NAG Library Routine Document

E04RAF

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details.

1 Purpose

E04RAF initializes a data structure for the NAG optimization modelling suite for problems such as, quadratic programming (QP), nonlinear programming (NLP), linear semidefinite programming (SDP) and semidefinite programming with bilinear matrix inequalities (BMI-SDP).

2 Specification

```
SUBROUTINE E04RAF (HANDLE, NVAR, IFAIL)
INTEGER      NVAR, IFAIL
TYPE (C_PTR) HANDLE
```

3 Description

E04RAF initializes an empty problem with n decision variables, x, and returns a handle to the data structure. This handle may then be passed to some of the routines E04REF, E04RFF, E04RGF, E04RHF, E04RJF, E04RKF, E04RLF, E04RNF and E04RPF to formulate the problem (define the objective function and constraints) and to a compatible solver, E04STF or E04SVF, to solve it. The handle **must not** be changed between calls. When the handle is no longer needed, E04RZF must be called to destroy it and deallocate all the allocated memory and data within. In addition, the suite comprises auxiliary routines for printing (E04RYF), for setting optional parameters (E04ZMF and E04ZPF), for retrieving them (E04ZNF) and for reading data files for linear semidefinite programming (E04RDF).

The handle can store various problem formulations, including quadratic programming (QP)

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

nonlinear programming (NLP)

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

linear semidefinite programming (SDP)

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

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

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

where H, A_i^k and Q_{ij}^k denote symmetric matrices, B is a general rectangular matrix, m_A is the number of semidefinite constraints (matrix inequalities) and c, l and u are vectors. The expression $S \succeq 0$ stands for a constraint on eigenvalues of a symmetric matrix S, namely, all the eigenvalues should be non-negative, i.e., the matrix S should be positive semidefinite.

3.1 Life Cycle of the Handle

Each handle should pass four stages in its life as depicted in the diagram below. These are *initialization*, *problem formulation*, *problem solution* and *deallocation*. The initialization by E04RAF and deallocation by E04RZF mark the beginning and the end of the life of the handle. During this time the handle must only be modified by the provided routines. Working with a handle which has not been properly initialized will result in IFAIL $= 1$ (uniform across the suite) and is potentially very dangerous as it may cause unpredictable behaviour.

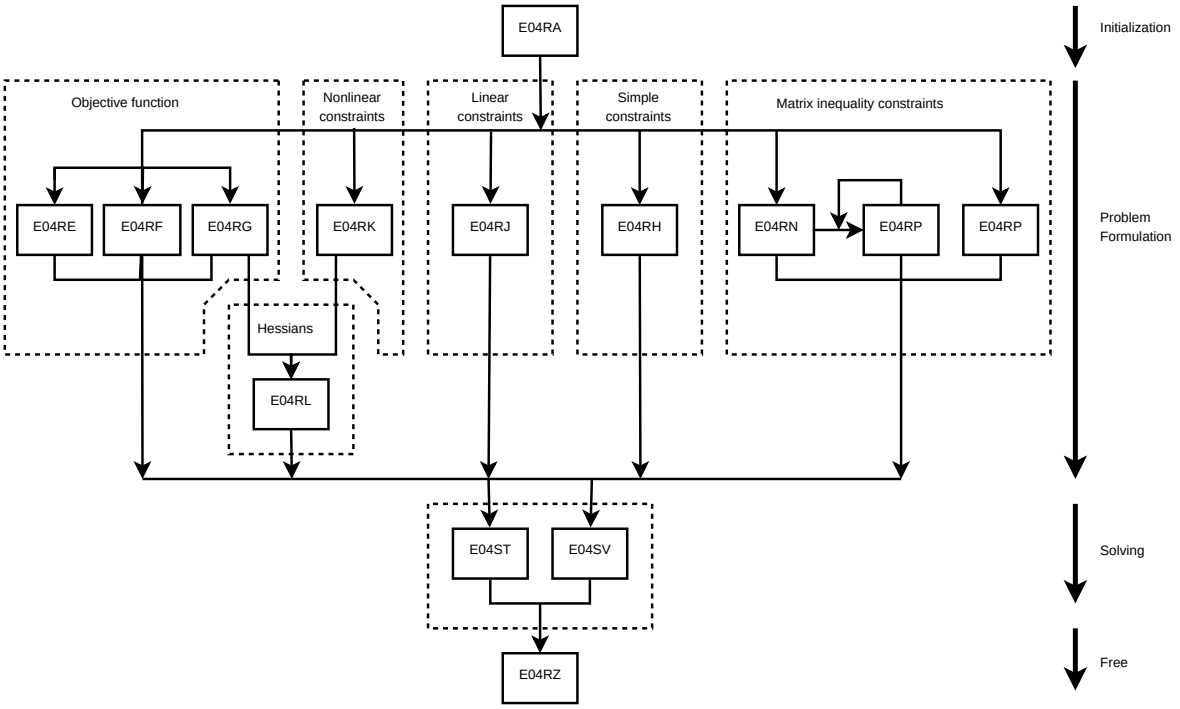
After the handle has been initialized, various routines are provided to add the following basic building blocks to the problem formulation: objective function, simple variable bounds, (standard) linear constraints and matrix constraints. Some of these can be defined at most once (e.g., objective function) and an attempt to redefine them will cause IFAIL $= 3$. Others (matrix constraints) may be composed by several repetitive calls. The routines work in a tight cooperation, if the provided data is not compatible with the previous information, IFAIL $= 4$ is returned.

