NAG Library Function Document

nag_opt_handle_set_get_real (e04rxc)

1 Purpose

nag_opt_handle_set_get_real (e04rxc) is a part of the NAG optimization modelling suite. It allows you to read or write a piece of information to the problem stored in the handle. For example, it may be used to extract the current approximation of the solution during a monitoring step.

2 Specification

```
#include <nag.h>
#include <nage04.h>
```

```
void nag_opt_handle_set_get_real (void *handle, const char *cmdstr,
     Integer ioflag, Integer *lrarr, double rarr[], NagError *fail)
```

3 Description

nag_opt_handle_set_get_real (e04rxc) adds an additional means of communication to functions within the NAG optimization modelling suite. It allows you to either read or write a piece of information in the handle in the form of a real array. The item is identified by **cmdstr** and the direction of the communication is set by **ioflag**.

The following **cmdstr** are available:

Primal Variables or **X**
The current value of the primal variables.

Dual Variables or **U**
The current value of the dual variables (Lagrangian multipliers).

The functionality is limited in this release of the NAG C Library to the retrieval of the approximate solution within the monitoring step of **nag_opt_handle_solve_lp_ipm (e04mtc)** or its final solution.

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 between calls to the NAG optimization modelling suite.

2: **cmdstr** – const char * *Input*

On entry: a string which identifies the item within the handle to be read or written. The string is case insensitive and space tolerant.

Constraint: **cmdstr** = 'Primal Variables', 'Dual Variables', 'X' or 'U'.

3: **ioflag** – Integer *Input*

On entry: indicates the direction of the communication.

ioflag $\neq 0$
: **nag_opt_handle_set_get_real (e04rxc)** will extract the requested information from the handle to **rarr**.

ioflag $= 0$
: The writing mode will apply and the content of **rarr** will be copied to the handle.

4: **lrarr** – Integer * *Input/Output*

On entry: the dimension of the array **rarr**.

On exit: the correct expected dimension of **rarr** if **lrarr** does not match the item identified by **cmdstr** (in this case **nag_opt_handle_set_get_real (e04rxc)** returns **fail.code** = NE_DIM_MATCH).

5: **rarr**[**lrarr**] – double *Input/Output*

On entry: if **ioflag** $= 0$ (write mode), **rarr** must contain the information to be written to the handle; otherwise it does not need to be set.

On exit: if **ioflag** $\neq 0$ (read mode), **rarr** contains the information requested by **cmdstr**; otherwise **rarr** is unchanged.

6: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 3.7 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_DIM_MATCH

On entry, **lrarr** = ⟨*value*⟩, expected value = ⟨*value*⟩.
Constraint: **lrarr** must match the size of the data identified in **cmdstr**.

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.
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_PHASE

The request cannot be processed at this phase.

The requested information is not available.

NE_READ_ERROR

Reading mode is not supported for the given **cmdstr**.

NE_STATE

The request cannot be processed by the current solver.

NE_STR_UNKNOWN

The provided **cmdstr** is not recognised.

NE_WRITE_ERROR

Writing mode is not supported for the given **cmdstr**.

7 Accuracy

Not applicable.

8 Parallelism and Performance

nag_opt_handle_set_get_real (e04rxc) is not threaded in any implementation.

9 Further Comments

None.

10 Example

This example demonstrates how to use **nag_opt_handle_set_get_real (e04rxc)** to extract the current approximation of the solution when the monitoring function **monit** is called during the solve by **nag_opt_handle_solve_lp_ipm (e04mtc)**.

