NAG Library Function Document

nag_opt_handle_print (e04ryc)

1 Purpose

nag_opt_handle_print (e04ryc) is a part of the NAG optimization modelling suite. It allows you to print information about the problem, stored as a handle, such as which parts have already been defined or details of the matrix constraints.

2 Specification

```
#include <nag.h>
#include <nage04.h>
void nag_opt_handle_print (void *handle, Nag_FileID fileid,
     const char *cmdstr, NagError *fail)
```

3 Description

nag_opt_handle_print (e04ryc) prints information on a problem handle which has been previously initialized by nag_opt_handle_init (e04rac). Various pieces of information can be retrieved and printed to the given output unit. This can be helpful when the function is interfaced from interactive environments, for debugging purposes or to help familiarize you with the NAG optimization modelling suite.

The printer is guided by a command string which contains one or more of the following keywords:

Overview
: Gives a brief overview of the problem handle, particularly, which phase it is in, if the problem or optional parameters can be edited and which parts of the problem have already been set. This might be helpful to clarify situations when **fail.code** = NE_ALREADY_DEFINED or NE_PHASE is obtained from functions, such as, nag_opt_handle_set_linobj (e04rec), nag_opt_handle_set_quadobj (e04rfc), nag_opt_handle_set_simplebounds (e04rhc) and na g_opt_handle_set_linconstr (e04rjc).

Objective
: Prints the objective function as it was defined by nag_opt_handle_set_linobj (e04rec) or nag_opt_handle_set_quadobj (e04rfc) if it is linear or quadratic. Prints the sparsity structure of the objective function as it was defined by nag_opt_handle_set_nlnobj (e04rgc) if it is nonlinear.

Simple bounds
: Prints the variable bounds as defined by nag_opt_handle_set_simplebounds (e04rhc). This might help you understand the effect of the optional parameter **Infinite Bound Size** on the bounds.

Linear constraints bounds
Linear constraints detailed
: Print bounds or linear constraint matrix as defined by nag_opt_handle_set_linconstr (e04rjc).

Matrix constraints
: Gives a list of the matrix constraints as defined by nag_opt_handle_set_linmatineq (e04rnc) and nag_opt_handle_set_quadmatineq (e04rpc). For each matrix constraint its **idblk**, dimension and order (e.g., linear, bilinear) are printed.

Matrix constraints detailed
: Prints all the matrix constraints including all nonzeros of all the matrices as formulated by nag_opt_handle_set_linmatineq (e04rnc) and nag_opt_handle_set_quadmatineq (e04rpc).

Nonlinear constraints bounds
Nonlinear constraints detailed
Print bounds or sparsity structure of the nonlinear constraints as defined by nag_opt_handle_set_nlnconstr (e04rkc).

Multipliers sizes
Prints the expected dimensions of array arguments **u** and **ua** of the solver nag_opt_handle_solve_pennon (e04svc) which store the Lagrangian multipliers for standard and matrix constraints, respectively. This might be helpful in particular in connection with **Overview** and **Matrix constraints** to check the way the sizes of the arrays are derived.

Options
Prints all the current optional parameters. It flags whether the argument is at its default choice, whether you have set it or whether it is chosen by the solver (for example, options left on 'AUTO' setting after the solver has been called).

Note that the output data might not match your input exactly. The sparse matrices are typically transposed, sorted and explicit zeros removed and in certain cases transformed as needed (for example, matrices Q_{ij} and Q_{ji} are merged by nag_opt_handle_set_quadmatineq (e04rpc)).

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: **fileid** – Nag_FileID *Input*

On entry: the identifier associated with the file (or standard output) to be written to, as returned by a call of nag_open_file (x04acc).

Constraint: **fileid** ≥ 0.

3: **cmdstr** – const char * *Input*

On entry: a command string which contains one or more keywords which identify the piece of information about the handle to be printed. Keywords are case-insensitive and space tolerant. Multiple keywords in **cmdstr** must be separated by commas or semicolons.

Constraint: **cmdstr** can only contain one or more of the following accepted keywords: overview, objective, simple bounds, linear constraints bounds, linear constraints detailed, matrix constraints, matrix constraints detailed, nonlinear constraints bounds, nonlinear constraints detailed, multipliers sizes, options.

4: **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_FILEID

On entry, **fileid** = ⟨*value*⟩.
Constraint: **fileid** ≥ 0.

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_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_STR_UNKNOWN

cmdstr does not contain any keywords or is empty.

