NAG Library Function Document

nag_opt_handle_set_linconstr (e04rjc)

1 Purpose

nag_opt_handle_set_linconstr (e04rjc) is a part of the NAG optimization modelling suite and defines the block of linear constraints of the problem.

2 Specification

```
#include <nag.h>
#include <nage04.h>
void nag_opt_handle_set_linconstr (void *handle, Integer nclin,
     const double bl[], const double bu[], Integer nnzb,
     const Integer irowb[], const Integer icolb[], const double b[],
     Integer *idlc, NagError *fail)
```

3 Description

After the initialization function nag_opt_handle_init (e04rac) has been called, nag_opt_handle_set_lin constr (e04rjc) may be used to define the linear constraints $l_B \leq Bx \leq u_B$ of the problem unless the linear constraints have already been defined. This will typically be used for problems, such as quadratic programming (QP)

$$\begin{array}{lll} \underset{x\in\mathbb{R}^n}{\text{minimize}} & \frac{1}{2}x^{\mathrm{T}}Hx + c^{\mathrm{T}}x & \text{(a)} \\ \text{subject to} & l_B \leq Bx \leq u_B & \text{(b)} \\ & l_x \leq x \leq u_x, & \text{(c)} \end{array} \tag{1}$$

nonlinear programming (NLP)

$$\begin{array}{lll} \underset{x\in\mathbb{R}^n}{\text{minimize}} & f(x) & \text{(a)} \\ \text{subject to} & l_g \leq g(x) \leq u_g & \text{(b)} \\ & l_B \leq Bx \leq u_B & \text{(c)} \\ & l_x \leq x \leq u_x & \text{(d)} \end{array} \tag{2}$$

linear semidefinite programming (SDP)

$$\begin{array}{lll} \underset{x\in\mathbb{R}^n}{\text{minimize}} & c^{\mathrm{T}}x & \text{(a)} \\ \text{subject to} & \sum_{i=1}^{n} x_i A_i^k - A_0^k \succeq 0, \quad k = 1,\ldots,m_A & \text{(b)} \\ & l_B \leq Bx \leq u_B & \text{(c)} \\ & l_x \leq x \leq u_x & \text{(d)} \end{array} \tag{3}$$

or semidefinite programming with bilinear matrix inequalities (BMI-SDP)

$$\begin{array}{lll} \underset{x\in\mathbb{R}^n}{\text{minimize}} & \frac{1}{2}x^{\mathrm{T}}Hx + c^{\mathrm{T}}x & \text{(a)} \\ \text{subject to} & \sum_{i,j=1}^{n} x_i x_j Q_{ij}^k + \sum_{i=1}^{n} x_i A_i^k - A_0^k \succeq 0, \quad k = 1,\ldots,m_A & \text{(b)} \\ & l_B \leq Bx \leq u_B & \text{(c)} \\ & l_x \leq x \leq u_x & \text{(d)} \end{array} \tag{4}$$

where n is the number of decision variables, B is a general $m_B \times n$ rectangular matrix and l_B and u_B are m_B-dimensional vectors. Note that upper and lower bounds are specified for all the constraints. This form allows full generality in specifying various types of constraint. In particular, the jth constraint may be defined as an equality by setting $l_j = u_j$. If certain bounds are not present, the associated elements of

l_B or u_B may be set to special values that are treated as $-\infty$ or $+\infty$. See the description of the optional parameter **Infinite Bound Size** of the solvers in the suite, nag_opt_handle_solve_ipopt (e04stc) and nag_opt_handle_solve_pennon (e04svc). Its value is denoted as $bigbnd$ further in this text. Note that the bounds are interpreted based on its value at the time of calling this function and any later alterations to **Infinite Bound Size** will not affect these constraints.

See nag_opt_handle_init (e04rac) for more details.

4 References

None.

5 Arguments

1: **handle** – void * *Input*

On entry: the handle to the problem. It needs to be initialized by nag_opt_handle_init (e04rac) and **must not** be changed.

2: **nclin** – Integer *Input*

On entry: m_B, the number of linear constraints (number of rows of the matrix B).

