
  NAG_FREE(flow);
  NAG_FREE(fmul);
  NAG_FREE(fupp);
  NAG_FREE(x);
  NAG_FREE(xlow);
  NAG_FREE(xmul);
  NAG_FREE(xupp);
  NAG_FREE(fstate);
  NAG_FREE(iafun);
  NAG_FREE(igfun);
  NAG_FREE(iuser);
  NAG_FREE(javar);
  NAG_FREE(jgvar);
```

```
  NAG_FREE(xstate);

  return exit_status;
}

static void NAG_CALL usrfun(Integer *status, Integer n, const double x[],
                            Integer needf, Integer nf, double f[],
                            Integer needg, Integer leng, double g[],
                            Nag_Comm *comm)
{
  if (comm->user[0] == -1.0)
    {
      fflush(stdout);
      printf("(User-supplied callback usrfun, first invocation.)\n");
      comm->user[0] = 0.0;
      fflush(stdout);
    }
  if (needf > 0)
    {
      /* The nonlinear components of f_i(x) need to be assigned, */
      f[0] = sin(-x[0] - .25) * 1e3 + sin(-x[1] - .25) * 1e3;
      f[1] = sin(x[0] - .25) * 1e3 + sin(x[0] - x[1] - .25) * 1e3;
      f[2] = sin(x[1] - x[0] - .25) * 1e3 + sin(x[1] - .25) * 1e3;
      /* N.B. in this example there is no need to assign for the wholly */
      /* linear components f_4(x) and f_5(x). */
      f[5] = x[2] * (x[2] * x[2]) * 1e-6 + x[3] * (x[3] * x[3]) * 2e-6 / 3.;
    }

  if (needg > 0)
    {
      /* The derivatives of the function f_i(x) need to be assigned.
       * g[k-1] should be set to partial derivative df_i(x)/dx_j where
       * i = igfun[k-1] and j = igvar[k-1], for k = 1 to LENG.
       */
      g[0] = cos(-x[0] - .25) * -1e3;
      g[1] = cos(-x[1] - .25) * -1e3;
      g[2] = cos(x[0] - .25) * 1e3 + cos(x[0] - x[1] - .25) * 1e3;
      g[3] = cos(x[0] - x[1] - .25) * -1e3;
      g[4] = cos(x[1] - x[0] - .25) * -1e3;
      g[5] = cos(x[1] - x[0] - .25) * 1e3 + cos(x[1] - .25) * 1e3;
      g[6] = x[2] * x[2] * 3e-6;
      g[7] = x[3] * x[3] * 2e-6;
    }

  return;
} /* usrfun */
```

10.2 Program Data

```
nag_opt_sparse_nlp_option_set_file (e04vkc) Example Program Data
 4   6                : Values of n and nf
 8   8   6   Nag_Cold  : Values of nea, neg, objrow and start

'X1      '  'X2      '  'X3      '  'X4      '  : XNAMES
'NlnCon_1'  'NlnCon_2'  'NlnCon_3'  'LinCon_1'  'LinCon_2'  'Objectiv' : FNAMES

1  3 -1.0E0  : Nonzero elements of sparse matrix A, the linear part of F.
2  4 -1.0E0  : Each row IAFUN(i), JAVAR(i), A(IAFUN(i),JAVAR(i)), i = 1 to nea
4  1 -1.0E0
4  2  1.0E0
5  1  1.0E0
5  2 -1.0E0
6  3  3.0E0
6  4  2.0E0

1  1     : Nonzero row/column structure of G, IGFUN(i), JGVAR(i), i = 1 to neg
1  2
2  1
2  2
3  1
```

```
3  2
6  3
6  4

-0.55E0    0.55E0  : Bounds on the variables, XLOW(i), XUPP(i), for i = 1 to n
-0.55E0    0.55E0
 0.0E0   1200.0E0
 0.0E0   1200.0E0