Keyword number ⟨*value*⟩ is not recognized.

Keyword number ⟨*value*⟩ is not recognized, it is too long.

NE_WRITE_ERROR

An error occurred when writing to output.

7 Accuracy

Not applicable.

8 Parallelism and Performance

nag_opt_handle_print (e04ryc) is not threaded in any implementation.

9 Further Comments

None.

10 Example

This example shows the life-cycle of a handle of a typical (BMI-SDP) problem by printing the overview of the handle in various stages of the problem formulation and after the solution is found. It is also helpful to notice how a linear matrix inequality is extended with the bilinear term, see nag_opt_handle_init (e04rac) and nag_opt_handle_set_quadmatineq (e04rpc) for further details.

The problem is as follows:

$$\begin{array}{ll} \underset{x,y \in \mathbb{R}}{\text{minimize}} & y \\ \text{subject to} & \begin{pmatrix} 1 & x-1 & y \\ x-1 & 3/4 & 0 \\ y & 0 & 16 \end{pmatrix} \succeq 0 \\ & \begin{pmatrix} x & -xy \\ -xy & 1 \end{pmatrix} \succeq 0 \\ & x \geq 0 \\ & -3 \leq y \leq 3 \end{array}$$

The solution is $x = 1/4$, $y = -2$.

Note that the matrix constraints need to be supplied in the form of equation (3) in nag_opt_handle_set_quadmatineq (e04rpc), i.e.,

$$\sum_{i,j=1}^{n} x_i x_j Q_{ij}^k + \sum_{i=1}^{n} x_i A_i^k - A_0^k \succeq 0, \quad k = 1, \ldots, m_A.$$

Therefore the first constraint is defined by matrices

$$A_0^1 = \begin{pmatrix} -1 & 1 & 0 \\ & -3/4 & 0 \\ & & -16 \end{pmatrix}, \quad A_1^1 = \begin{pmatrix} 0 & 1 & 0 \\ & 0 & 0 \\ & & 0 \end{pmatrix}, \quad A_2^1 = \begin{pmatrix} 0 & 0 & 1 \\ & 0 & 0 \\ & & 0 \end{pmatrix}$$

and the second one by

$$A_0^2 = \begin{pmatrix} 0 & 0 \\ & -1 \end{pmatrix}, \quad A_1^2 = \begin{pmatrix} 1 & 0 \\ & 0 \end{pmatrix}, \quad A_2^2 \text{ empty}, \quad Q_{12}^2 = \begin{pmatrix} 0 & -1 \\ 0 & 0 \end{pmatrix}.$$

