NAG Library Function Document

nag_opt_sparse_convex_qp_option_set_file (e04nrc)

1 Purpose

nag_opt_sparse_convex_qp_option_set_file (e04nrc) may be used to supply optional parameters to nag_opt_sparse_convex_qp_solve (e04nqc) from an external file. The initialization function nag_opt_sparse_convex_qp_init (e04npc) **must** have been called before calling nag_opt_sparse_convex_qp_option_set_file (e04nrc).

2 Specification

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

3 Description

nag_opt_sparse_convex_qp_option_set_file (e04nrc) may be used to supply values for optional parameters to nag_opt_sparse_convex_qp_solve (e04nqc). nag_opt_sparse_convex_qp_option_set_file (e04nrc) 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 parameter. It is only necessary to supply values for those arguments whose values are to be different from their default values.

Each optional parameter 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 parameter. 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 40 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 parameter settings are preserved following a call to nag_opt_sparse_convex_qp_solve (e04nqc) and so the keyword **Defaults** is provided to allow you to reset all the optional parameters to their default values before a subsequent call to nag_opt_sparse_convex_qp_solve (e04nqc).

A complete list of optional parameters, their abbreviations, synonyms and default values is given in Section 12 in nag_opt_sparse_convex_qp_solve (e04nqc).

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 2.7 in How to Use the NAG Library and its Documentation).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 2.3.1.2 in How to Use the NAG Library and its Documentation for further information.

NE_BAD_PARAM

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

NE_E04NPC_NOT_INIT

The initialization function nag_opt_sparse_convex_qp_init (e04npc) 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 2.7.6 in How to Use the NAG Library and its Documentation for further information.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
See Section 2.7.5 in How to Use the NAG Library and its Documentation for further information.

NE_OPTIONS_FILE_READ_FAILURE

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.

7 Accuracy

Not applicable.

8 Parallelism and Performance

nag_opt_sparse_convex_qp_option_set_file (e04nrc) is not threaded in any implementation.

9 Further Comments

nag_opt_sparse_convex_qp_option_set_string (e04nsc), nag_opt_sparse_convex_qp_option_set_integer (e04ntc) or nag_opt_sparse_convex_qp_option_set_double (e04nuc) may also be used to supply optional parameters to nag_opt_sparse_convex_qp_solve (e04nqc).

10 Example

This example minimizes the quadratic function $f(x) = c^{\mathrm{T}}x + \frac{1}{2}x^{\mathrm{T}}Hx$, where

$$c = (-200.0, -2000.0, -2000.0, -2000.0, -2000.0, 400.0, 400.0)^{\mathrm{T}}$$

and

$$H = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 & 2 & 2 \end{pmatrix}$$

subject to the bounds

$$\begin{aligned} 0 &\leq x_1 \leq 200 \\ 0 &\leq x_2 \leq 2500 \\ 400 &\leq x_3 \leq 800 \\ 100 &\leq x_4 \leq 700 \\ 0 &\leq x_5 \leq 1500 \\ 0 &\leq x_6 \\ 0 &\leq x_7 \end{aligned}$$

and to the linear constraints

$$\begin{array}{rcrcrcrcrcrcrcr} & & x_1 & + & x_2 & + & x_3 & + & x_4 & + & x_5 & + & x_6 & + & x_7 & = & 2000 \\ & & 0.15x_1 & + & 0.04x_2 & + & 0.02x_3 & + & 0.04x_4 & + & 0.02x_5 & + & 0.01x_6 & + & 0.03x_7 & \leq & 60 \\ & & 0.03x_1 & + & 0.05x_2 & + & 0.08x_3 & + & 0.02x_4 & + & 0.06x_5 & + & 0.01x_6 & & & \leq & 100 \\ & & 0.02x_1 & + & 0.04x_2 & + & 0.01x_3 & + & 0.02x_4 & + & 0.02x_5 & & & & & \leq & 40 \\ & & 0.02x_1 & + & 0.03x_2 & + & & & & & 0.01x_5 & & & & & \leq & 30 \\ 1500 & \leq & 0.70x_1 & + & 0.75x_2 & + & 0.80x_3 & + & 0.75x_4 & + & 0.80x_5 & + & 0.97x_6 & & & & \\ 250 & \leq & 0.02x_1 & + & 0.06x_2 & + & 0.08x_3 & + & 0.12x_4 & + & 0.02x_5 & + & 0.01x_6 & + & 0.97x_7 & \leq & 300 \end{array}$$

The initial point, which is infeasible, is

$$x_0 = (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0)^{\mathrm{T}}.$$

The optimal solution (to five figures) is

$$x^* = (0.0, 349.40, 648.85, 172.85, 407.52, 271.36, 150.02)^{\mathrm{T}}.$$

One bound constraint and four linear constraints are active at the solution. Note that the Hessian matrix H is positive semidefinite.