If **nclin** $= 0$, no linear constraints will be defined and **bl**, **bu**, **nnzb**, **irowb**, **icolb** and **b** will not be referenced and may be **NULL**.

Constraint: **nclin** ≥ 0.

3: **bl**[**nclin**] – const double *Input*

4: **bu**[**nclin**] – const double *Input*

On entry: **bl** and **bu** define lower and upper bounds of the linear constraints, l_B and u_B, respectively. To define the jth constraint as equality, set **bl**$[j-1]$ = **bu**$[j-1] = \beta$, where $|\beta| < bigbnd$. To specify a nonexistent lower bound (i.e., $l_j = -\infty$), set **bl**$[j-1] \leq -bigbnd$; to specify a nonexistent upper bound, set **bu**$[j-1] \geq bigbnd$.

Constraints:

bl$[j-1] \leq$ **bu**$[j-1]$, for $j = 1, 2, \ldots,$ **nclin**;
bl$[j-1] < bigbnd$, for $j = 1, 2, \ldots,$ **nclin**;
bu$[j-1] > -bigbnd$, for $j = 1, 2, \ldots,$ **nclin**;
if **bl**$[j-1] =$ **bu**$[j-1]$, $|$**bl**$[j-1]| < bigbnd$, for $j = 1, 2, \ldots,$ **nclin**.

5: **nnzb** – Integer *Input*

On entry: **nnzb** gives the number of nonzeros in matrix B.

Constraint: if **nclin** > 0, **nnzb** > 0.

6: **irowb**[**nnzb**] – const Integer *Input*

7: **icolb**[**nnzb**] – const Integer *Input*

8: **b**[**nnzb**] – const double *Input*

On entry: arrays **irowb**, **icolb** and **b** store **nnzb** nonzeros of the sparse matrix B in coordinate storage (CS) format (see Section 2.1.1 in the f11 Chapter Introduction). The matrix B has dimensions $m_B \times n$, where n, the number of variables in the problem, was set in **nvar** during the initialization of the handle by nag_opt_handle_init (e04rac). **irowb** specifies one-based row indices, **icolb** specifies one-based column indices and **b** specifies the values of the nonzero elements in such a way that $b_{ij} =$ **b**$[l-1]$ where $i =$ **irowb**$[l-1]$ and $j =$ **icolb**$[l-1]$, for $l = 1, 2, \ldots,$ **nnzb**. No particular order of elements is expected, but elements should not repeat.

Constraint: $1 \leq$ **irowb**$[l-1] \leq$ **nclin**, $1 \leq$ **icolb**$[l-1] \leq n$, for $l = 1, 2, \ldots,$ **nnzb**.

9: **idlc** – Integer * *Input/Output*

Note: **idlc** is reserved for future releases of the NAG C Library.

On entry: if **idlc** $= 0$, new linear constraints are added to the problem definition. This is the only value allowed at the moment.

Constraint: **idlc** $= 0$.

On exit: the number of the last linear constraint added, thus **nclin**.

10: **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 3.2.1.2 in How to Use the NAG Library and its Documentation for further information.

NE_ALREADY_DEFINED

A set of linear constraints has already been defined.

NE_BAD_PARAM

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

NE_BOUND

On entry, $j = \langle value \rangle$, **bl**$[j-1] = \langle value \rangle$, $bigbnd = \langle value \rangle$.
Constraint: **bl**$[j-1] < bigbnd$.

On entry, $j = \langle value \rangle$, **bl**$[j-1] = \langle value \rangle$ and **bu**$[j-1] = \langle value \rangle$.
Constraint: **bl**$[j-1] \leq$ **bu**$[j-1]$.

On entry, $j = \langle value \rangle$, **bu**$[j-1] = \langle value \rangle$, $bigbnd = \langle value \rangle$.
Constraint: **bu**$[j-1] > -bigbnd$.

NE_HANDLE

The supplied **handle** does not define a valid handle to the data structure for the NAG optimization modelling suite. It has not been initialized by nag_opt_handle_init (e04rac) or it has been corrupted.

NE_INT

On entry, **nclin** $= \langle value \rangle$.
Constraint: **nclin** ≥ 0.