 -894.8E0 -894.8E0 : Bounds on the functions, FLOW(i), FUPP(i), for i = 1 to nf
 -894.8E0 -894.8E0
-1294.8E0 -1294.8E0
-0.55E0    1.0E25
-0.55E0    1.0E25
-1.0E25    1.0E25

 0.0  0.0  0.0  0.0           : Initial values of X(i), for i = 1 to n
 0    0    0    0             : Initial values of XSTATE(i), for i = 1 to n
 0.0  0.0  0.0  0.0           : Initial values of XMUL(i), for i = 1 to n

 0.0  0.0  0.0  0.0  0.0  0.0 : Initial values of F(i), for i = 1 to nf
 0    0    0    0    0    0   : Initial values of FSTATE(i), for i = 1 to nf
 0.0  0.0  0.0  0.0  0.0  0.0 : Initial values of FMUL(i), for i = 1 to nf
Begin nag_opt_sparse_nlp_option_set_file (e04vkc) 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
```

10.3 Program Results

```
nag_opt_sparse_nlp_option_set_file (e04vkc) Example Program Results


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

      Begin nag_opt_sparse_nlp_option_set_file (e04vkc) 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

 E04VKZ EXIT 100 -- finished successfully
 E04VKZ 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)         6

 The SQP Method
 --------------
 Minimize...............                Cold start.............                Proximal Point method..         1
 Nonlinear objectiv vars         4      Objective Row..........         6      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 option......         1
 Major iterations limit.        50      Linesearch tolerance...   0.90000      Verify level...........         0
 Minor iterations limit.       500      Penalty parameter......  0.00E+00      Major Print Level......         1
                                        Major optimality tol...  2.00E-06

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

 Nonlinear constraints
 ---------------------
 Nonlinear constraints..         3      Major feasibility tol..  1.00E-06      Violation limit........  1.00E+06
 Nonlinear Jacobian vars         2

 Miscellaneous
 -------------
 LU factor tolerance....      3.99      LU singularity tol.....  2.05E-11      Timing level...........         0
 LU update tolerance....      3.99      LU swap tolerance......  1.03E-04      Debug level............         0
 LU partial  pivoting...                eps (machine precision)  1.11E-16      System information.....        No



 Matrix statistics
 -----------------
               Total      Normal        Free       Fixed     Bounded
 Rows              6           2           1           3           0
 Columns           4           0           0           0           4

 No. of matrix elements                   14     Density      58.333
 Biggest                         1.0000E+00  (excluding fixed columns,
 Smallest                        0.0000E+00   free rows, and RHS)

 No. of objective coefficients             2
 Biggest                         3.0000E+00  (excluding fixed columns)
 Smallest                        2.0000E+00

 Nonlinear constraints       3     Linear constraints       3
 Nonlinear variables         4     Linear variables         0
 Jacobian  variables         2     Objective variables      4
 Total constraints           6     Total variables          4



(User-supplied callback usrfun, first invocation.)
 The user has defined       8   out of       8   first  derivatives

 Cheap test of user-supplied problem derivatives...

 The constraint gradients seem to be OK.

 -->  The largest discrepancy was    2.23E-08  in constraint     7
```

```
 The objective  gradients seem to be OK.

 Gradient projected in one direction   0.00000000000E+00
 Difference approximation              4.49060460280E-21