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

10.1 Program Text

```
/* nag_opt_handle_print (e04ryc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

/* Demonstrate the life-cycle of a handle of a typical BMI-SDP problem
 * by printing the evolution of the HANDLE in certain stages to
 * the standard output.
 */

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

int main(void)
{
  Integer exit_status = 0;
  Integer dima, idblk, inform, nblk, nnzasum, nvar;
  double a[6], bl[2], bu[2], cvec[2], rinfo[32], stats[32], x[2];
  Integer icola[6], irowa[6], nnza[3], qi[1], qj[1];
  void *h = 0;
  /* Nag Types */
  Nag_FileID fileid;
  NagError fail;

  printf("nag_opt_handle_print (e04ryc) Example Program Results\n\n");
  fflush(stdout);
```

```
  /* Get Nag_FileID associated with the standard output by
   * nag_open_file (x04aac). */
  nag_open_file(NULL, 1, &fileid, NAGERR_DEFAULT);

  /* nag_opt_handle_init (e04rac).
   * Initialize an empty problem handle with 2 variables. */
  nvar = 2;
  nag_opt_handle_init(&h, nvar, NAGERR_DEFAULT);

  /* Anything can be defined at this phase. */
  printf("Freshly created handle\n");
  fflush(stdout);
  /* nag_opt_handle_print (e04ryc).
   * Print information currently stored in the handle. */
  nag_opt_handle_print(h, fileid, "Overview", NAGERR_DEFAULT);

  /* nag_opt_handle_set_linobj (e04rec).
   * Define linear objective (min y). */
  cvec[0] = 0.0;
  cvec[1] = 1.0;
  nag_opt_handle_set_linobj(h, nvar, cvec, NAGERR_DEFAULT);

  /* nag_opt_handle_set_simplebounds (e04rhc).
   * Add simple bounds (x>=0, -3<=y<=3) to the problem formulation. */
  bl[0] = 0.0;
  bu[0] = 1.0e20;
  bl[1] = -3.0;
  bu[1] = 3.0;
  nag_opt_handle_set_simplebounds(h, nvar, bl, bu, NAGERR_DEFAULT);

  /* The simple bounds and the objective are set and cannot be changed. */
  printf("\nHandle after definition of simple bounds and the objective\n");
  fflush(stdout);
  nag_opt_handle_print(h, fileid, "Overview,Objective,Simple Bounds",
                       NAGERR_DEFAULT);

  /* Definition of the first (linear) matrix constraint
   *    ( 1   x-1   y )
   *    (x-1  3/4   0 )  >= 0
   *    ( y    0   16 )
   * only upper triangles, thus we have matrices
   *       ( 1  -1  0 )        ( 0  1  0 )       ( 0  0  1 )
   * A0 = -(   3/4  0 ),  A1 = (    0  0 ), A2 = (    0  0 )
   *       (       16 )        (       0 )       (       0 )
   * Note: don't forget the minus at A0 term! */
  dima = 3;
  nnzasum = 6;
  nblk = 1;
  idblk = 0;
  /* A0 */
  irowa[0] = 1;
  icola[0] = 1;
  a[0] = -1.0;
  irowa[1] = 1;
  icola[1] = 2;
  a[1] = 1.0;
  irowa[2] = 2;
  icola[2] = 2;
  a[2] = -0.75;
  irowa[3] = 3;
  icola[3] = 3;
  a[3] = -16.0;
  nnza[0] = 4;
  /* A1 */
  irowa[4] = 1;
  icola[4] = 2;
  a[4] = 1.0;
  nnza[1] = 1;
  /* A2 */
  irowa[5] = 1;
```

```
  icola[5] = 3;
  a[5] = 1.0;
  nnza[2] = 1;
  /* nag_opt_handle_set_linmatineq (e04rnc).
   * Add a linear matrix inequality constraint to the problem formulation. */
  nag_opt_handle_set_linmatineq(h, nvar, dima, nnza, nnzasum, irowa, icola,
