NAG Library Function Document

nag_opt_handle_solve_lp_ipm (e04mtc)

Note: *this function uses* **optional parameters** *to define choices in the problem specification and in the details of the algorithm. If you wish to use* default *settings for all of the optional parameters, you need only read Sections 1 to 10 of this document. If, however, you wish to reset some or all of the settings please refer to Section 11 for a detailed description of the algorithm and to Section 12 for a detailed description of the specification of the optional parameters.*

1 Purpose

nag_opt_handle_solve_lp_ipm (e04mtc) is a solver from the NAG optimization modelling suite for large-scale linear programming (LP) problems based on an interior point method (IPM).

2 Specification

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

void nag_opt_handle_solve_lp_ipm (void *handle, Integer nvar, double x[],
     Integer nnzu, double u[], double rinfo[], double stats[],

     void (*monit)(void *handle, const double rinfo[], const double stats[],
          Nag_Comm *comm, Integer *inform),

     Nag_Comm *comm, NagError *fail)
```

3 Description

nag_opt_handle_solve_lp_ipm (e04mtc) solves a large-scale linear optimization problem in the following form

$$\begin{array}{lll} \underset{x\in R^n}{\text{minimize}} & c^{\mathrm{T}}x & \text{(a)} \\ \text{subject to} & l_A \leq Ax \leq u_A & \text{(b)} \\ & l_x \leq x \leq u_x, & \text{(c)} \end{array} \qquad (1)$$

where n is the number of decision variables and m is the number of linear constraints. Here c, x, l_x, u_x are n-dimensional vectors, A is an m by n sparse matrix and l_A, u_A are m-dimensional vectors.

nag_opt_handle_solve_lp_ipm (e04mtc) implements two algorithmic variants of the interior point method for solving linear optimization problems: the infeasible Primal-Dual interior point method and homogeneous Self-Dual interior point method. In general, the Self-Dual algorithm has a slightly higher price per iteration, however, it is able to declare infeasibility or unboundness of the problem, whereas the Primal-Dual relies, in this case, on heuristics. For a detailed description of both algorithms see Section 11. The algorithm is chosen by the **LPIPM Algorithm**, the default is Primal-Dual.

nag_opt_handle_solve_lp_ipm (e04mtc) solves linear programming problems stored as a handle. The handle points to an internal data structure which defines the problem and serves as a means of communication for functions in the NAG optimization modelling suite. First, the problem handle is initialized by **nag_opt_handle_init (e04rac)**. Then some of the functions **nag_opt_handle_set_linobj (e04rec)**, **nag_opt_handle_set_quadobj (e04rfc)**, **nag_opt_handle_set_simplebounds (e04rhc)** or **nag_opt_handle_set_linconstr (e04rjc)** may be used to formulate the objective function, bounds of the variables, and the block of linear constraints, respectively. Once the problem is fully set, the handle may be passed to the solver. When the handle is not needed anymore, **nag_opt_handle_free (e04rzc)** should be called to destroy it and deallocate the memory held within it. See **nag_opt_handle_init (e04rac)** for more details.

The solver method can be modified by various optional parameters (see Section 12) which can be set by **nag_opt_handle_opt_set (e04zmc)** and **nag_opt_handle_opt_set_file (e04zpc)** any time between the initialization of the handle by **nag_opt_handle_init (e04rac)** and a call to the solver. Once the solver has finished, options may be modified for the next solve. The solver may be called repeatedly with various optional parameters.

The optional parameter **Task** may be used to switch the problem to maximization or to ignore the objective function and find only a feasible point.

Several options might have significant impact on the performance of the solver. Even if the defaults were chosen to suit the majority of problems, it is recommended to experiment to find the most suitable set of options for a particular problem, see Sections 11 and 12 for further details.

3.1 Structure of the Lagrangian Multipliers

The algorithm works internally with estimates of both the decision variables, denoted by x, and the Lagrangian multipliers (dual variables), denoted by u. The multipliers u are stored in the array **u** and conform to the structure of the constraints.

If the simple bounds have been defined (**nag_opt_handle_set_simplebounds (e04rhc)** was successfully called), the first $2n$ elements of **u** belong to the corresponding Lagrangian multipliers, interleaving a multiplier for the lower and the upper bound for each x_i. If any of the bounds were set to infinity, the corresponding Lagrangian multipliers are set to 0 and may be ignored.

Similarly, the following $2m$ elements of **u** belong to multipliers for the linear constraints (if **nag_opt_handle_set_linconstr (e04rjc)** has been successfully called). The organization is the same, i.e., the multipliers for each constraint for the lower and upper bounds are alternated and zeros are used for any missing (infinite bound) constraint.

Some solvers merge multipliers for both lower and upper inequality into one element whose sign determines the inequality. Negative multipliers are associated with the upper bounds and positive with the lower bounds. An equivalent result can be achieved with this storage scheme by subtracting the upper bound multiplier from the lower one. This is also consistent with equality constraints.

4 References

Andersen E D, Gondzio J, Mészáros C and Xu X (1996) Implementation of interior point methods for large scale linear programming. *HEC/Université de Genève*

Colombo M and Gondzio J (2008) Further development of multiple centrality correctors for interior point methods. *Computational Optimization and Algorithms* **41(3)** 277–305

Goldfard D and Scheinberg K (2004) A product-form Cholesky factorization method for handling dense columns in interior point methods for linear programming *Mathematical Programming* **99(1)** 1–34

Gondzio J (1996) Multiple centrality corrections in a primal-dual method for linear programming *Computational Optimization and Algorithms* **6(2)** 137–156

Gondzio J (2012) Interior point methods 25 years later *European Journal of Operations Research* **218 (3)** 587–601

Hogg J D and Scott J A (2011) HSL MA97: a bit-compatible multifrontal code for sparse symmetric systems *RAL Technical Report. RAL-TR-2011-024*

HSL (2011) A collection of Fortran codes for large-scale scientific computation http://www.hsl.rl.ac.uk/

Karypis G and Kumar V (1998) A fast and high quality multilevel scheme for partitioning irregular graphs *SIAM J. Sci. Comput.* **20(1)** 359–392

Mészáros C (1996) The efficient implementation of interior point methods for linear programming and their applications *PhD Thesis* Eötvös Loránd University of Science, Budapest

Nocedal J and Wright S J (2006) *Numerical Optimization* (2nd Edition) Springer Series in Operations Research, Springer, New York

Wright S W (1997) Primal-dual interior point methods *SIAM, Philadelphia*

Xu X, Hung P-F and Ye Y (1996) A simplified homogeneous and self-dual linear programming algorithm and its implementation *Annals of Operations Research* **62(1)** 151–171

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 the problem formulated by some of the functions **nag_opt_handle_set_linobj (e04rec)**, **nag_opt_handle_set_quadobj (e04rfc)**, **nag_opt_handle_set_simplebounds (e04rhc)** and **nag_opt_handle_set_linconstr (e04rjc)**. It **must not** be changed between calls to the NAG optimization modelling suite.

2: **nvar** – Integer *Input*

On entry: n, the number of variables in the problem. It must be unchanged from the value set during the initialization of the handle by **nag_opt_handle_init (e04rac)**.

3: **x**[**nvar**] – double *Input/Output*

On entry: the input of **x** is reserved for future releases of the NAG C Library and it is ignored at the moment.

On exit: the final values of the variables x.

4: **nnzu** – Integer *Input*

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

If **nnzu** $= 0$, **u** will not be referenced; otherwise it needs to match the dimension of constraints defined by **nag_opt_handle_set_simplebounds (e04rhc)** and **nag_opt_handle_set_linconstr (e04rjc)** as explained in Section 3.1.

Constraint: **nnzu** ≥ 0.

5: **u**[**nnzu**] – double *Input/Output*

Note: if **nnzu** > 0, **u** holds Lagrange multipliers (dual variables) for the bound constraints and linear constraints. If **nnzu** $= 0$, **u** will not be referenced and may be **NULL**.

On entry: the input of **u** is reserved for future releases of the NAG C Library and it is ignored at the moment.

On exit: the final values of the variables u.

6: **rinfo**[**100**] – double *Output*

On exit: error measures and various indicators of the algorithm (see Section 11 for details) as given in the table below:

0	value of the primal objective;
1	value of the dual objective;
2	flag indicating the system formulation used by the solver, 0: augmented system, 1: normal equation;
3	factorization type, 3: Cholesky, 4: Bunch–Parlett;
4 – 13	Primal-Dual specific information (will be 0 if the Self-Dual algorithm is chosen)

4	relative dual feasibility (optimality), see (9);
5	relative primal feasibility, see (10);
6	relative duality gap (complementarity), see (11);
7	average complementarity error μ (see Section 11.1);
8	centring parameter σ (see Section 11.1);
9	primal step length;

10	dual step length;
11 – 13	reserved for future use;

14 – 23 Self-Dual specific information (will be 0 if the Primal-Dual algorithm is chosen)

14	relative primal infeasibility, see (12);
15	relative dual infeasibility, see (13);
16	relative duality gap, see (14);
17	accuracy, see (15);
18	τ, see (8);
19	κ, see (8);
20	step length;
21 – 23	reserved for future use;

24 – 99 reserved for future use.

7: **stats**[**100**] – double *Output*

On exit: solver statistics as given in the table below. Note that time statistics are provided only if **Stats Time** is set (the default is NO), the measured time is returned in seconds.

0	number of iterations;
1	total number of centrality correction steps performed;
2	total number of iterative refinements performed;
3	value of the perturbation added to the diagonal in the normal equation formulation or on the zero block in the augmented system formulation;
4	total number of factorizations performed;
5	total time spent in the solver;
6	time spent in the presolve phase;
7	time spent in the last iteration;
8	total time spent factorizing the system matrix;
9	total time spent backsolving the system matrix;
10	total time spent in the multiple centrality correctors phase;
11	time spent in the initialization phase;
12	number of nonzeros in the system matrix;
13	number of nonzeros in the system matrix factor;
14	maximum error of the backsolve;
15	number of columns in A considered dense by the solver;
16	maximum number of centrality corrector steps;
17 – 99	reserved for future use.

8: **monit** – function, supplied by the user *External Function*

monit is provided to enable you to monitor the progress of the optimization and optionally to terminate the solver early if necessary, using parameter **inform**. It is invoked at the end of every ith iteration where i is given by the optional parameter **LPIPM Monitor Frequency** (the default is 0, **monit** is not called).

monit may be specified as **NULLFN**.

The specification of **monit** is:

```
void monit (void *handle, const double rinfo[], const double stats[],
     Nag_Comm *comm, Integer *inform)
```

1: **handle** – void * *Input*

On entry: the handle to the problem as provided on entry to **nag_opt_handle_solve_lp_ipm (e04mtc)**. It may be used to query the model during the solve, and extract current approximation of the solution by **nag_opt_handle_set_get_real (e04rxc)**.

2: **rinfo**[**100**] – const double *Input*

On entry: error measures and various indicators at the end of the current iteration as described in **rinfo**.

3: **stats**[**100**] – const double *Input*

On entry: solver statistics at the end of the current iteration as described in **stats**, however, elements 2, 3, 5, 9, 10, 11 and 15 refer to the quantities in the last iteration rather than accumulated over all iterations through the whole algorithm run.

4: **comm** – Nag_Comm *

Pointer to structure of type Nag_Comm; the following members are relevant to **monit**.

user – double *
iuser – Integer *
p – Pointer

The type Pointer will be `void *`. Before calling **nag_opt_handle_solve_lp_ipm (e04mtc)** you may allocate memory and initialize these pointers with various quantities for use by **monit** when called from **nag_opt_handle_solve_lp_ipm (e04mtc)** (see Section 3.3.1.1 in How to Use the NAG Library and its Documentation).

5: **inform** – Integer * *Input/Output*

On entry: a non-negative value.

On exit: must be set to a value describing the action to be taken by the solver on return from **monit**. Specifically, if the value is negative the solution of the current problem will terminate immediately with **fail.code** = NE_USER_STOP; otherwise, computations will continue.

9: **comm** – Nag_Comm *

The NAG communication argument (see Section 3.3.1.1 in How to Use the NAG Library and its Documentation).

10: **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, **nnzu** = ⟨*value*⟩.
nnzu does not match the size of the Lagrangian multipliers for constraints.
The correct value is 0 for no constraints.

On entry, **nnzu** = ⟨*value*⟩.
nnzu does not match the size of the Lagrangian multipliers for constraints.
The correct value is either 0 or ⟨*value*⟩.

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_INFEASIBLE

The problem was found to be primal infeasible.

The primal infeasibility was detected either during the presolve phase or by the Self-Dual algorithm.

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_MAYBE_INFEASIBLE

The problem seems to be primal or dual infeasible, the algorithm was stopped.

This error is returned if the internal heuristics detected the problem to be primal or dual infeasible. It is only raised by the Primal-Dual algorithm. It is recommended to rerun the problem with the Self-Dual algorithm to confirm the infeasibility.

NE_NO_IMPROVEMENT

No progress, stopping early.

The solver predicted that it is unable to make further progress and stopped prematurely. This might be due to the scaling of the problem, its conditioning or numerical difficulties.

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 problem is already being solved.

NE_REF_MATCH

On entry, **nvar** = ⟨*value*⟩, expected value = ⟨*value*⟩.
Constraint: **nvar** must match the value given during initialization of **handle**.

NE_SETUP_ERROR

This solver does not support this problem type.

NE_TOO_MANY_ITER

Maximum number of iterations exceeded.

NE_UNBOUNDED

The problem was found to be dual infeasible.

The dual infeasibility or unboundness was detected during the presolve phase or by the Self-Dual algorithm.

NE_USER_STOP

User requested termination during a monitoring step.

NW_NOT_CONVERGED

Suboptimal solution.

The solver predicted that it is unable to reach a better estimate of the solution. However, the error measures indicate that the point is a reasonable approximation.

7 Accuracy

The accuracy of the solution is determined by optional parameters **LPIPM Stop Tolerance** and **LPIPM Stop Tolerance 2**.

If **fail.code** = NE_NOERROR on the final exit, the returned point satisfies Karush–Kuhn–Tucker (KKT) conditions to the requested accuracy (under the default settings close to $\sqrt{\epsilon}$) and thus it is a good estimate of the solution. If **fail.code** = NW_NOT_CONVERGED, some of the convergence conditions were not fully satisfied but the point is a reasonable estimate and still usable. Please refer to Section 11.3 and the description of the particular options.

8 Parallelism and Performance

nag_opt_handle_solve_lp_ipm (e04mtc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_opt_handle_solve_lp_ipm (e04mtc) makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the x06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this function. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

9.1 Description of the Printed Output

The solver can print information to give an overview of the problem and of the progress of the computation. The output may be sent to two independent streams (files) which are set by optional parameters **Print File** and **Monitoring File**. Optional parameters **Print Level**, **Print Solution** and **Print Options** determine the exposed level of detail. This allows, for example, a detailed log file to be generated while the condensed information is displayed on the screen.

By default (**Print File** = 6, **Print Level** = 2), six sections are printed to the standard output:

Header

Optional parameters list

Problem statistics

Iteration log

Summary

Solution

The iteration log varies depending on which algorithm has been selected to solve the problem (Primal-Dual or Self-Dual).

Header

The header is a message indicating the start of the solver. It should look like:

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

Optional parameters list

The list shows all options of the solver, each displayed on one line. The output contains the option name, its current value and an indicator for how it was set. The options unchanged from the default setting are noted by ‘d’, options set by the user are noted by ‘U’, and options reset by the solver are noted by ‘S’. Note that the output format is compatible with the file format expected by **nag_opt_handle_opt_set_file (e04zpc)**. The output might look as follows:

```
     Stats Time                    =                  Yes     * U
     Task                          =             Minimize     * d
     Lpipm Centrality Correctors   =                    6     * d
     Lp Presolve                   =                  Yes     * d
```

Problem statistics

If **Print Level** ≥ 2, statistics on the original and the presolved problems are printed. More detailed statistics as well as a list of the presolve operations are also printed for **Print Level** 3 or above. It should look as follows:

```
 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
```

Iteration log

If **Print Level** ≥ 2, the solver prints the status of each iteration.

Primal-Dual algorithm

If **Print Level** $= 2$, the output shows the iteration number (0 represents the starting point), the current primal and dual objective value, KKT measures (optimality, feasibility and complementarity), average complementarity error μ, number of centrality correction steps (MCC) performed and a column for additional information (I). Note that all these values are also available in **rinfo** and **stats**. The output might look as follows:

```
 ------------------------------------------------------------------------------
 it|    pobj    |    dobj    | optim  | feas   | compl  |   mu   | mcc | I
 ------------------------------------------------------------------------------
  0  2.02532E+03 -7.37272E+02  7.71E+00  4.91E+00  2.16E+00  4.4E+03
  1 -2.13398E+01 -1.62136E+04  5.82E-02  2.09E-01  2.12E+01  4.7E+02   2
  2 -1.09237E+02 -2.81254E+03  3.12E-03  4.84E-15  4.84E-01  5.3E+01   0
  3 -2.10923E+02 -6.07429E+02  4.61E-04  4.99E-14  3.70E-02  7.8E+00   0
```

If **Print Level** $= 3$, the solver also prints for each iteration the primal and dual steps as well as the maximum error of the backsolves performed. The output might look as follows:

```
---------------------------------------------------------------------------------------------------
it|    pobj     |     dobj     |  optim  |  feas    |  compl  |    mu    |  pstep  |  dstep  |  errbs  | mcc | I
---------------------------------------------------------------------------------------------------
 0  2.02532E+03 -7.37272E+02  7.71E+00  4.91E+00  2.16E+00  4.4E+03
 1 -2.13398E+01 -1.62136E+04  5.82E-02  2.09E-01  2.12E+01  4.7E+02  9.57E-01  9.92E-01  1.54E-12   2
 2 -1.09237E+02 -2.81254E+03  3.12E-03  4.84E-15  4.84E-01  5.3E+01  1.00E+00  9.46E-01  3.02E-12   0
 3 -2.10923E+02 -6.07429E+02  4.61E-04  4.99E-14  3.70E-02  7.8E+00  1.00E+00  8.53E-01  7.66E-11   0
```

Self-Dual algorithm

If **Print Level** $= 2$, the output shows the iteration number (0 represents the starting point), the current primal and dual objective value, convergence measures (primal infeasibility (12), dual infeasibility (13) and duality gap (14)), the value of the additional variable τ and the number of centrality correction steps performed. The output might look as follows:

```
--------------------------------------------------------------------------
it|    pobj     |     dobj     | p.inf  | d.inf  | d.gap  |   tau  | mcc | I
--------------------------------------------------------------------------
 0  1.01907E+01  0.00000E+00  6.98E+02  1.12E+01  1.35E+01  1.0E+00
 1  5.80391E+00 -2.39478E+00  4.98E-04  3.11E-02  2.58E-02  3.5E-01   0
 2  7.09323E+00  3.62789E+00  1.89E-04  1.18E-02  9.79E-03  1.4E-01   0
 3 -1.33628E+01  5.87563E+00  1.94E-05  1.21E-03  1.01E-03  3.3E-02   0
```

If **Print Level** $= 3$, the solver also prints for each iteration ρ_A (15), the value of the variable κ, the stepsize as well as the maximum error of the backsolves performed. The output might look as follows:

```
---------------------------------------------------------------------------------------------
----------------
it|   pobj    |    dobj    | p.inf  | d.inf  | d.gap  | rhoa  |   tau  | kappa |   step  | errbs  | mcc | I
---------------------------------------------------------------------------------------------
----------------
 0  1.01907E+01  0.00000E+00  6.98E+02  1.12E+01  1.35E+01  1.0E+01
 1  5.80391E+00 -2.39478E+00  4.98E-04  3.11E-02  2.58E-02  2.4E+00  3.5E-01  1.0E+00  6.52E-01  3.43E-13   0
 2  7.09323E+00  3.62789E+00  1.89E-04  1.18E-02  9.79E-03  7.5E-01  1.4E-01  9.8E-01  6.21E-01  2.85E-13   0
 3 -1.33628E+01  5.87563E+00  1.94E-05  1.21E-03  1.01E-03  2.8E+00  3.3E-02  7.9E-01  8.97E-01  9.31E-13   0
```

Occasionally, when numerical instabilities are too big, the solver will restart the iteration and switch to an augmented system formulation. In such cases the letters RS will be printed in the information column (I).

If **Print Level** > 3, for both the Primal-Dual and the Self-Dual algorithms, each iteration produces more information that expands over several lines. This additional information contains:

The method used (normal equation, augmented system);

The centring parameter σ;

The total number of iterative refinements performed;

The number of iterative refinements performed in the centrality correction steps;

The number of factorizations performed at the current iteration;

The type of factorization performed (Cholesky, Bunch–Parlett);

The value of the perturbation added to the diagonal in the normal equation formulation or on the zero block in the augmented system formulation;

The total time spent in the iteration if **Stats Time** is not set to NO.

The output might look as follows:

```
----------- Details of Iteration   1 ------------
method                            Normal Equation
sigma                                    6.72E-02
iterative refinements                           0
iterative refinements wmcc                      0
factorizations                                  1
matrix type                              Cholesky
diagonal perturbation                    0.00E+00
time iteration                           0.02 sec
```

Summary

Once the solver finishes, a detailed summary is produced:

```
 -------------------------------------------------
 Status: converged, an optimal solution found
 -------------------------------------------------
 Final primal objective value        -4.647531E+02
 Final dual objective value          -4.647531E+02
 Absolute primal infeasibility        1.605940E-15
 Relative primal infeasibility        1.210247E-16
 Absolute dual infeasibility          4.272004E-12
 Relative dual infeasibility          6.068111E-15
 Absolute complementarity gap         9.387105E-07
 Relative complementarity gap         2.507021E-10
 Iterations                                      7
```

It starts with the status line of the overall result which matches the **fail** value and is followed by the final primal and dual-objective values as well as the error measures and iteration count.

Optionally, if **Stats Time** is set, the timings of the different parts of the algorithm are displayed. It might look as follows:

```
 Timing
   Total time                               28.78 sec
   Presolver                                 0.07 sec   (  0.2%)
   Core                                     28.71 sec   ( 99.8%)
     Initialization                          1.67 sec   (  5.8%)
     Factorization                          16.18 sec   ( 56.5%)
     Compute directions                      5.29 sec   ( 18.5%)
     Weighted MCC                            5.50 sec   ( 19.2%)
   Iterative refinement                      0.24 sec   (  0.8%)
```

Solution

If **Print Solution** = X, the values of the primal variables and their bounds on the primary and secondary outputs. It might look as follows:

```
 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
```

If **Print Solution** = YES or ALL, the values of the dual variables are also printed. It should look as follows:

```
 Box bounds dual variables:
   idx   Lower bound       Value      Upper bound       Value
     1  -1.00000E-02    3.30098E-01   1.00000E-02    0.00000E+00
     2  -1.00000E-01    1.43844E-02   1.50000E-01    0.00000E+00
     3  -1.00000E-02    0.00000E+00   3.00000E-02    9.09967E-02
     4  -4.00000E-02    0.00000E+00   2.00000E-02    7.66124E-02
     5  -1.00000E-01    4.92258E-12   5.00000E-02    0.00000E+00
     6  -1.00000E-02    2.42274E-11       inf        0.00000E+00
     7  -1.00000E-02    4.83752E-12       inf        0.00000E+00

 Constraints dual variables:
   idx   Lower bound       Value      Upper bound       Value
     1  -1.30000E-01    0.00000E+00  -1.30000E-01    1.43111E+00
     2       -inf       0.00000E+00  -4.90000E-03    4.07810E-10
     3       -inf       0.00000E+00  -6.40000E-03    5.64870E-10
     4       -inf       0.00000E+00  -3.70000E-03    1.25984E-10
     5       -inf       0.00000E+00  -1.20000E-03    1.87338E-11
     6  -9.92000E-02    1.50098E+00       inf        0.00000E+00
     7  -3.00000E-03    1.51661E+00   2.00000E-03    0.00000E+00
```

10 Example

This example demonstrates how to use **nag_opt_handle_solve_lp_ipm (e04mtc)** to solve a small LP problem with the two algorithms implemented (Primal-Dual and Self-Dual). The solver is called twice on the same handle with different values of optional parameters.

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

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

and the general constraints

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

10.1 Program Text

```
/* nag_opt_handle_solve_lp_ipm (e04mtc) 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;
  Integer idlc;
  Integer *irowa = 0, *icola = 0;
  Integer iuser[2];
  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;

  exit_status = 0;

  printf("nag_opt_handle_solve_lp_ipm (e04mtc) 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 a high accuracy solution */
  nag_opt_handle_opt_set(handle, "LPIPM Stop Tolerance = 1.0e-10",
                         NAGERR_DEFAULT);
  /* 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);

  comm.iuser = iuser;
  iuser[0] = 1;

  /* nag_opt_handle_solve_lp_ipm (e04mtc)
   * Call LP interior point solver with the default (primal-dual) algorithm */
  printf("\n++++++++++ Use the Primal-Dual algorithm ++++++++++\n");
  fflush(stdout);
  nag_opt_handle_solve_lp_ipm(handle,nvar,x,nnzu,u,rinfo,stats,monit,
                              &comm,NAGERR_DEFAULT);

  iuser[0] = 2;
  /* Solve the same problem with the self-dual algorithm */
  printf("\n++++++++++ Use the Self-Dual algorithm ++++++++++\n");
  fflush(stdout);
  nag_opt_handle_opt_set(handle, "LPIPM Algorithm = Self-Dual",
                         NAGERR_DEFAULT);
  nag_opt_handle_opt_set(handle, "LPIPM Stop Tolerance 2 = 1.0e-11",
                         NAGERR_DEFAULT);
  nag_opt_handle_solve_lp_ipm(handle,nvar,x,nnzu,u,rinfo,stats,monit,
                              &comm,NAGERR_DEFAULT);

 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 = 1.2e-08;

  if (!comm || !comm->iuser){
    /* The communication structure is not correctly allocated, abort solve */
    *inform = -1;
    return;
  }
  /* If x is close to the solution, print a message */
  if (comm->iuser[0]==1){
    if (rinfo[4]<tol && rinfo[5]<tol &&rinfo[6]<tol){
      printf("     Iteration %"NAG_IFMT"\n", (Integer)stats[0]);
      printf("     monit() reports good approximate solution "
             "(tol =, %8.2e):\n",tol);
    }
  }
  else {
    if (rinfo[14]<tol && rinfo[15]<tol &&rinfo[16]<tol){
      printf("     Iteration %"NAG_IFMT"\n", (Integer)stats[0]);
      printf("     monit() reports good approximate solution "
             "(tol =, %8.2e):\n",tol);
    }
  }
  fflush(stdout);
}
```

10.2 Program Data

```
nag_opt_handle_solve_lp_ipm (e04mtc) 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_solve_lp_ipm (e04mtc) Example Program Results


++++++++++ Use the Primal-Dual algorithm ++++++++++

 ----------------------------------------------
  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
   5  2.35658E-02  2.35803E-02  1.32E-15  1.02E-04  8.41E-06  9.6E-06   0
   6  2.35965E-02  2.35965E-02  1.64E-15  7.02E-08  6.35E-09  7.2E-09   0
     Iteration 7
     monit() reports good approximate solution (tol =, 1.20e-08):
   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

 Box bounds dual variables:
   idx   Lower bound       Value      Upper bound       Value
     1  -1.00000E-02    3.30098E-01    1.00000E-02    0.00000E+00
     2  -1.00000E-01    1.43844E-02    1.50000E-01    0.00000E+00
     3  -1.00000E-02    0.00000E+00    3.00000E-02    9.09967E-02
     4  -4.00000E-02    0.00000E+00    2.00000E-02    7.66124E-02
     5  -1.00000E-01    3.51391E-11    5.00000E-02    0.00000E+00
     6  -1.00000E-02    3.42902E-11        inf        0.00000E+00
```

```
    7  -1.00000E-02    8.61040E-12         inf       0.00000E+00

 Constraints dual variables:
   idx   Lower bound        Value      Upper bound        Value
     1  -1.30000E-01    0.00000E+00   -1.30000E-01    1.43111E+00
     2       -inf       0.00000E+00   -4.90000E-03    4.00339E-10
     3       -inf       0.00000E+00   -6.40000E-03    1.54305E-08
     4       -inf       0.00000E+00   -3.70000E-03    3.80136E-10
     5       -inf       0.00000E+00   -1.20000E-03    4.72629E-11
     6  -9.92000E-02    1.50098E+00         inf       0.00000E+00
     7  -3.00000E-03    1.51661E+00    2.00000E-03    0.00000E+00