We solve the following linear programming problem:

$$-0.02x_1 - 0.2x_2 - 0.2x_3 - 0.2x_4 - 0.2x_5 + 0.04x_6 + 0.04x_7$$

subject to the bounds

$$\begin{array}{l} -0.01 \leq x_1 \leq 0.01 \\ -0.1 \quad \leq x_2 \leq 0.15 \\ -0.01 \leq x_3 \leq 0.03 \\ -0.04 \leq x_4 \leq 0.02 \\ -0.1 \quad \leq x_5 \leq 0.05 \\ -0.01 \leq x_6 \\ -0.01 \leq x_7 \end{array}$$

and the general constraints

$$\begin{array}{rcrcrcrcrcrcrcrcrr} & & x_1 & + & x_2 & + & x_3 & + & x_4 & + & x_5 & + & x_6 & + & x_7 & = & -0.13 \\ & & 0.15x_1 & + & 0.04x_2 & + & 0.02x_3 & + & 0.04x_4 & + & 0.02x_5 & + & 0.01x_6 & + & 0.03x_7 & \leq & -0.0049 \\ & & 0.03x_1 & + & 0.05x_2 & + & 0.08x_3 & + & 0.02x_4 & + & 0.06x_5 & + & 0.01x_6 & & & \leq & -0.0064 \\ & & 0.02x_1 & + & 0.04x_2 & + & 0.01x_3 & + & 0.02x_4 & + & 0.02x_5 & & & & & \leq & -0.0037 \\ & & 0.02x_1 & + & 0.03x_2 & & & & & + & 0.01x_5 & & & & & \leq & -0.0012 \\ -0.0992 & \leq & 0.70x_1 & + & 0.75x_2 & + & 0.80x_3 & + & 0.75x_4 & + & 0.80x_5 & + & 0.97x_6 & & & & \\ -0.003 & \leq & 0.02x_1 & + & 0.06x_2 & + & 0.08x_3 & + & 0.12x_4 & + & 0.02x_5 & + & 0.01x_6 & + & 0.97x_7 & \leq & 0.002 \end{array}$$

During the monitoring step of **nag_opt_handle_solve_lp_ipm (e04mtc)**, if the three convergence measures are below an acceptable threshold, the approximate solution is extracted with **nag_opt_handle_set_get_real (e04rxc)** and printed on the standard output.