                                a, nblk, NULL, &idblk, NAGERR_DEFAULT);

  /* It is possible to add or extend existing matrix constraints. */
  printf("\nHandle after definition of the 1st matrix constraint\n");
  fflush(stdout);
  nag_opt_handle_print(h, fileid, "Overview,Matrix Constraints",
                       NAGERR_DEFAULT);

  /* Definition of the absolute term and linear part of BMI
   *    (  x   -xy )
   *    (-xy    1  ) >= 0
   * thus
   *       ( 0  0 )        ( 1  0 )
   * A0 = -(    1 ),  A1 = (    0 ), A2 = zero
   * Note: don't forget the minus at A0 term! */
  dima = 2;
  nnzasum = 2;
  nblk = 1;
  idblk = 0;
  /* A0 */
  irowa[0] = 2;
  icola[0] = 2;
  a[0] = -1.0;
  nnza[0] = 1;
  /* A1 */
  irowa[1] = 1;
  icola[1] = 1;
  a[1] = 1.0;
  nnza[1] = 1;
  /* A2 */
  nnza[2] = 0;
  /* nag_opt_handle_set_linmatineq (e04rnc).
   * Add another linear matrix inequality which will be extended to BMI. */
  nag_opt_handle_set_linmatineq(h, nvar, dima, nnza, nnzasum, irowa, icola,
                                a, nblk, NULL, &idblk, NAGERR_DEFAULT);

  /* It is possible to add or extend existing matrix constraints. */
  printf("\nHandle after partial definition of the 2nd matrix constraint\n");
  fflush(stdout);
  nag_opt_handle_print(h, fileid, "Matrix Constraints", NAGERR_DEFAULT);

  /* Extending current matrix constraint (with IDBLK) by bilinear term
   *       ( 0  -1 )
   * Q12 = ( 0   0 ). */
  dima = 2;
  nnzasum = 1;
  nnza[0] = 1;
  irowa[0] = 1;
  icola[0] = 2;
  a[0] = -1.0;
  qi[0] = 1;
  qj[0] = 2;
  /* nag_opt_handle_set_quadmatineq (e04rpc).
   * Add a bilinear matrix term. */
  nag_opt_handle_set_quadmatineq(h, 1, qi, qj, dima, nnza, nnzasum, irowa,
                                 icola, a, &idblk, NAGERR_DEFAULT);

  /* The problem is completely defined. */
  printf("\nHandle with the complete problem formulation\n");
  fflush(stdout);
  nag_opt_handle_print(h, fileid,
                       "Overview,Matrix Constraints,Multipliers Sizes",
                       NAGERR_DEFAULT);
  nag_opt_handle_print(h, fileid, "Matrix Constraints Detailed",
                       NAGERR_DEFAULT);
```

```
  /* Set optional arguments of the solver by calling
   * nag_opt_handle_opt_set (e04zmc). */
  nag_opt_handle_opt_set(h, "Print Options = No", NAGERR_DEFAULT);
  nag_opt_handle_opt_set(h, "Initial X = Automatic", NAGERR_DEFAULT);

  /* Options can be printed even outside the solver. */
  printf("\n");
  fflush(stdout);
  nag_opt_handle_print(h, fileid, "Options", NAGERR_DEFAULT);

  /* Pass the handle to the solver
   * nag_opt_handle_solve_pennon (e04svc). */
  INIT_FAIL(fail);
  nag_opt_handle_solve_pennon(h, nvar, x, 0, NULL, 0, NULL, 0, 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;
  }

  /* After the solver finished. */
  printf("\nProblem solved\n");
  fflush(stdout);
  nag_opt_handle_print(h, fileid, "Overview", NAGERR_DEFAULT);

  /* Print the result. */
  printf("\nFinal objective function = %f\n", rinfo[0]);
  printf("Final x = [%f, %f].\n", x[0], x[1]);

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

END:
  return exit_status;
}
```