++++++++++ Use the Self-Dual algorithm ++++++++++

 ----------------------------------------------
  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    |  p.inf  |  d.inf  |  d.gap  |   tau  | mcc | I
 ------------------------------------------------------------------------------
   0 -6.39941E-01  4.94000E-02  1.07E+01  2.69E+00  5.54E+00  1.0E+00
   1 -8.56025E-02 -1.26938E-02  2.07E-01  2.07E-01  2.07E-01  1.7E+00   0
   2  4.09196E-03  1.24373E-02  4.00E-02  4.00E-02  4.00E-02  2.8E+00   0
   3  1.92404E-02  2.03658E-02  6.64E-03  6.64E-03  6.64E-03  3.2E+00   1
   4  1.99631E-02  2.07574E-02  3.23E-03  3.23E-03  3.23E-03  2.3E+00   1
   5  2.03834E-02  2.11141E-02  1.68E-03  1.68E-03  1.68E-03  1.4E+00   0
   6  2.22419E-02  2.25057E-02  5.73E-04  5.73E-04  5.73E-04  1.4E+00   1
   7  2.35051E-02  2.35294E-02  6.58E-05  6.58E-05  6.58E-05  1.4E+00   6
   8  2.35936E-02  2.35941E-02  1.19E-06  1.19E-06  1.19E-06  1.4E+00   0
     Iteration 9
     monit() reports good approximate solution (tol =, 1.20e-08):
   9  2.35965E-02  2.35965E-02  5.37E-10  5.37E-10  5.37E-10  1.4E+00   0
     Iteration 10
     monit() reports good approximate solution (tol =, 1.20e-08):
  10  2.35965E-02  2.35965E-02  2.68E-13  2.68E-13  2.68E-13  1.4E+00   0
 ------------------------------------------------------------------------------
 Status: converged, an optimal solution found
 ------------------------------------------------------------------------------
 Final primal objective value         2.359648E-02
 Final dual objective value           2.359648E-02
 Absolute primal infeasibility        2.853383E-12
 Relative primal infeasibility        2.677658E-13
 Absolute dual infeasibility          1.485749E-12
 Relative dual infeasibility          2.679654E-13
 Absolute complementarity gap         7.228861E-13
 Relative complementarity gap         2.683908E-13
 Iterations                                     10

 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

 Box bounds dual variables:
   idx   Lower bound        Value      Upper bound        Value
     1  -1.00000E-02    3.30098E-01    1.00000E-02    0.00000E+00
     2  -1.00000E-01    1.43844E-02    1.50000E-01    0.00000E+00
     3  -1.00000E-02    0.00000E+00    3.00000E-02    9.09967E-02
     4  -4.00000E-02    0.00000E+00    2.00000E-02    7.66124E-02
     5  -1.00000E-01    3.66960E-12    5.00000E-02    0.00000E+00
     6  -1.00000E-02    2.47652E-11         inf       0.00000E+00
     7  -1.00000E-02    7.82645E-13         inf       0.00000E+00