10.1 Program Text

```
/* nag_opt_handle_set_get_real (e04rxc) Example Program.
 *
 * Copyright 2017 Numerical Algorithms Group.
 *
 * Mark 26.1, 2017.
 */

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

#ifdef __cplusplus
extern "C"
{
#endif
static void NAG_CALL monit(void *handle, const double rinfo[],
                           const double stats[], Nag_Comm *comm,
                           Integer *inform);
#ifdef __cplusplus
}
#endif

int main(void){

  Integer nclin, nvar, nnza, nnzc, nnzu, exit_status, i, idlc;
  Integer *irowa = 0, *icola = 0;
  Integer iuser[1];
  double *cvec = 0, *a = 0, *bla = 0, *bua = 0, *xl = 0, *xu = 0,
    *x = 0, *u = 0;
  double rinfo[100], stats[100];
  void *handle = 0;
  /* Nag Types */
  Nag_Comm comm;
  NagError fail;

  exit_status = 0;

  printf("nag_opt_handle_set_get_real (e04rxc) Example Program Results\n\n");
  fflush(stdout);

  /* Read the data file and allocate memory */
#ifdef _WIN32
  scanf_s(" %*[^\n]"); /* Skip heading in data file */
#else
  scanf(" %*[^\n]"); /* Skip heading in data file */
#endif
#ifdef _WIN32
  scanf_s("%"NAG_IFMT" %"NAG_IFMT" %"NAG_IFMT" %"NAG_IFMT" %*[^\n]",&nclin,&nvar,
&nnza,&nnzc);
#else
   scanf("%"NAG_IFMT" %"NAG_IFMT" %"NAG_IFMT" %"NAG_IFMT" %*[^\n]",&nclin,&nvar,
&nnza,&nnzc);
#endif
  /* Allocate memory */
  nnzu = 2*nvar + 2*nclin;
  if (!(irowa = NAG_ALLOC(nnza, Integer))  ||
      !(icola = NAG_ALLOC(nnza, Integer))  ||
      !(cvec = NAG_ALLOC(nnzc,double))     ||
      !(a = NAG_ALLOC(nnza,double))        ||
      !(bla = NAG_ALLOC(nclin,double))     ||
```

```
      !(bua = NAG_ALLOC(nclin,double))      ||
      !(xl = NAG_ALLOC(nvar,double))        ||
      !(xu = NAG_ALLOC(nvar,double))        ||
      !(x = NAG_ALLOC(nvar,double))         ||
      !(u = NAG_ALLOC(nnzu,double)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }
  for (i=0; i< nvar; i++){
    x[i] = 0.0;
  }

  /* Read objective */
  for (i=0; i<nnzc; i++){
#ifdef _WIN32
    scanf_s("%lf",&cvec[i]);
#else
    scanf("%lf",&cvec[i]);
#endif
  }
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif
  /* Read constraint matrix row indices */
  for (i=0; i<nnza; i++){
#ifdef _WIN32
    scanf_s("%"NAG_IFMT,&irowa[i]);
#else
    scanf("%"NAG_IFMT,&irowa[i]);
#endif
  }
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif
  /* Read constraint matrix col indices */
  for (i=0; i<nnza; i++){
#ifdef _WIN32
    scanf_s("%"NAG_IFMT,&icola[i]);
#else
    scanf("%"NAG_IFMT,&icola[i]);
#endif
  }
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif
  /* Read constraint matrix values */
  for (i=0; i<nnza; i++){
#ifdef _WIN32
    scanf_s("%lf",&a[i]);
#else
    scanf("%lf",&a[i]);
#endif
  }
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif
  /* Read linear constraints lower bounds */
  for (i=0; i<nclin; i++){
#ifdef _WIN32
    scanf_s("%lf ",&bla[i]);
#else
```

```
    scanf("%lf ",&bla[i]);
#endif
  }
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif
  /* Read linear constraints upper bounds */
  for (i=0; i<nclin; i++){
#ifdef _WIN32
    scanf_s("%lf ",&bua[i]);
#else
    scanf("%lf ",&bua[i]);
#endif
  }
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif
  /* Read variables lower bounds */
  for (i=0; i<nvar; i++){
#ifdef _WIN32
    scanf_s("%lf ",&xl[i]);
#else
    scanf("%lf ",&xl[i]);
#endif
  }
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif
  /* Read variables upper bounds */
  for (i=0; i<nvar; i++){
#ifdef _WIN32
    scanf_s("%lf ",&xu[i]);
#else
    scanf("%lf ",&xu[i]);
#endif
  }
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif

  /* Create the problem handle */
  /* nag_opt_handle_init (e04rac).
   * Initialize an empty problem handle with NVAR variables. */
  nag_opt_handle_init(&handle, nvar, NAGERR_DEFAULT);

  /* nag_opt_handle_set_linobj (e04rec)
   * Define a linear objective */
  nag_opt_handle_set_linobj(handle,nvar,cvec,NAGERR_DEFAULT);

  /* nag_opt_handle_set_simplebounds (e04rhc)
   * Define bounds on the variables */
  nag_opt_handle_set_simplebounds(handle,nvar,xl,xu,NAGERR_DEFAULT);

  /* nag_opt_handle_set_linconstr (e04rjc)
   * Define linear constraints */
  idlc = 0;
  nag_opt_handle_set_linconstr(handle,nclin,bla,bua,nnza,irowa,
                               icola,a,&idlc,NAGERR_DEFAULT);

  /* nag_opt_handle_opt_set (e04zmc)
   * Require printing of the solution at the end of the solve
   */
  nag_opt_handle_opt_set(handle, "Print Solution = Yes",
```

```
                       NAGERR_DEFAULT);
  /* Deactivate option printing */
  nag_opt_handle_opt_set(handle, "Print Options = No",
                       NAGERR_DEFAULT);
  /* Use a constant number of centrality correctors steps */
  nag_opt_handle_opt_set(handle, "LPIPM Centrality Correctors = -6",
                       NAGERR_DEFAULT);
  /* Turn on monitoring */
  nag_opt_handle_opt_set(handle, "LPIPM Monitor Frequency = 1",
                       NAGERR_DEFAULT);
  /* Print the solution at the end of the solve */
  nag_opt_handle_opt_set(handle, "Print Solution = X",
                       NAGERR_DEFAULT);
  comm.iuser = iuser;
  iuser[0] = nvar;

  INIT_FAIL(fail);
  /* nag_opt_handle_solve_lp_ipm (e04mtc)
   * Call LP interior point solver with the primal-dual algorithm */
  nag_opt_handle_solve_lp_ipm(handle,nvar,x,nnzu,u,rinfo,stats,monit,
                            &comm,&fail);

 END:
  NAG_FREE(cvec);
  NAG_FREE(irowa);
  NAG_FREE(icola);
  NAG_FREE(a);
  NAG_FREE(bla);
  NAG_FREE(bua);
  NAG_FREE(xl);
  NAG_FREE(xu);
  NAG_FREE(x);
  NAG_FREE(u);
  /* nag_opt_handle_free (e04rzc).
   * Destroy the problem handle and deallocate all the memory. */
  if (handle)
    nag_opt_handle_free(&handle, NAGERR_DEFAULT);

  return exit_status;
}


static void NAG_CALL monit(void *handle, const double rinfo[],
                           const double stats[], Nag_Comm *comm,
                           Integer *inform){
  /* Monitoring function */
  double tol = 1.0e-03;
  Integer nvar, i;
  double *x = 0;

  if (!comm || !comm->iuser){
    /* The communication structure is not correctly allocated, abort solve */
    *inform = -1;
    return;
  }

  nvar = comm->iuser[0];
  x = NAG_ALLOC(nvar,double); assert(x);

  /* x is close to the solution, extract the values with
   * nag_opt_handle_set_get_real (e04rxc) and print it
   */
  if (rinfo[4]<tol && rinfo[5]<tol &&rinfo[6]<tol){
    nag_opt_handle_set_get_real(handle,"Primal Variables",1,&nvar,x,
                              NAGERR_DEFAULT);
    printf("\n");
    printf("    monit() reports good approximate solution "
           "(tol =, %8.2e):\n",tol);
    for (i=0; i<nvar; i++){
      printf("      X%1"NAG_IFMT": %9.2e\n",i+1,x[i]);
    }
```

```
    printf("     end of monit()\n");
  }
  fflush(stdout);

  if (x)
    NAG_FREE(x);
}
```