10.2 Program Data

None.

10.3 Program Results

```
nag_opt_handle_print (e04ryc) Example Program Results

Freshly created handle
 Overview
   Status:                  Problem and option settings are editable.
   No of variables:         2
   Objective function:      not defined yet
   Simple bounds:           not defined yet
   Linear constraints:      not defined yet
   Nonlinear constraints:   not defined yet
   Matrix constraints:      not defined yet

Handle after definition of simple bounds and the objective
 Overview
   Status:                  Problem and option settings are editable.
   No of variables:         2
   Objective function:      linear
   Simple bounds:           defined
   Linear constraints:      not defined yet
   Nonlinear constraints:   not defined yet
   Matrix constraints:      not defined yet
 Objective function
   linear part
   c(      2) =  1.00E+00,
```

```
 Simple bounds
    0.000E+00 <= X_      1
   -3.000E+00 <= X_      2 <=  3.000E+00

Handle after definition of the 1st matrix constraint
 Overview
   Status:                  Problem and option settings are editable.
   No of variables:         2
   Objective function:      linear
   Simple bounds:           defined
   Linear constraints:      not defined yet
   Nonlinear constraints:   not defined yet
   Matrix constraints:      1
 Matrix constraints
   IDblk =     1, size =      3 x     3, linear

Handle after partial definition of the 2nd matrix constraint
 Matrix constraints
   IDblk =     1, size =      3 x     3, linear
   IDblk =     2, size =      2 x     2, linear

Handle with the complete problem formulation
 Overview
   Status:                  Problem and option settings are editable.
   No of variables:         2
   Objective function:      linear
   Simple bounds:           defined
   Linear constraints:      not defined yet
   Nonlinear constraints:   not defined yet
   Matrix constraints:      2
 Matrix constraints
   IDblk =     1, size =      3 x     3, linear
   IDblk =     2, size =      2 x     2, polynomial of order 2
 Lagrangian multipliers sizes
   (Standard) multipliers U: 4 + 0 + 0
   Matrix multipliers UA:    9
 Matrix constraints (detailed)
   Matrix inequality IDBLK =     1, dimension     3
     multiindex k =     0
       A_k(     1,     1) = -1.000E+00
       A_k(     2,     1) =  1.000E+00
       A_k(     2,     2) = -7.500E-01
       A_k(     3,     3) = -1.600E+01

     multiindex k =     1
       A_k(     2,     1) =  1.000E+00

     multiindex k =     2
       A_k(     3,     1) =  1.000E+00

   Matrix inequality IDBLK =     2, dimension     2
     multiindex k =     0
       A_k(     2,     2) = -1.000E+00

     multiindex k =     1
       A_k(     1,     1) =  1.000E+00

     multiindex k =     1,     2
       Q_k(     2,     1) = -1.000E+00


 Option settings
 Begin of Options
     Outer Iteration Limit         =                100     * d
     Inner Iteration Limit         =                100     * d
     Infinite Bound Size           =         1.00000E+20     * d
     Initial X                     =           Automatic     * U
     Initial U                     =           Automatic     * d
     Initial P                     =           Automatic     * d
     Hessian Density               =                Auto     * d
     Init Value P                  =         1.00000E+00     * d
```

```
    Init Value Pmat               =         1.00000E+00     * d
    Presolve Block Detect         =                 Yes     * d
    Print File                    =                   6     * d
    Print Level                   =                   2     * d
    Print Options                 =                  No     * U
    Monitoring File               =                  -1     * d
    Monitoring Level              =                   4     * d
    Monitor Frequency             =                   0     * d
    Stats Time                    =                  No     * d
    P Min                         =         1.05367E-08     * d
    Pmat Min                      =         1.05367E-08     * d
    U Update Restriction          =         5.00000E-01     * d
    Umat Update Restriction       =         3.00000E-01     * d
    Preference                    =               Speed     * d
    Transform Constraints         =                Auto     * d
    Dimacs Measures               =               Check     * d
    Stop Criteria                 =                Soft     * d
    Stop Tolerance 1              =         1.00000E-06     * d
    Stop Tolerance 2              =         1.00000E-07     * d
    Stop Tolerance Feasibility    =         1.00000E-07     * d
    Linesearch Mode               =                Auto     * d
    Inner Stop Tolerance          =         1.00000E-02     * d
    Inner Stop Criteria           =           Heuristic     * d
    Task                          =            Minimize     * d
    P Update Speed                =                  12     * d
    Hessian Mode                  =                Auto     * d
    Verify Derivatives            =                Auto     * d
    Time Limit                    =         1.00000E+06     * d
End of Options
E04SV, NLP-SDP Solver (Pennon)
------------------------------
Number of variables            2                  [eliminated            0]
                          simple  linear  nonlin
(Standard) inequalities        3       0       0
(Standard) equalities                  0       0
Matrix inequalities                    1       1 [dense    2, sparse    0]
                                                 [max dimension         3]

------------------------------------------------------------
 it|  objective |  optim  |   feas  |  compl  | pen min |inner
------------------------------------------------------------
  0  0.00000E+00  4.56E+00  1.23E-01  4.41E+01  1.00E+00   0
  1 -3.01854E-01  1.21E-03  0.00E+00  1.89E+00  1.00E+00   7
  2 -6.21230E-01  2.58E-03  0.00E+00  6.72E-01  4.65E-01   2
  3 -2.11706E+00  4.31E-03  3.39E-02  6.07E-02  2.16E-01   5
  4 -2.01852E+00  5.71E-03  6.05E-03  8.55E-03  1.01E-01   3
  5 -2.00164E+00  3.36E-03  6.26E-04  1.02E-03  4.68E-02   2
  6 -2.00022E+00  4.45E-03  8.37E-05  1.82E-04  2.18E-02   1
  7 -2.00001E+00  4.73E-04  4.01E-06  3.96E-05  1.01E-02   1
  8 -2.00000E+00  4.77E-06  2.25E-07  9.20E-06  4.71E-03   1
  9 -2.00000E+00  4.52E-08  3.61E-08  2.14E-06  2.19E-03   1
 10 -2.00000E+00  6.63E-09  3.19E-08  4.98E-07  1.02E-03   1
 11 -2.00000E+00  8.80E-10  5.34E-09  1.16E-07  4.74E-04   1
 12 -2.00000E+00  1.02E-10  5.41E-09  2.69E-08  2.21E-04   1
------------------------------------------------------------
Status: converged, an optimal solution found
------------------------------------------------------------
Final objective value              -2.000000E+00
Relative precision                  9.839057E-10
Optimality                          1.019125E-10
Feasibility                         5.406175E-09
Complementarity                     2.693704E-08
Iteration counts
  Outer iterations                            12
  Inner iterations                            26
  Linesearch steps                            37
Evaluation counts
  Augm. Lagr. values                          50
  Augm. Lagr. gradient                        39
  Augm. Lagr. hessian                         26
------------------------------------------------------------
```

```
Problem solved
 Overview
   Status:                  Solver finished, only options can be changed.
   No of variables:         2
   Objective function:      linear
   Simple bounds:           defined
   Linear constraints:      not defined
   Nonlinear constraints:   not defined
   Matrix constraints:      2

Final objective function = -2.000000
Final x = [0.250000, -2.000000].
```