 Constraints dual variables:
   idx   Lower bound        Value      Upper bound        Value
     1  -1.30000E-01    0.00000E+00   -1.30000E-01    1.43111E+00
     2        -inf      0.00000E+00   -4.90000E-03    1.07904E-10
     3        -inf      0.00000E+00   -6.40000E-03    1.14799E-09
     4        -inf      0.00000E+00   -3.70000E-03    4.09190E-12
     5        -inf      0.00000E+00   -1.20000E-03    1.52421E-12
     6  -9.92000E-02    1.50098E+00         inf       0.00000E+00
     7  -3.00000E-03    1.51661E+00    2.00000E-03    0.00000E+00
```

11 Algorithmic Details

This section contains the description of the underlying algorithms used in **nag_opt_handle_solve_lp_ipm (e04mtc)**, which implements the infeasible Primal-Dual and homogeneous Self-Dual methods. For further details, see Andersen *et al.* (1996), Gondzio (2012), Mészáros (1996) and Wright (1997).

For simplicity, we consider the following primal linear programming formulation

$$\begin{array}{lll} \underset{x\in R^n}{\text{minimize}} & c^{\mathrm{T}}x & \text{(a)} \\ \text{subject to} & Ax=b & \text{(b)} \\ & x\geq 0 & \text{(c)} \end{array} \qquad (2)$$

where c, $x\in R^n$, $b\in R^m$ and $A\in R^{m\times n}$ with full row rank. The dual formulation for (2) is given by

$$\begin{array}{lll} \underset{y\in R^m, z\in R^n}{\text{maximize}} & b^{\mathrm{T}}y & \text{(a)} \\ \text{subject to} & A^{\mathrm{T}}y+z=c & \text{(b)} \\ & z\geq 0 & \text{(c)} \\ & y\ \text{(free)} & \text{(d)} \end{array} \qquad (3)$$

where y and z denote the dual variables. Solutions of the primal (2) and dual (3) problem are connected by the strong duality theory (see for example, Nocedal and Wright (2006)) and are characterized by the first order optimality conditions, the so-called Karush–Kuhn–Tucker (KKT) conditions, which are stated as follows:

$$\begin{aligned} A^{\mathrm{T}}y+z&=c \\ Ax&=b \\ XZe&=0 \\ (x,z)&\geq 0 \end{aligned} \qquad (4)$$

where we define X and Z as the diagonal matrices with the elements x_i and z_i, respectively, and $e=(1,1,\ldots,1)^{\mathrm{T}}$ as the n-vector of ones.

The underlying algorithm applies an iterative method to find an optimal solution (x^*,y^*,z^*) of the system (4) employing variants of Newton's method and modifying the search directions and step lengths so that the non-negative constraints are preserved at every iteration.

11.1 The Infeasible-interior-point Primal-Dual Algorithm

A direct solution of the nonlinear system of equations (4) by Newton's method is impractical as it exhibits slow progress towards the solution. Instead, a sequence of the following *perturbed* KKT conditions are solved so that the complementarity is driven to zero through the iteration sequence

$$\begin{aligned} Ax &= b \\ A^{\mathrm{T}}y + z &= c \\ XZe &= \sigma\mu e \\ (x,z) &> 0. \end{aligned} \tag{5}$$

Here μ is the average complementarity error at the current iteration $\mu = x^{\mathrm{T}}z/n$, called *duality measure*, and $\sigma \in (0,1)$, called *centring* parameter, is the reduction factor that we wish to achieve in the duality measure.

Each iteration of the Primal-Dual algorithm makes one step of Newton's method applied to the perturbed first order optimality conditions (5) with a given σ and μ. In particular, a Newton search direction is computed by solving a system of linear equations and a length of the step α is determined so that the bounds $(x,z) > 0$ are not violated. The residual of (4) and μ define stopping criteria and the algorithm terminates once they are reduced to the requested accuracy, see Section 11.3.

Given an x, $z \in R^n_+$ and $y \in R^m$, Newton's direction is obtained by solving the following system:

$$\begin{pmatrix} A & 0 & 0 \\ 0 & A^{\mathrm{T}} & I \\ Z & 0 & X \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta s \\ \Delta z \end{pmatrix} = \begin{pmatrix} r_b \\ r_c \\ \mu e - XZe \end{pmatrix}, \tag{6}$$

where

$$\begin{aligned} r_b &= b - Ax, && \text{(Primal infeasibilities)} \\ r_c &= c - A^{\mathrm{T}}y - z && \text{(Dual infeasibilities)} \end{aligned}$$

denote the violations of the primal and the dual constraints, respectively. Primal and dual infeasibilities, r_b and r_c, are reduced at the same rate $(1-\alpha)$, given a stepsize $\alpha \in (0,1)$. The Primal-Dual algorithm does not require feasibility of the solutions during the optimization process. Feasibility is attained during the process as optimality is approached.

Once the system is solved, Δx and Δz are used to compute the maximum stepsizes in primal space (α_P) and dual space (α_D) such that the non-negativity of variables is preserved

$$\begin{aligned} \alpha_P &= \min\{1, \rho \cdot \max\{\alpha \geq 0 : x^k + \alpha\Delta x^k \geq 0\}\} \\ \alpha_D &= \min\{1, \rho \cdot \max\{\alpha \geq 0 : z^k + \alpha\Delta z^k \geq 0\}\} \end{aligned},$$

where $0 < \rho < 1$ is a reduction parameter close to 1, typically $\rho \in (0.9, 1)$. The next iterate is updated as follows:

$$\begin{aligned} x^{k+1} &\leftarrow x^k + \alpha_P \Delta x^k \\ (y^{k+1}, z^{k+1}) &\leftarrow (y^k, z^k) + \alpha_D(\Delta y^k, \Delta z^k) \end{aligned}$$

Finally, the barrier parameter μ is decreased by a given factor and the process is repeated until the stopping criteria (see Section 11.3) or maximum number of iterations is reached.

11.1.1 The Barrier Method

Note that there is also another way to obtain the perturbed KKT conditions (5). They can be derived starting from the primal formulation (2) and replacing the non-negativity constraints $x \geq 0$ by a logarithmic barrier term $\sum_{j=1}^{n} \log x_j$ with a *barrier parameter* $\tau > 0$. This approach leads to the primal logarithmic barrier problem defined as

$$\begin{aligned} &\underset{x \in R^n}{\text{minimize}} && c^{\mathrm{T}}x - \tau \textstyle\sum_{j=1}^{n} \log x_j \\ &\text{subject to} && Ax = b. \end{aligned} \tag{7}$$

The Lagrangian formulation for (7) is

$$L(x, y, \tau) = c^{\mathrm{T}}x - \tau\sum_{j=1}^{n}\log x_j - y^{\mathrm{T}}(Ax - b),$$

and the first order optimality conditions for problem (7) are

$$\begin{array}{rcl} \nabla_x L & = & c - \tau X^{-1}e - A^{\mathrm{T}}y = 0 \\ \nabla_y L & = & b - Ax = 0, \\ x & > & 0 \quad \text{(Barrier avoids} x = 0) \end{array}$$

Finally, by introducing $Z = \tau X^{-1}$ and $z = Ze$ the same system of equations (5) is formulated.

11.2 Homogeneous Self-Dual Algorithm

A homogeneous Self-Dual (HSD) embedding of the primal linear programming and its dual was proposed in Xu *et al.* (1996). As its name suggest, the HSD and its dual are equivalent. Self-Dual formulations embed the original problem (2) in a larger linear programming problem such that the latter is primal and dual feasible, with known feasible points, and from which solution we can extract optimal solutions or certificates of infeasibility of the original problem.

We define the homogeneous and Self-Dual linear feasibility (HLF) model as follows:

$$\begin{array}{ll} \underset{x,z\in R^n, y\in R^m, \tau,\kappa\in R}{\text{minimize}} & 0 \\ \text{subject to} & Ax - b\tau = 0 \\ & A^{\mathrm{T}}y + z - c\tau = 0 \\ & -c^{\mathrm{T}}x + b^{\mathrm{T}}y - \kappa = 0 \\ & x, \tau, z, \kappa \geq 0, y \text{ (free)} \end{array}, \tag{8}$$

where τ and κ are two additional variables. The model (8) is a Self-Dual linear programming problem with zero right-hand side and a zero objective vector. If $(\hat{x}, \hat{\tau}, \hat{y}, \hat{z}, \hat{\kappa})$ is a strictly complementarity solution for (8), then if $\hat{\tau} > 0$, the linear programming problem has an optimal solution given by

$$(x^*, y^*, z^*) = (\hat{x}, \hat{y}, \hat{z})/\hat{\tau},$$

and the duality gap is given by $c^{\mathrm{T}}x^* - b^{\mathrm{T}}y^* = \hat{\kappa}/\hat{\tau} = 0$. The homogeneous algorithm is an application of the Primal-Dual method for the computation of a strictly complementarity solution to (8).

Homogeneous and Self-Dual interior point methods have several advantages besides an inherent ability to detect infeasibility (which improves the detection of divergence in Primal-Dual algorithms). The HSD model includes the ease of finding a suitable starting point and it is generally more robust in the presence of free variables. However, some disadvantages need to be noted: HSD is larger than the original problem. In particular, it increases the number of linear equations solved per iteration by one, requiring an extra backsolve step, which make it slightly slower than the Primal-Dual algorithm. Moreover, numerical experiments indicate that the required number of iterations on feasible problems might be slightly increased.

11.3 Stopping Criteria

11.3.1 Convergence – Optimal Termination

11.3.1.1 Primal-Dual algorithm

The Primal-Dual algorithm is stopped when the first order optimality conditions are satisfied to the requested accuracy. These conditions are relative primal and dual feasibility and duality gap, defined as:

relative dual feasibility

$$\frac{\|A^{\mathrm{T}}y + z - c\|}{1 + \|c\|} \leq \epsilon_1 \tag{9}$$

relative primal feasibility

$$\frac{\|Ax - b\|}{1 + \|b\|} \leq \epsilon_1 \tag{10}$$

relative duality gap

$$\frac{\mu}{1 + |c^{\mathrm{T}}x|} \leq \epsilon_1 \tag{11}$$

Furthermore, final absolute primal and dual infeasibilities, and duality gap are also returned. Here ϵ_1 may be set using **LPIPM Stop Tolerance** and the norm denotes the 2-norm.

11.3.1.2 Self-Dual algorithm