10.1 Program Text

```
/* nag_opt_sparse_convex_qp_option_set_file (e04nrc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

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

#ifdef __cplusplus
extern "C"
{
#endif
  static void NAG_CALL qphx(Integer ncolh, const double x[], double hx[],
                            Integer nstate, Nag_Comm *comm);
#ifdef __cplusplus
}
#endif

int main(void)
{
  const char *optionsfile = "e04nrce.opt";
  Integer exit_status = 0;

  /* Scalars */
  double bndinf, featol, obj, objadd, sinf;
  Integer elmode, i, icol, iobj, j, jcol, lenc, m, n, ncolh, ne;
  Integer ninf, nname, ns;

  /* Arrays */
  static double ruser[1] = { -1.0 };
  char nag_enum_arg[40];
  char *cuser = 0, *prob = 0;
  char **names = 0;
  double *acol = 0, *bl = 0, *bu = 0, *c = 0, *pi = 0, *rc = 0;
  double *x = 0;
  Integer *helast = 0, *hs = 0, *inda = 0, *iuser = 0, *loca = 0;

  /* Nag Types */
  Nag_E04State state;
  Nag_Start start;
  Nag_Comm comm;
  Nag_FileID fileidin;
  NagError fail;

  /* By default e04nrc does not print monitoring information.
     Define SHOW_MONITORING_INFO to turn it on - see further below. */
#ifdef SHOW_MONITORING_INFO
  Nag_FileID fileidout;
#endif

  INIT_FAIL(fail);

  printf("%s", "nag_opt_sparse_convex_qp_option_set_file (e04nrc) Example"
         " Program Results\n");
  printf("\n");

  /* For communication with user-supplied functions: */
  comm.user = ruser;

  /* This program demonstrates the use of routines to set and
   * get values of optional parameters associated with
   * nag_opt_sparse_convex_qp_solve (e04nqc).
```

```
   */

  /* Skip heading in data file. */
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif
#ifdef _WIN32
  scanf_s("%" NAG_IFMT " %" NAG_IFMT " ", &n, &m);
#else
  scanf("%" NAG_IFMT " %" NAG_IFMT " ", &n, &m);
#endif
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

  if (n >= 1 && m >= 1) {
    /* Read ne, iobj, ncolh, start and nname from data file. */
#ifdef _WIN32
    scanf_s("%" NAG_IFMT " %" NAG_IFMT " %" NAG_IFMT " %39s %" NAG_IFMT "",
            &ne, &iobj, &ncolh, nag_enum_arg,
            (unsigned)_countof(nag_enum_arg), &nname);
#else
    scanf("%" NAG_IFMT " %" NAG_IFMT " %" NAG_IFMT " %39s %" NAG_IFMT "",
          &ne, &iobj, &ncolh, nag_enum_arg, &nname);
#endif
#ifdef _WIN32
    scanf_s("%*[^\n] ");
#else
    scanf("%*[^\n] ");
#endif
    /* nag_enum_name_to_value (x04nac).
     * Converts NAG enum member name to value
     */
    start = (Nag_Start) nag_enum_name_to_value(nag_enum_arg);

    /* Allocate memory */
    if (!(names = NAG_ALLOC(n + m, char *)) ||
        !(prob = NAG_ALLOC(9, char)) ||
        !(acol = NAG_ALLOC(ne, double)) ||
        !(bl = NAG_ALLOC(m + n, double)) ||
        !(bu = NAG_ALLOC(m + n, double)) ||
        !(c = NAG_ALLOC(1, double)) ||
        !(pi = NAG_ALLOC(m, double)) ||
        !(rc = NAG_ALLOC(n + m, double)) ||
        !(x = NAG_ALLOC(n + m, double)) ||
        !(helast = NAG_ALLOC(n + m, Integer)) ||
        !(hs = NAG_ALLOC(n + m, Integer)) ||
        !(inda = NAG_ALLOC(ne, Integer)) ||
        !(iuser = NAG_ALLOC(1, Integer)) ||
        !(loca = NAG_ALLOC(n + 1, Integer)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }
  }
  else {
    printf("Invalid n or nf or nea or neg\n");
    exit_status = 1;
    goto END;
  }
  /* Read names from data file. */

  for (i = 1; i <= nname; ++i) {
    names[i - 1] = NAG_ALLOC(9, char);
#ifdef _WIN32
    scanf_s(" ' %8s '", names[i - 1], 9);
```

```
#else
    scanf(" ' %8s '", names[i - 1]);
#endif
  }
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

  /* Read the matrix acol from data file. Set up loca. */
  jcol = 1;
  loca[jcol - 1] = 1;
  for (i = 1; i <= ne; ++i) {
    /* Element ( inda[i-1], icol ) is stored in acol[i-1]. */

#ifdef _WIN32
    scanf_s("%lf %" NAG_IFMT " %" NAG_IFMT "", &acol[i - 1], &inda[i - 1],
            &icol);
#else
    scanf("%lf %" NAG_IFMT " %" NAG_IFMT "", &acol[i - 1], &inda[i - 1],
          &icol);
#endif
#ifdef _WIN32