10.2 Program Data

```
nag_opt_handle_set_get_real (e04rxc) Example Program Data
  7  7  41  7                                             : Problem dimensions
  -0.02  -0.20  -0.20  -0.20  -0.20   0.04   0.04  : Objective values
  1  1  1  1  1  1  1
  2  2  2  2  2  2  2
  3  3  3  3  3  3
  4  4  4  4  4
  5  5  5
  6  6  6  6  6  6
  7  7  7  7  7  7  7                                     : End of irowa
  1  2  3  4  5  6  7
  1  2  3  4  5  6  7
  1  2  3  4  5  6
  1  2  3  4  5
  1  2  5
  1  2  3  4  5  6
  1  2  3  4  5  6  7                                     : End of icola
  1.00   1.00   1.00   1.00   1.00   1.00   1.00
  0.15   0.04   0.02   0.04   0.02   0.01   0.03
  0.03   0.05   0.08   0.02   0.06   0.01
  0.02   0.04   0.01   0.02   0.02
  0.02   0.03   0.01
  0.70   0.75   0.80   0.75   0.80   0.97
  0.02   0.06   0.08   0.12   0.02   0.01   0.97   : End of a
 -0.13 -1.0e20 -1.0e20 -1.0e20 -1.0e20 -0.0992 -0.003  : bla
 -0.13 -0.0049 -0.0064 -0.0037 -0.0012  1.0e20  0.002  : bua
 -0.01 -0.1 -0.01 -0.04 -0.1 -0.01 -0.01               : xl
  0.01  0.15 0.03  0.02 0.05 1.0e20 1.0e20             : xu
```