Similar to the Primal-Dual algorithm, the homogeneous Self-Dual algorithm is stopped when the following measures are satisfied to the requested accuracy.

relative primal feasibility

$$\frac{\|b\tau - Ax\|}{\max(1, \|b\tau_0 - Ax_0\|)} \leq \epsilon_1 \tag{12}$$

relative dual feasibility

$$\frac{\|\tau c - A^{\mathrm{T}}y - z\|}{\max(1, \|c\tau_0 - A^{\mathrm{T}}y_0 - z_0\|)} \leq \epsilon_1 \tag{13}$$

relative duality gap

$$\frac{\|\kappa + c^{\mathrm{T}}x - b^{\mathrm{T}}y\|}{\max(1, |\kappa_0 + c^{\mathrm{T}}x_0 - b^{\mathrm{T}}y_0|)} \leq \epsilon_1 \tag{14}$$

which measure the relative reduction in the primal, dual and gap infeasibility, respectively. In addition, an extra measure is considered to quantify the accuracy in the objective function, which is given by

$$\rho_A = \frac{|c^{\mathrm{T}}x - b^{\mathrm{T}}y|}{\tau + |b^{\mathrm{T}}y|} \leq \epsilon_2 \tag{15}$$

Here ϵ_1 and ϵ_2 may be set using **LPIPM Stop Tolerance** and **LPIPM Stop Tolerance 2**, respectively.

Premature termination is triggered if the current iteration exhibits fast convergence and the optimality measures lie within a small range. In particular, the Self-Dual algorithm is stopped if the above termination conditions are met within a small factor and $\tau > 1000\kappa$. This measure is tracked after the first 10 iterations.

11.3.2 Infeasibility/Unboundedness Detection

11.3.2.1 Primal-Dual algorithm

The Primal-Dual algorithm detects infeasible problems fairly reliably by using a set of heuristics. When several of these heuristics classify the problem as infeasible throughout a sufficient number of iterations, the algorithm is stopped.

Note that in order to obtain a certificate of infeasibility, the use of homogeneous Self-Dual algorithm is recommended, see Section 11.3.2.2.

11.3.2.2 Self-Dual algorithm

The problem is concluded to be primal or dual infeasible if one of the following conditions hold:

1. Both the relative primal (12) and dual (13) feasibility of the HLF model (8) are satisfied and the value of τ satisfies

$$\tau \leq \epsilon_2 \max(1, \kappa)$$

2. or if the following inequalities hold

$$\mu \leq \epsilon_2 \mu_0$$
$$\tau \leq \epsilon_2 \min(1, \kappa)$$

Then the problem is declared dual infeasible if $c^{\mathrm{T}}x < 0$ or primal infeasible otherwise.

11.3.3 Suboptimal Solution

The solver stops prematurely and reports suboptimal solution when it predicts that the current estimate of the solution will not be improved in subsequent iterations. In most cases the returned solution should be acceptable.

11.4 Solving the KKT System

The solution of the Newton system of equations (6) is the most computationally costly operation. In practice, system (6) is reduced to the *augmented system* by eliminating Δz from the last block of equations as follows:

$$\begin{pmatrix} -D^2 & A^{\mathrm{T}} \\ A & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} r \\ h \end{pmatrix}, \tag{16}$$

where

$$\begin{aligned} D^2 &= Z^{-1}X \\ r &= rc - X^{-1}(\mu e - XZe) \\ h &= r_b \end{aligned}$$

and $\Delta z = -X^{-1}(\mu e - XZe) - X^{-1}Z\Delta x$. This is a system of $m + n$ variables and constraints, symmetric and indefinite. Submatrix D is diagonal and positive definite.

The system (16) can be reduced further by eliminating Δx, to a positive definite system usually called *normal-equations* defined as

$$\left(AD^2A^{\mathrm{T}}\right)\Delta y = AD^2r + h \tag{17}$$

and

$$\begin{aligned} \Delta z &= -rc - A^{\mathrm{T}}\Delta y \\ \Delta x &= -Z^{-1}(\mu e - XZe) - XZ^{-1}\Delta z \end{aligned}$$

Typically, formulation (17) is preferred for many problems as the system matrix can be factorized by a sparse Cholesky. However, this brings some well-known disadvantages: Ill-conditioning of the system is often observed during the final stages of the algorithm, and free (unbounded) variables require certain modifications. If matrix A contains dense columns (columns with relatively many nonzeros) then AD^2A^{T} has many nonzeros, which in turn makes the factorization expensive. On the other hand, solving the augmented system is usually slower, but it normally avoids the fill-in caused by dense columns and can handle free variables directly in the formulation.

nag_opt_handle_solve_lp_ipm (e04mtc) can detect and handle dense columns effectively: depending on the number and the density of the ‘dense’ columns, the solver may either choose to directly use an augmented system formulation or to treat these columns separately in a product-form Cholesky factorization as described in Goldfard and Scheinberg (2004). It is also possible to manually override the automatic choice via the optional parameter **LPIPM System Formulation** and let the solver use a normal-equations or an augmented system formulation.

Badly scaled optimal solutions may present numerical challenges, therefore iterative refinement using mixed-precision is employed for reducing the roundoff errors produced during the solution of the system. When the condition number of the system AD^2A^T prevents the satisfactory use of iterative refinement, **nag_opt_handle_solve_lp_ipm (e04mtc)** switches automatically to an augmented system formulation, reporting RS (Restart) in the last column of the iteration log (I). Furthermore, **nag_opt_handle_solve_lp_ipm (e04mtc)** provides several scaling techniques to adjust the numerical characteristics of the problem data, see **LPIPM Scaling**.

Finally, factorization of the system matrix can degrade sparsity, so the resulting fill-in can be large, therefore several ordering techniques are included to minimize it. **nag_opt_handle_solve_lp_ipm (e04mtc)** uses Harwell packages MA97 (see Hogg and Scott (2011) and HSL (2011)) for the underlying sparse linear algebra factorization and MC68 approximate minimum degree algorithm, and METIS (Karypis and Kumar (1998)) nested dissection algorithm for the ordering.

11.5 Weighted Multiple Centrality Correctors

As previously stated, the factorization of the system at every iteration usually accounts for most of the computation time, therefore it is always desirable to reuse the factors if possible and to reduce the total number of iterations. An efficient computational method is obtained by splitting the computation of the Newton direction into two steps, namely the affine-scaling direction and its correction step, called Mehrotra's predictor-corrector. However, Mehrotra's predictor-corrector technique aims to correct the affine-direction in a *full* step, which is considered an aggressive approach.

nag_opt_handle_solve_lp_ipm (e04mtc) implements a high-order method, called weighted multiple centrality correctors (WMCC), see Gondzio (1996) and Colombo and Gondzio (2008). This technique attempts to correct the affine-direction recursively as long as the stepsizes increase at least by a fraction of a given aspiration level, up to a maximum number of times. The heuristic to determine the maximum number of wmcc is based on the ratio between the cost of the factorization and that of backsolving and therefore may lead to **non-repeatable** results. To avoid an undesired behaviour, you can fix the maximum number of WMCC with option **LPIPM Centrality Correctors**.

11.6 Further Details

nag_opt_handle_solve_lp_ipm (e04mtc) includes an advance preprocessing phase (called presolve) to reduce the dimensions of the problem before passing it to the solver. The reduction in problem size generally improves the behaviour of the solver, shortening the total computation time. In addition, infeasibility may also be detected during preprocessing. The default behaviour of the presolve can be modified by **LP Presolve**.

The initial point x_0 is always computed using heuristics. Effective *warm-starting* strategies for interior point methods are still subject to intensive academic research, and it is not available in this release.

12 Optional Parameters

Several optional parameters in **nag_opt_handle_solve_lp_ipm (e04mtc)** define choices in the problem specification or the algorithm logic. In order to reduce the number of formal arguments of **nag_opt_handle_solve_lp_ipm (e04mtc)** these optional parameters have associated *default values* that are appropriate for most problems. Therefore, you need only specify those optional parameters whose values are to be different from their default values.

The remainder of this section can be skipped if you wish to use the default values for all optional parameters.

The optional parameters can be changed by calling **nag_opt_handle_opt_set (e04zmc)** anytime between the initialization of the handle by **nag_opt_handle_init (e04rac)** and the call to the solver. Modification of the arguments during intermediate monitoring stops is not allowed. Once the solver finishes, the optional parameters can be altered again for the next solve.

If any options are set by the solver (typically those with the choice of AUTO), their value can be retrieved by **nag_opt_handle_opt_get (e04znc)**. If the solver is called again, any such arguments are reset to their default values and the decision is made again.

The following is a list of the optional parameters available. A full description of each optional parameter is provided in Section 12.1.

Defaults
Infinite Bound Size
LPIPM Algorithm
LPIPM Centrality Correctors
LPIPM Iteration Limit
LPIPM Monitor Frequency
LPIPM Scaling
LPIPM Stop Tolerance