    scanf_s("%*[^\n] ");
#else
    scanf("%*[^\n] ");
#endif

    if (icol < jcol) {
      /* Elements not ordered by increasing column index. */
      printf("%s %5" NAG_IFMT " %s %5" NAG_IFMT "",
             "Element in column", icol,
             " found after element in column", jcol);
      printf("%s\n\n", ". Problem abandoned.");
    }
    else if (icol == jcol + 1) {
      /* Index in acol of the start of the icol-th column equals i. */
      loca[icol - 1] = i;
      jcol = icol;
    }
    else if (icol > jcol + 1) {
      /* Index in acol of the start of the icol-th column equals i,
       * but columns jcol+1,jcol+2,...,icol-1 are empty. Set the
       * corresponding elements of loca to i.
       */
      for (j = jcol + 1; j <= icol - 1; ++j) {
        loca[j - 1] = i;
      }
      loca[icol - 1] = i;
      jcol = icol;
    }
  }
  loca[n] = ne + 1;
  if (n > icol) {
    /* Columns n,n-1,...,icol+1 are empty. Set the corresponding */
    /* elements of loca accordingly. */
    for (i = n; i >= icol + 1; --i) {
      loca[i - 1] = loca[i];
    }
  }

  /* Read bl, bu, hs and x from data file. */
  for (i = 1; i <= n + m; ++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 + m; ++i) {
#ifdef _WIN32
    scanf_s("%lf", &bu[i - 1]);
#else
    scanf("%lf", &bu[i - 1]);
#endif
  }
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

  if (start == Nag_Cold) {
    for (i = 1; i <= n; ++i) {
#ifdef _WIN32
      scanf_s("%" NAG_IFMT "", &hs[i - 1]);
#else
      scanf("%" NAG_IFMT "", &hs[i - 1]);
#endif
    }
#ifdef _WIN32
    scanf_s("%*[^\n] ");
#else
    scanf("%*[^\n] ");
#endif
  }
  else if (start == Nag_Warm) {
    for (i = 1; i <= n + m; ++i) {
#ifdef _WIN32
      scanf_s("%" NAG_IFMT "", &hs[i - 1]);
#else
      scanf("%" NAG_IFMT "", &hs[i - 1]);
#endif
    }
#ifdef _WIN32
    scanf_s("%*[^\n] ");
#else
    scanf("%*[^\n] ");
#endif
  }

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

  /* We have no explicit objective vector so set lenc = 0; the
   * objective vector is stored in row iobj of acol.
   */
  lenc = 0;
  objadd = 0.;
#ifdef _WIN32
  strcpy_s(prob, 9, "        ");
#else
  strcpy(prob, "        ");
#endif
```

```
  /* nag_opt_sparse_convex_qp_init (e04npc).
   * Initialization function for
   * nag_opt_sparse_convex_qp_solve (e04nqc)
   */
  nag_opt_sparse_convex_qp_init(&state, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Initialization of nag_opt_sparse_convex_qp_solve (e04nqc)"
           " failed.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }

#ifdef SHOW_MONITORING_INFO
  /*  Call nag_open_file (x04acc) to set the print file fileidout */

  /* 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) {
    exit_status = 2;
    goto END;
  }
  /* nag_opt_sparse_convex_qp_option_set_integer (e04ntc).
   * Set a single option for nag_opt_sparse_convex_qp_solve
   * (e04nqc) from an integer argument
   */
  fflush(stdout);
  nag_opt_sparse_convex_qp_option_set_integer("Print file", fileidout, &state,
                                              &fail);
  if (fail.code != NE_NOERROR) {
    exit_status = 1;
    goto END;
  }
#endif

  /* Set input to options file to read. */
  /* nag_open_file (x04acc), see above. */
  nag_open_file(optionsfile, 0, &fileidin, &fail);
  if (fail.code != NE_NOERROR) {
    exit_status = 1;
    goto END;
  }
  /* nag_opt_sparse_convex_qp_option_set_file (e04nrc).
   * Supply optional parameter values for
   * nag_opt_sparse_convex_qp_solve (e04nqc) from external
   * file
   */
  nag_opt_sparse_convex_qp_option_set_file(fileidin, &state, &fail);
  if (fail.code != NE_NOERROR) {
    exit_status = 1;
    goto END;
  }
  nag_close_file(fileidin, &fail); /* Close Library output */

  /* Use nag_opt_sparse_convex_qp_option_get_integer (e04nxc) to find the value
   * of Integer-valued option 'Elastic mode'.
   */
  /* nag_opt_sparse_convex_qp_option_get_integer (e04nxc).
   * Get the setting of an integer valued option of
   * nag_opt_sparse_convex_qp_solve (e04nqc)
   */