   Itns Major Minors    Step   nCon  Feasible   Optimal  MeritFunction     L+U BSwap     nS  condHz Penalty
      3     0      3             1   8.0E+02   1.0E-00  0.0000000E+00      17            1 1.7E+07         _  r
      4     1      1 1.2E-03     2   4.0E+02   9.9E-01  9.6317131E+05      16            1 4.8E+06 2.8E+00 _n rl
      5     2      1 1.3E-03     3   2.7E+02   5.5E-01  9.6122945E+05      16                      2.8E+00 _s  l
      5     3      0 7.5E-03     4   8.8E+01   5.4E-01  9.4691061E+05      16                      2.8E+00 _   l
      5     4      0 2.3E-02     5   2.9E+01   5.3E-01  9.0468403E+05      16                      2.8E+00 _   l
      5     5      0 6.9E-02     6   8.9E+00   5.0E-01  7.8452897E+05      16                      2.8E+00 _   l
      6     6      1 2.2E-01     7   2.3E+00   5.5E+01  4.8112339E+05      16            1 8.7E+03 2.8E+00 _   l
      7     7      1 8.3E-01     8   1.7E-01   4.2E+00  2.6898257E+04      16            1 7.6E+03 2.8E+00 _   l
      8     8      1 1.0E+00     9   1.8E-02   8.7E+01  6.2192920E+03      15     1      1 1.2E+02 2.8E+00 _
      9     9      1 1.0E+00    10   1.7E-02   7.9E+00  5.4526185E+03      15            1 9.4E+01 2.8E+00 _
     10    10      1 1.0E+00    11   1.7E-04   9.6E-01  5.1266089E+03      15            1 1.0E+02 2.8E+00 _
     11    11      1 1.0E+00    12   1.7E-06   5.8E-02  5.1264988E+03      15            1 9.5E+01 2.8E+00 _
     12    12      1 1.0E+00    13 ( 1.2E-08)  6.9E-05  5.1264981E+03      15            1 9.5E+01 2.8E+00 _
     13    13      1 1.0E+00    14 ( 6.7E-15)( 3.0E-09) 5.1264981E+03      15            1 9.5E+01 6.0E+00 _

 E04VHU EXIT   0 -- finished successfully
 E04VHU INFO   1 -- optimality conditions satisfied

 Problem name
 No. of iterations                 13   Objective value      5.1264981096E+03
 No. of major iterations           13   Linear objective     4.0919702248E+03
 Penalty parameter          6.029E+00   Nonlinear objective  1.0345278848E+03
 No. of calls to funobj            15   No. of calls to funcon            15
 No. of superbasics                 1   No. of basic nonlinears            3
 No. of degenerate steps            0   Percentage                      0.00
 Max x                      4 1.0E+03   Max pi                     3 5.5E+00
 Max Primal infeas          0 0.0E+00   Max Dual infeas            1 4.6E-08
 Nonlinear constraint violn   5.7E-12

 Name                                   Objective Value      5.1264981096E+03

 Status         Optimal Soln            Iteration     13    Superbasics     1

 Objective              (Min)
 RHS
 Ranges
 Bounds

 Section 1 - Rows

  Number  ...Row.. State  ...Activity...  Slack Activity  ..Lower Limit.  ..Upper Limit.  .Dual Activity    ..i

       5  NlnCon_1    EQ      -894.80000         0.00000      -894.80000      -894.80000        -4.38698       1
       6  NlnCon_2    EQ      -894.80000         0.00000      -894.80000      -894.80000        -4.10563       2
       7  NlnCon_3    EQ     -1294.80000         0.00000     -1294.80000     -1294.80000        -5.46328       3
       8  LinCon_1    BS        -0.51511         0.03489        -0.55000            None             .          4
       9  LinCon_2    BS         0.51511         1.06511        -0.55000            None             .          5
      10  Objectiv    BS      4091.97022      4091.97022            None            None         -1.0           6

 Section 2 - Columns

  Number  .Column. State  ...Activity...  .Obj Gradient.  ..Lower Limit.  ..Upper Limit.  Reduced Gradnt    m+j

       1  X1          BS         0.11888          .             -0.55000         0.55000        -0.00000       7
       2  X2          BS        -0.39623          .             -0.55000         0.55000         0.00000       8
       3  X3         SBS       679.94532         4.38698          .           1200.00000         0.00000       9
       4  X4          BS      1026.06713         4.10563          .           1200.00000        -0.00000      10
 Final objective value =      5126.5
 Optimal X =      0.12     -0.40    679.95   1026.07
```