On entry, **nnzb** $= \langle value \rangle$.
Constraint: **nnzb** > 0.

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 How to Use the NAG Library and its Documentation for further information.

NE_INVALID_CS

On entry, $i = \langle value \rangle$, **icolb**$[i-1] = \langle value \rangle$ and $n = \langle value \rangle$.
Constraint: $1 \leq$ **icolb**$[i-1] \leq n$.

On entry, $i = \langle value \rangle$, **irowb**$[i-1] = \langle value \rangle$ and **nclin** $= \langle value \rangle$.
Constraint: $1 \leq$ **irowb**$[i-1] \leq$ **nclin**.

On entry, more than one element of **b** has row index $\langle value \rangle$ and column index $\langle value \rangle$.
Constraint: each element of **b** must have a unique row and column index.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
See Section 3.6.5 in How to Use the NAG Library and its Documentation for further information.

NE_PHASE

The problem cannot be modified in this phase any more, the solver has already been called.

NE_REF_MATCH

On entry, **idlc** $= \langle value \rangle$.
Constraint: **idlc** $= 0$.

7 Accuracy

Not applicable.

8 Parallelism and Performance

nag_opt_handle_set_linconstr (e04rjc) is not threaded in any implementation.

9 Further Comments

None.

10 Example

This example demonstrates how to use the MPS file reader nag_opt_miqp_mps_read (e04mxc) and this suite of functions to define and solve a QP problem. nag_opt_miqp_mps_read (e04mxc) uses a different output format to the one required by nag_opt_handle_set_linconstr (e04rjc), in particular, it uses the compressed column storage (CCS) (see Section 2.1.3 in the f11 Chapter Introduction) instead of the coordinate storage and the linear objective vector is included in the system matrix. Therefore a simple transformation is needed before calling nag_opt_handle_set_linconstr (e04rjc) as demonstrated in the example program.

The data file stores the following problem:

$$\text{minimize} \quad c^{\mathrm{T}}x + \tfrac{1}{2}x^{\mathrm{T}}Hx \quad \text{subject to} \quad \begin{array}{rcl} l_B & \leq Bx & \leq u_B, \\ -2 & \leq x & \leq 2, \end{array}$$

where

$$c = \begin{pmatrix} -4.0 \\ -1.0 \\ -1.0 \\ -1.0 \\ -1.0 \\ -1.0 \\ -1.0 \\ -0.1 \\ -0.3 \end{pmatrix}, \quad H = \begin{pmatrix} 2 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 2 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 2 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 2 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$B = \begin{pmatrix} 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 4.0 \\ 1.0 & 2.0 & 3.0 & 4.0 & -2.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 1.0 & -1.0 & 1.0 & -1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \end{pmatrix},$$

$$l_B = \begin{pmatrix} -2.0 \\ -2.0 \\ -2.0 \end{pmatrix} \quad \text{and} \quad u_B = \begin{pmatrix} 1.5 \\ 1.5 \\ 4.0 \end{pmatrix}.$$

The optimal solution (to five figures) is

$$x^* = (2.0, -0.23333, -0.26667, -0.3, -0.1, 2.0, 2.0, -1.7777, -0.45555)^{\mathrm{T}}.$$

See also Section 10 in nag_opt_handle_init (e04rac) for links to further examples in this suite.