  nag_opt_sparse_convex_qp_option_get_integer("Elastic mode", &elmode, &state,
                                              &fail);
  if (fail.code != NE_NOERROR) {
    exit_status = 1;
    goto END;
  }
  printf("Option 'Elastic mode' has the value %3" NAG_IFMT ".\n", elmode);
  /* Use nag_opt_sparse_convex_qp_option_set_double (e04nuc) to set the value of
```

```
   *  real-valued option 'Infinite bound size'.
   */
  bndinf = 1e10;
  /* nag_opt_sparse_convex_qp_option_set_double (e04nuc).
   * Set a single option for nag_opt_sparse_convex_qp_solve
   * (e04nqc) from a double argument
   */
  nag_opt_sparse_convex_qp_option_set_double("Infinite bound size", bndinf,
                                             &state, &fail);
  if (fail.code != NE_NOERROR) {
    exit_status = 1;
    goto END;
  }

  /* Use nag_opt_sparse_convex_qp_option_get_double (e04nyc) to find the value
   *  of real-valued option 'Feasibility tolerance'.
   */
  /* nag_opt_sparse_convex_qp_option_get_double (e04nyc).
   * Get the setting of a double valued option of
   * nag_opt_sparse_convex_qp_solve (e04nqc)
   */
  nag_opt_sparse_convex_qp_option_get_double("Feasibility tolerance", &featol,
                                             &state, &fail);
  if (fail.code != NE_NOERROR) {
    exit_status = 1;
    goto END;
  }
  printf("Option 'Feasibility tolerance' has the value %14.5e.\n", featol);

  /* Use nag_opt_sparse_convex_qp_option_set_string (e04nsc) to set the option
   *  'Iterations limit'.
   */
  /* nag_opt_sparse_convex_qp_option_set_string (e04nsc).
   * Set a single option for nag_opt_sparse_convex_qp_solve
   * (e04nqc) from a character string
   */
  nag_opt_sparse_convex_qp_option_set_string("Iterations limit 50", &state,
                                             &fail);
  if (fail.code != NE_NOERROR) {
    exit_status = 1;
    goto END;
  }
  printf("\n");
  fflush(stdout);

#ifdef SHOW_MONITORING_INFO
  nag_open_file("", 2, &fileidout, &fail);
  fflush(stdout);
#endif
  /* Solve the QP problem. */
  /* nag_opt_sparse_convex_qp_solve (e04nqc).
   * LP or QP problem (suitable for sparse problems)
   */
  nag_opt_sparse_convex_qp_solve(start, qphx, m, n, ne, nname, lenc, ncolh,
                                 iobj, objadd, prob, acol, inda, loca, bl, bu,
                                 c, (const char **) names, helast, hs, x, pi,
                                 rc,
                                 &ns, &ninf, &sinf, &obj, &state, &comm,
                                 &fail);

  if (n >= 1 && m >= 1) {
    for (i = 1; i <= nname; ++i) {
      NAG_FREE(names[i - 1]);
    }
  }

  if (fail.code == NE_NOERROR) {
    printf("Final objective value = %12.3e\n", obj);
    printf("Optimal X = ");

    for (i = 1; i <= n; ++i) {
```

```
      printf("%8.2f%s", x[i - 1], i % 7 == 0 || i == n ? "\n" : " ");
    }
  }
  else {
    printf("Error from nag_opt_sparse_convex_qp_solve (e04nqc).\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
  }

  if (fail.code != NE_NOERROR) {
    exit_status = 2;
  }

END:
  NAG_FREE(cuser);
  NAG_FREE(names);
  NAG_FREE(prob);
  NAG_FREE(acol);
  NAG_FREE(bl);
  NAG_FREE(bu);
  NAG_FREE(c);
  NAG_FREE(pi);
  NAG_FREE(rc);
  NAG_FREE(x);
  NAG_FREE(helast);
  NAG_FREE(hs);
  NAG_FREE(inda);
  NAG_FREE(iuser);
  NAG_FREE(loca);

  return exit_status;
}

static void NAG_CALL qphx(Integer ncolh, const double x[], double hx[],
                          Integer nstate, Nag_Comm *comm)
{
  /* Routine to compute H*x. (In this version of qphx, the Hessian
   * matrix H is not referenced explicitly.)
   */

  /* Parameter adjustments */
#define HX(I) hx[(I) -1]
#define X(I)  x[(I) -1]

  /* Function Body */
  if (comm->user[0] == -1.0) {
    fflush(stdout);
    printf("(User-supplied callback qphx, first invocation.)\n");
    comm->user[0] = 0.0;
    fflush(stdout);
  }
  HX(1) = X(1) * 2;
  HX(2) = X(2) * 2;
  HX(3) = (X(3) + X(4)) * 2;
  HX(4) = HX(3);
  HX(5) = X(5) * 2;
  HX(6) = (X(6) + X(7)) * 2;
  HX(7) = HX(6);
  return;
} /* qphx */
```