LPIPM Stop Tolerance 2
LPIPM System Formulation
LP Presolve
Monitoring File
Monitoring Level
Print File
Print Level
Print Options
Print Solution
Stats Time
Task

12.1 Description of the Optional Parameters

For each option, we give a summary line, a description of the optional parameter and details of constraints.

The summary line contains:

the keywords;

a parameter value, where the letters a, i and r denote options that take character, integer and real values respectively;

the default value, where the symbol ϵ is a generic notation for ***machine precision*** (see **nag_machine_precision (X02AJC)**).

All options accept the value DEFAULT to return single options to their default states.

Keywords and character values are case and white space insensitive.

Defaults

This special keyword may be used to reset all optional parameters to their default values. Any argument value given with this keyword will be ignored.

Infinite Bound Size r Default $= 10^{20}$

This defines the 'infinite' bound $bigbnd$ in the definition of the problem constraints. Any upper bound greater than or equal to $bigbnd$ will be regarded as $+\infty$ (and similarly any lower bound less than or equal to $-bigbnd$ will be regarded as $-\infty$). Note that a modification of this optional parameter does not influence constraints which have already been defined; only the constraints formulated after the change will be affected.

Constraint: **Infinite Bound Size** ≥ 1000.

LP Presolve a Default $=$ FULL

This argument allows you to reduce the level of presolving of the problem or turn it off completely. If the presolver is turned off, the solver will try to handle the problem as given by the user. In such a case, the presence of fixed variables or linear dependencies in the constraint matrix can cause numerical instabilities to occur. In normal circumstances, it is recommended to use the full presolve which is the default.

Constraint: **LP Presolve** $=$ FULL, BASIC or NO.

LPIPM Algorithm a Default $=$ PRIMAL $-$ DUAL

As described in Section 11, **nag_opt_handle_solve_lp_ipm (e04mtc)** implements the infeasible Primal-Dual algorithm, see Section 11.1, and the homogeneous Self-Dual algorithm, see Section 11.2. This argument controls which one to use.

Constraint: **LPIPM Algorithm** $=$ PRIMAL $-$ DUAL, PD, SELF $-$ DUAL or SD.

LPIPM Centrality Correctors i Default $= 6$

This argument controls the number of centrality correctors (see Section 11.5) used at each iteration. Each corrector step attempts to improve the current iterate for the price of additional solve(s) of the factorized system matrix in order to reduce the total number of iterations. Therefore, it trades the additional solves of the system with the number of factorizations. The more expensive the factorization is with respect to the solve, the more corrector steps should be allowed.

If $i > 0$, the maximum number of corrector steps will be computed by timing heuristics (the ratio between the times of the factorization and the solve in the first iteration) but will not be greater than i. The number computed by the heuristic can be recovered after the solve or during a monitoring step in **stats**. This might cause non-repeatable results.

If $i < 0$, the maximum number of corrector steps will be set to $|i|$.

If it is set to 0, no additional centrality correctors will be used and the algorithm reverts to Mehrotra's predictor-corrector.

LPIPM Iteration Limit i Default $= 100$

The maximum number of iterations to be performed by **nag_opt_handle_solve_lp_ipm (e04mtc)**. Setting the option too low might lead to **fail.code** $=$ NE_TOO_MANY_ITER.

Constraint: **LPIPM Iteration Limit** ≥ 1.

LPIPM Scaling a Default $=$ ARITHMETIC

This argument controls the type of scaling to be applied on the constraint matrix A before solving the problem. More precisely, the scaling procedure will try to find diagonal matrices D_1 and D_2 such that the values in D_1AD_2 are of a similar order of magnitude. The solver is less likely to run into numerical difficulties when the constraint matrix is well scaled.

Constraint: **LPIPM Scaling** $=$ ARITHMETIC, GEOMETRIC or NONE.

LPIPM Monitor Frequency i Default $= 0$

This argument defines the frequency of how often function **monit** is called. If $i > 0$, the solver calls **monit** at the end of every ith iteration. If it is set to 0, the function is not called at all.

Constraint: **LPIPM Monitor Frequency** ≥ 0.

LPIPM Stop Tolerance r Default $= \sqrt{\epsilon}$

This argument sets the value ϵ_1 which is the tolerance for the convergence measures in the stopping criteria, see Section 11.3.

Constraint: **LPIPM Stop Tolerance** $> \epsilon$.

LPIPM Stop Tolerance 2 r Default $= \epsilon^{0.6}$

This argument sets the additional tolerance ϵ_2 used in the stopping criteria for the Self-Dual algorithm, see Section 11.3.

Constraint: **LPIPM Stop Tolerance 2** $> \epsilon$.

LPIPM System Formulation a Default $=$ AUTO

As described in Section 11.4, **nag_opt_handle_solve_lp_ipm (e04mtc)** can internally work either with the normal equations formulation (17) or with the augmented system (16). A brief discussion of advantages and disadvantages is presented in Section 11.4. Option AUTO leaves the decision to the solver based on the structure of the constraints and it is the recommended option. This will typically lead to the normal equations formulation unless there are many dense columns or the system is significantly cheaper to factorize as the augmented system. Note that in some cases even if **LPIPM System Formulation** $=$ NORMAL EQUATIONS the solver might switch the formulation through the computation to the augmented system due to numerical instabilities or computational cost.

Constraint: **LPIPM System Formulation** $=$ AUTO, AUGMENTED SYSTEM, AS, NORMAL EQUATIONS or NE.

Monitoring File i Default $= -1$

(See Section 3.3.1.1 in How to Use the NAG Library and its Documentation for further information on NAG data types.)

If $i \geq 0$, the Nag_FileID number (as returned from **nag_open_file (x04acc)**) for the secondary (monitoring) output. If set to -1, no secondary output is provided. The following information is output to the unit:

- a listing of the optional parameters if set by **Print Options**;
- problem statistics, the iteration log and the final status as set by **Monitoring Level**;
- the solution if set by **Print Solution**.

Constraint: **Monitoring File** ≥ -1.

Monitoring Level i Default $= 4$

This argument sets the amount of information detail that will be printed by the solver to the secondary output. The meaning of the levels is the same as with **Print Level**.

Constraint: $0 \leq$ **Monitoring Level** ≤ 5.

Print File i Default
$=$ Nag_FileID number associated with stdout

(See Section 3.3.1.1 in How to Use the NAG Library and its Documentation for further information on NAG data types.)

If $i \geq 0$, the Nag_FileID number (as returned from **nag_open_file (x04acc)**, `stdout` as the default) for the primary output of the solver. If **Print File** $= -1$, the primary output is completely turned off independently of other settings. The following information is output to the unit:

- a listing of optional parameters if set by **Print Options**;
- problem statistics, the iteration log and the final status from the solver as set by **Print Level**;
- the solution if set by **Print Solution**.

Constraint: **Print File** ≥ -1.

Print Level i Default $= 2$

This argument defines how detailed information should be printed by the solver to the primary output.

i	**Output**
0	No output from the solver
1	Only the final status and the primal and dual objective value
2	Problem statistics, one line per iteration showing the progress of the solution with respect to the convergence measures, final status and statistics

3	As level 2 but each iteration line is longer, including step lengths and errors
4, 5	As level 3 but further details of each iteration are presented

Constraint: $0 \leq$ **Print Level** ≤ 5.

Print Options a Default $=$ YES

If **Print Options** $=$ YES, a listing of optional parameters will be printed to the primary and secondary output.

Constraint: **Print Options** $=$ YES or NO.

Print Solution a Default $=$ NO

If **Print Solution** $=$ X, the final values of the primal variables are printed on the primary and secondary outputs.

If **Print Solution** $=$ YES or ALL, in addition to the primal variables, the final values of the dual variables are printed on the primary and secondary outputs.

Constraint: **Print Solution** $=$ YES, NO, X or ALL.

Stats Time a Default $=$ NO

This argument allows you to turn on timings of various parts of the algorithm to give a better overview of where most of the time is spent. This might be helpful for a choice of different solving approaches. It is possible to choose between CPU and wall clock time. Choice YES is equivalent to wall clock.

Constraint: **Stats Time** $=$ YES, NO, CPU or WALL CLOCK.

Task a Default $=$ MINIMIZE

This argument specifies the required direction of the optimization. If **Task** $=$ FEASIBLE POINT, the objective function (if set) is ignored and the algorithm stops as soon as a feasible point is found with respect to the given tolerance. If no objective function is set, **Task** reverts to FEASIBLE POINT automatically.

Constraint: **Task** $=$ MINIMIZE, MAXIMIZE or FEASIBLE POINT.