10.1 Program Text

```
/* nag_opt_handle_set_linconstr (e04rjc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

/* Read in LP/QP problem stored in a MPS file, formulated it
 * as a handle and pass it to the solver.
 */
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nage04.h>
#include <nagx04.h>

/* Make a typedef for convenience when allocating crname. */
typedef char e04mx_name[9];

int main(int argc, char *argv[])
{
  char fname_default[] = "e04rjce.opt";
  Integer mpslst = 1;

  Integer exit_status = 0;
  Integer idlc, idx, idx_dest, inform, iobj, j, lintvar, m, maxlintvar,
         maxm, maxn, maxncolh, maxnnz, maxnnzh, minmax, n, ncolh, nname,
         nnz, nnzc, nnzh, nnzu, nnzua, nnzuc;
  char *fname = 0;
  char pnames[5][9] = { "", "", "", "", "" };
  double rinfo[32], stats[32];
  double *a = 0, *bl = 0, *bu = 0, *c = 0, *h = 0, *x = 0;
  Integer *iccola = 0, *iccolh = 0, *icola = 0, *icolh = 0, *idxc = 0,
         *irowa = 0, *irowh = 0;
  char (*crname)[9] = 0;
  void *handle = 0;
  /* Nag Types */
  Nag_FileID fileid;
```

```
  NagError fail;

  printf("nag_opt_handle_set_linconstr (e04rjc) Example Program Results\n\n");

  /* Use the first command line argument as the filename or
   * choose default filename stored in 'fname_default'. */
  if (argc > 1)
    fname = argv[1];
  else
    fname = fname_default;
  printf("Reading MPS file: %s\n", fname);
  fflush(stdout);

  /* nag_open_file (x04acc).
   * Open unit number for reading and associate unit with named file. */
  nag_open_file(fname, 0, &fileid, NAGERR_DEFAULT);

  /* nag_opt_miqp_mps_read (e04mxc).
   * Reads MPS data file defining LP or QP problem.
   * Query call to estimate the size of the problem. */
  nag_opt_miqp_mps_read(fileid, 0, 0, 0, 0, 0, 0, mpslst, &n, &m, &nnz,
                        &ncolh, &nnzh, &lintvar, NULL, NULL, NULL, NULL, NULL,
                        NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
                        NAGERR_DEFAULT);

  /* nag_close_file (x04adc).
   * Close file associated with given unit number. */
  nag_close_file(fileid, NAGERR_DEFAULT);

  maxm = m;
  maxn = n;
  maxnnz = nnz;
  maxnnzh = nnzh;
  maxncolh = ncolh;
  maxlintvar = -1;

  if (!(irowa = NAG_ALLOC(maxnnz, Integer)) ||
      !(iccola = NAG_ALLOC(maxn + 1, Integer)) ||
      !(a = NAG_ALLOC(maxnnz, double)) ||
      !(bl = NAG_ALLOC(maxn + maxm, double)) ||
      !(bu = NAG_ALLOC(maxn + maxm, double)) ||
      !(irowh = NAG_ALLOC(maxnnzh, Integer)) ||
      !(iccolh = NAG_ALLOC(maxncolh + 1, Integer)) ||
      !(h = NAG_ALLOC(maxnnzh, double)) ||
      !(crname = NAG_ALLOC(maxn + maxm, e04mx_name)) ||
      !(x = NAG_ALLOC(maxn, double)) ||
      !(icolh = NAG_ALLOC(maxnnzh, Integer)) ||
      !(icola = NAG_ALLOC(maxnnz, Integer)))
  {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }

  /* Reopen the same file. */
  nag_open_file(fname, 0, &fileid, NAGERR_DEFAULT);

  /* Full call to the reader. */
  nag_opt_miqp_mps_read(fileid, maxn, maxm, maxnnz, maxncolh, maxnnzh,
                        maxlintvar, mpslst, &n, &m, &nnz, &ncolh, &nnzh,
                        &lintvar, &iobj, a, irowa, iccola, bl, bu, pnames,
                        &nname, crname, h, irowh, iccolh, &minmax, NULL,
                        NAGERR_DEFAULT);

  nag_close_file(fileid, NAGERR_DEFAULT);

  printf("MPS/QPS file read.\n");
  fflush(stdout);

  /* Data has been read. Set up the problem to the solver. */
```

```
  /* nag_opt_handle_init (e04rac).
   * Initialize an empty problem handle with n variables. */
  nag_opt_handle_init(&handle, n, NAGERR_DEFAULT);

  /* iccola[] was returned as 1-based, i.e., iccola[0]=1.
   * Change it to 0-based to simplify C operations. */
  for (j = 0; j <= n; j++)
    iccola[j]--;

  /* Move the linear objective from a to c. */
  if (iobj > 0) {
    /* Shift bounds. */
    for (j = n + iobj - 1; j < n + m - 1; j++) {
      bl[j] = bl[j + 1];