10.3 Program Results

```
nag_opt_handle_set_get_real (e04rxc) Example Program Results


 ----------------------------------------------
  E04MT, Interior point method for LP problems
 ----------------------------------------------

 Original Problem Statistics

   Number of variables          7
   Number of constraints        7
   Free variables               0
   Number of nonzeros          41


 Presolved Problem Statistics

   Number of variables         13
   Number of constraints        7
   Free variables               0
   Number of nonzeros          47


 ------------------------------------------------------------------------------
  it|    pobj     |    dobj     |  optim  |  feas   |  compl  |   mu    | mcc | I
 ------------------------------------------------------------------------------
   0 -7.86591E-02  1.71637E-02  1.27E+00  1.06E+00  8.89E-02  1.5E-01
   1  5.74135E-03 -2.24369E-02  6.11E-16  1.75E-01  2.25E-02  2.8E-02   0
   2  1.96803E-02  1.37067E-02  5.06E-16  2.28E-02  2.91E-03  3.4E-03   0
```

```
   3  2.15232E-02  1.96162E-02  7.00E-15  9.24E-03  1.44E-03  1.7E-03   0
   4  2.30321E-02  2.28676E-02  1.15E-15  2.21E-03  2.97E-04  3.4E-04   0

     monit() reports good approximate solution (tol =, 1.00e-03):
       X1: -9.99e-03
       X2: -1.00e-01
       X3:  3.00e-02
       X4:  2.00e-02
       X5: -6.73e-02
       X6: -2.35e-03
       X7: -2.27e-04
     end of monit()
   5  2.35658E-02  2.35803E-02  1.32E-15  1.02E-04  8.41E-06  9.6E-06   0

     monit() reports good approximate solution (tol =, 1.00e-03):
       X1: -1.00e-02
       X2: -1.00e-01
       X3:  3.00e-02
       X4:  2.00e-02
       X5: -6.75e-02
       X6: -2.28e-03
       X7: -2.35e-04
     end of monit()
   6  2.35965E-02  2.35965E-02  1.64E-15  7.02E-08  6.35E-09  7.2E-09   0

     monit() reports good approximate solution (tol =, 1.00e-03):
       X1: -1.00e-02
       X2: -1.00e-01
       X3:  3.00e-02
       X4:  2.00e-02
       X5: -6.75e-02
       X6: -2.28e-03
       X7: -2.35e-04
     end of monit()
   7  2.35965E-02  2.35965E-02  1.35E-15  3.52E-11  3.18E-12  3.6E-12   0
 ------------------------------------------------------------------------------
 Status: converged, an optimal solution found
 ------------------------------------------------------------------------------
 Final primal objective value         2.359648E-02
 Final dual objective value           2.359648E-02
 Absolute primal infeasibility        4.168797E-15
 Relative primal infeasibility        1.350467E-15
 Absolute dual infeasibility          5.084353E-11
 Relative dual infeasibility          3.518607E-11
 Absolute complementarity gap         2.685778E-11
 Relative complementarity gap         3.175366E-12
 Iterations                                      7

 Primal variables:
   idx   Lower bound       Value      Upper bound
     1  -1.00000E-02   -1.00000E-02    1.00000E-02
     2  -1.00000E-01   -1.00000E-01    1.50000E-01
     3  -1.00000E-02    3.00000E-02    3.00000E-02
     4  -4.00000E-02    2.00000E-02    2.00000E-02
     5  -1.00000E-01   -6.74853E-02    5.00000E-02
     6  -1.00000E-02   -2.28013E-03         inf
     7  -1.00000E-02   -2.34528E-04         inf
```