10.2 Program Data

```
Begin nag_opt_sparse_convex_qp_option_set_file (e04nrc) example options
* 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 1.0D-4
End
```

```
nag_opt_sparse_convex_qp_option_set_file (e04nrc) Example Program Data
 7  8                     : Values of n and m
48  8  7  Nag_Cold  15     : Values of nnz, iobj, ncolh, start and nname

'...X1...'  '...X2...'  '...X3...'  '...X4...'  '...X5...'
'...X6...'  '...X7...'  '..ROW1..'  '..ROW2..'  '..ROW3..'
'..ROW4..'  '..ROW5..'  '..ROW6..'  '..ROW7..'  '..COST..' : End of array NAMES

    0.02   7   1   : Sparse matrix A, ordered by increasing column index;
    0.02   5   1   : each row contains ACOL(i), INDA(i), ICOL (= column index)
    0.03   3   1   : The row indices may be in any order. In this example
    1.00   1   1   : row 8 defines the linear objective term transpose(C)*X.
    0.70   6   1
    0.02   4   1
    0.15   2   1
 -200.00   8   1
    0.06   7   2
    0.75   6   2
    0.03   5   2
    0.04   4   2
    0.05   3   2
    0.04   2   2
    1.00   1   2
-2000.00   8   2
    0.02   2   3
    1.00   1   3
    0.01   4   3
    0.08   3   3
    0.08   7   3
    0.80   6   3
-2000.00   8   3
    1.00   1   4
    0.12   7   4
    0.02   3   4
    0.02   4   4
    0.75   6   4
    0.04   2   4
-2000.00   8   4
    0.01   5   5
    0.80   6   5
    0.02   7   5
    1.00   1   5
    0.02   2   5
    0.06   3   5
    0.02   4   5
-2000.00   8   5
    1.00   1   6
    0.01   2   6
    0.01   3   6
    0.97   6   6
    0.01   7   6
  400.00   8   6
    0.97   7   7
    0.03   2   7
    1.00   1   7
  400.00   8   7        : End of matrix A

 0.0       0.0       4.0E+02   1.0E+02   0.0       0.0
 0.0       2.0E+03  -1.0E+25  -1.0E+25  -1.0E+25  -1.0E+25
```

```
 1.5E+03   2.5E+02  -1.0E+25             : End of lower bounds array BL

 2.0E+02   2.5E+03   8.0E+02   7.0E+02  1.5E+03  1.0E+25
 1.0E+25   2.0E+03   6.0E+01   1.0E+02  4.0E+01  3.0E+01
 1.0E+25   3.0E+02   1.0E+25             : End of upper bounds array BU

 0    0    0    0    0    0    0         : Initial array HS
 0.0  0.0  0.0  0.0  0.0  0.0  0.0       : Initial vector X
```

10.3 Program Results

```
nag_opt_sparse_convex_qp_option_set_file (e04nrc) Example Program Results

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

(User-supplied callback qphx, first invocation.)
Final objective value =   -1.848e+06
Optimal X =     0.00   349.40   648.85   172.85   407.52   271.36   150.02
```