      bu[j] = bu[j + 1];
    }
    m--;

    /* Count how many nonzeros will be in c. */
    nnzc = 0;
    for (idx = 0; idx < nnz; idx++)
      if (irowa[idx] == iobj)
        nnzc++;

    if (!(idxc = NAG_ALLOC(nnzc, Integer)) || !(c = NAG_ALLOC(nnzc, double)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

    /* Extract row iobj form column compressed matrix a. */
    idx = 0;
    idx_dest = 0;
    nnzc = 0;
    for (j = 0; j < n; j++) {
      for (; idx < iccola[j + 1]; idx++) {
        if (irowa[idx] < iobj) {
          a[idx_dest] = a[idx];
          irowa[idx_dest] = irowa[idx];
          idx_dest++;
        }
        else if (irowa[idx] == iobj) {
          idxc[nnzc] = j + 1;
          c[nnzc] = a[idx];
          nnzc++;
        }
        else {
          a[idx_dest] = a[idx];
          irowa[idx_dest] = irowa[idx] - 1;
          idx_dest++;
        }
      }
      iccola[j + 1] = idx_dest;
    }
    nnz = idx_dest;
  }
  else
    /* There is no linear part of the objective function. */
    nnzc = 0;

  /* Convert (decompress) iccola[] to icola[]. */
  for (j = 0; j < n; j++)
    for (idx = iccola[j]; idx < iccola[j + 1]; idx++)
      icola[idx] = j + 1;

  /* Add objective function to the problem formulation. */
  if (nnzh == 0)
    /* nag_opt_handle_set_quadobj (e04rfc).
     * Define the objective as a (sparse) linear function (no nonzeros in H).*/
    nag_opt_handle_set_quadobj(handle, nnzc, idxc, c, 0, NULL, NULL, NULL,
```

```
                                NAGERR_DEFAULT);
  else {
    /* The objective is a quadratic function.
     * Firstly, transform (decompress) iccolh[] to icolh[], both are 1-based.
     * Secondly, e04mxc() returned a lower triangle but here
     * the upper triangle is needed ==> swap rows and columns. */
    for (j = 0; j < ncolh; j++)
      for (idx = iccolh[j]; idx < iccolh[j + 1]; idx++)
        icolh[idx - 1] = j + 1;
    /* nag_opt_handle_set_quadobj (e04rfc).
     * Add the quadratic objective to the handle.*/
    nag_opt_handle_set_quadobj(handle, nnzc, idxc, c, nnzh, icolh, irowh, h,
                               NAGERR_DEFAULT);
  }

  /* nag_opt_handle_set_simplebounds (e04rhc).
   * Define bounds on the variables. */
  nag_opt_handle_set_simplebounds(handle, n, bl, bu, NAGERR_DEFAULT);

  idlc = 0;
  /* nag_opt_handle_set_linconstr (e04rjc).
   * Add a block of linear constraints to the problem formulation.
   * Bounds of the constraints are stored after bounds on the variables. */
  nag_opt_handle_set_linconstr(handle, m, bl + n, bu + n, nnz, irowa, icola,
                               a, &idlc, NAGERR_DEFAULT);

  printf("The problem was set-up\n");
  fflush(stdout);

  /* Set optional arguments of the solver by calling
   * nag_opt_handle_opt_set (e04zmc). */
  nag_opt_handle_opt_set(handle, "Print Options = No", NAGERR_DEFAULT);

  /* Set up a starting point and call the solver,
   * ignore Lagrangian multipliers U/UA. */
  for (j = 0; j < n; j++)
    x[j] = 0.0;
  nnzu = 0;
  nnzuc = 0;
  nnzua = 0;

  /* Call the solver nag_opt_handle_solve_pennon (e04svc). */
  INIT_FAIL(fail);
  nag_opt_handle_solve_pennon(handle, n, x, nnzu, NULL, nnzuc, NULL, nnzua,
                              NULL, rinfo, stats, &inform, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_opt_handle_solve_pennon (e04svc).\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
  }

  printf("\nOptimal solution:\n");
  for (j = 0; j < n; j++)
    printf("  %f\n", x[j]);
  fflush(stdout);

END:

  /* nag_opt_handle_free (e04rzc).
   * Destroy the problem handle and deallocate all the memory. */
  if (handle)
    nag_opt_handle_free(&handle, NAGERR_DEFAULT);

  NAG_FREE(a);
  NAG_FREE(bl);
  NAG_FREE(bu);
  NAG_FREE(idxc);
  NAG_FREE(c);
  NAG_FREE(h);
  NAG_FREE(x);
```

```
  NAG_FREE(iccola);
  NAG_FREE(iccolh);
  NAG_FREE(icola);
  NAG_FREE(icolh);
  NAG_FREE(irowa);
  NAG_FREE(irowh);
  NAG_FREE(crname);
  return exit_status;
}
```

10.2 Program Data

```
NAME          E04RJ.EX
ROWS
 L  ..ROW1..
 L  ..ROW2..
 L  ..ROW3..
 N  ..COST..
COLUMNS
    ...X1...  ..ROW1..      1.0        ..ROW2..         1.0
    ...X1...  ..ROW3..      1.0        ..COST..        -4.0
    ...X2...  ..ROW1..      1.0        ..ROW2..         2.0
    ...X2...  ..ROW3..     -1.0        ..COST..        -1.0
    ...X3...  ..ROW1..      1.0        ..ROW2..         3.0
    ...X3...  ..ROW3..      1.0        ..COST..        -1.0
    ...X4...  ..ROW1..      1.0        ..ROW2..         4.0
    ...X4...  ..ROW3..     -1.0        ..COST..        -1.0
    ...X5...  ..ROW1..      1.0        ..ROW2..        -2.0
    ...X5...  ..ROW3..      1.0        ..COST..        -1.0
    ...X6...  ..ROW1..      1.0        ..ROW2..         1.0
    ...X6...  ..ROW3..      1.0        ..COST..        -1.0
    ...X7...  ..ROW1..      1.0        ..ROW2..         1.0
    ...X7...  ..ROW3..      1.0        ..COST..        -1.0
    ...X8...  ..ROW1..      1.0        ..ROW2..         1.0
    ...X8...  ..ROW3..      1.0        ..COST..        -0.1
    ...X9...  ..ROW1..      4.0        ..ROW2..         1.0
    ...X9...  ..ROW3..      1.0        ..COST..        -0.3
RHS
    RHS1      ..ROW1..      1.5
    RHS1      ..ROW2..      1.5
    RHS1      ..ROW3..      4.0
    RHS1      ..COST..      1000.0
RANGES
    RANGE1    ..ROW1..      3.5
    RANGE1    ..ROW2..      3.5
    RANGE1    ..ROW3..      6.0
BOUNDS
 LO BOUND     ...X1...     -2.0
 LO BOUND     ...X2...     -2.0
 LO BOUND     ...X3...     -2.0
 LO BOUND     ...X4...     -2.0
 LO BOUND     ...X5...     -2.0
 LO BOUND     ...X6...     -2.0
 LO BOUND     ...X7...     -2.0
 LO BOUND     ...X8...     -2.0
 LO BOUND     ...X9...     -2.0
 UP BOUND     ...X1...      2.0
 UP BOUND     ...X2...      2.0
 UP BOUND     ...X3...      2.0
 UP BOUND     ...X4...      2.0
 UP BOUND     ...X5...      2.0
 UP BOUND     ...X6...      2.0
 UP BOUND     ...X7...      2.0
 UP BOUND     ...X8...      2.0
 UP BOUND     ...X9...      2.0
QUADOBJ
    ...X1...  ...X1...  2.00000000E0   ...X2...  1.00000000E0
    ...X1...  ...X3...  1.00000000E0   ...X4...  1.00000000E0
    ...X1...  ...X5...  1.00000000E0
    ...X2...  ...X2...  2.00000000E0   ...X3...  1.00000000E0
```

```
    ...X2...  ...X4...  1.00000000E0   ...X5...  1.00000000E0
    ...X3...  ...X3...  2.00000000E0   ...X4...  1.00000000E0
    ...X3...  ...X5...  1.00000000E0
    ...X4...  ...X4...  2.00000000E0   ...X5...  1.00000000E0
    ...X5...  ...X5...  2.00000000E0
ENDATA
```

10.3 Program Results

```
nag_opt_handle_set_linconstr (e04rjc) Example Program Results

Reading MPS file: e04rjce.opt

 MPSX INPUT LISTING
 ------------------
 Searching for indicator line
 Line          1: Found NAME indicator line
                  Query mode - Ignoring NAME data.
 Line          2: Found ROWS indicator line
                  Query mode - Counting ROWS data.
 Line          7: Found COLUMNS indicator line
                  Query mode - Counting COLUMNS data.
 Line         26: Found RHS indicator line
                  Query mode - Ignoring RHS data.
 Line         31: Found RANGES indicator line
                  Query mode - Ignoring RANGES data.
 Line         35: Found BOUNDS indicator line
                  Query mode - Counting BOUNDS data.
 Line         54: Found QUADOBJ indicator line
                  Query mode - Counting QUADOBJ data.
                  Query mode - End of QUADOBJ data. Exit

 MPSX INPUT LISTING
 ------------------
 Searching for indicator line
 Line          1: Found NAME indicator line
 Line          2: Found ROWS indicator line
 Line          7: Found COLUMNS indicator line
 Line         26: Found RHS indicator line
 Line         31: Found RANGES indicator line
 Line         35: Found BOUNDS indicator line
 Line         54: Found QUADOBJ indicator line
 Line         64: Found ENDATA indicator line
MPS/QPS file read.
The problem was set-up
 E04SV, NLP-SDP Solver (Pennon)
 ------------------------------
 Number of variables             9                 [eliminated            0]
                          simple  linear  nonlin
 (Standard) inequalities      18       6       0
 (Standard) equalities                 0       0
 Matrix inequalities                   0       0 [dense    0, sparse    0]
                                                 [max dimension         0]

 ------------------------------------------------------------------
  it|  objective |  optim  |   feas  |  compl  | pen min |inner
 ------------------------------------------------------------------
   0  0.00000E+00  4.70E+00  0.00E+00  1.00E+01  1.00E+00   0
   1 -6.76762E+00  9.46E-04  0.00E+00  6.25E-01  1.00E+00   4
   2 -7.82467E+00  1.79E-03  3.69E-02  1.45E-01  4.65E-01   4
   3 -8.02059E+00  7.27E-03  2.27E-02  3.38E-02  2.16E-01   4
   4 -8.06187E+00  3.18E-03  5.75E-03  7.86E-03  1.01E-01   4
   5 -8.06653E+00  1.30E-03  1.13E-03  1.83E-03  4.68E-02   5
   6 -8.06739E+00  6.98E-03  1.42E-04  4.25E-04  2.18E-02   3
   7 -8.06775E+00  2.16E-04  2.80E-05  9.89E-05  1.01E-02   2
   8 -8.06778E+00  4.44E-05  6.94E-05  2.30E-05  4.71E-03   1
   9 -8.06778E+00  1.88E-06  1.15E-05  5.35E-06  2.19E-03   1
  10 -8.06778E+00  4.38E-08  1.52E-06  1.24E-06  1.02E-03   1
  11 -8.06778E+00  6.52E-10  1.74E-07  2.90E-07  4.74E-04   1
  12 -8.06778E+00  8.12E-12  1.90E-08  6.73E-08  2.21E-04   1
```

```
 ------------------------------------------------------------
 Status: converged, an optimal solution found
 ------------------------------------------------------------
 Final objective value               -8.067778E+00
 Relative precision                   1.518278E-09
 Optimality                           8.116995E-12
 Feasibility                          1.900689E-08
 Complementarity                      6.734260E-08
 Iteration counts
   Outer iterations                            12
   Inner iterations                            31
   Linesearch steps                            43
 Evaluation counts
   Augm. Lagr. values                          56
   Augm. Lagr. gradient                        44
   Augm. Lagr. hessian                         31
 ------------------------------------------------------------

Optimal solution:
  2.000000
  -0.233332
  -0.266666
  -0.299997
  -0.100004
  2.000000
  2.000000
  -1.777812
  -0.455547
```