The handle may be passed to E04REF to define the linear objective function (3)(a), to E04RFF for the quadratic objective function (1)(a), (4)(a), to E04RGF to declare the objective function as a nonlinear function (2)(a) or neither of them if the problem is just to find a feasible point satisfying the constraints. If present, the simple bounds on variables (box constraints, (1)(c), (2)(d), (3)(d), (4)(d)) may be defined by E04RHF. The linear constraints ((1)(b), (2)(c), (3)(c) and (4)(c)) are set by E04RJF. The nonlinear constraints (2)(b) may be declared by E04RKF. If the second derivatives of the nonlinear objective and constraints are available they may be supplied via E04RLF. The linear matrix inequalities (3)(b) or the linear part of (4)(b) are defined by E04RNF, and this call can be repeated several times if more matrix inequality constraints are required. Any existing (already defined) linear matrix inequalities can be extended by bilinear matrix terms in (4)(b) by one or more calls to E04RPF. The routines E04REF, E04RFF, E04RGF, E04RHF, E04RJF, E04RKF, E04RLF, E04RNF and E04RPF may be called in an arbitrary order, however, a call to E04RNF must precede a call to E04RPF for the matrix inequalities with bilinear terms and the nonlinear objective or constraints (E04RGF or E04RKF) must precede the definition of the second derivatives by E04RLF.

When the problem is fully formulated, the handle can be passed to a solver which is compatible with the defined problem. At Mark 26 the NAG optimization modelling suite comprises of E04STF and E04SVF. If the solver cannot deal with the given problem, IFAIL $= 2$ is returned. Once the solver is called, no further modifications of the problem formulation are allowed and calling any of the routines defining the objective function or the constraints will result in IFAIL $= 2$. The solver may be called repetitively, for example, with various optional parameters and/or starting points.

Any optional parameters may be set by a call to E04ZMF at any time between the initialization by E04RAF and the call to the solver or after the solver returns. Several optional parameters can be modified at once by E04ZPF when an option file is used. The current value of the optional parameters may be retrieved by E04ZNF.

For further details, see the documentation of the individual routines and the solvers which also contain a description of all the optional parameters.



4 References

None.

5 Arguments

1: HANDLE – TYPE (C_PTR) *Output*

Note: HANDLE does not need to be set on input.

On exit: holds a handle to the internal data structure where an empty problem with NVAR variables is defined. You **must not** change the handle until the call to E04RZF (deallocation).

2: NVAR – INTEGER *Input*

On entry: n, the number of decision variables in the problem.

Constraint: NVAR > 0.

3: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation for details.

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, if you are not familiar with this argument, the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

On exit: IFAIL $= 0$ unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

If on entry IFAIL $= 0$ or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

IFAIL $= 6$

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

IFAIL $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

See Section 3.9 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -399$

Your licence key may have expired or may not have been installed correctly.

See Section 3.8 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.7 in How to Use the NAG Library and its Documentation for further information.

7 Accuracy

Not applicable.

8 Parallelism and Performance

E04RAF is not threaded in any implementation.

9 Further Comments

None.

10 Example

See examples associated with other routines of the suite:

– the example in Section 10 in E04RDF demonstrates how to use the SDPA file reader and how to solve linear semidefinite programming problems, including printing of the matrix Lagrangian multipliers,

– the example in Section 10 in E04RFF presents an alternative way to compute the nearest correlation matrix by means of nonlinear semidefinite programming,

– a matrix completion problem (minimization of a rank of a partially unknown matrix) formulated as SDP is demonsrated in Section 10 in E04RHF, the example also demonstrates monitoring mode of the solver E04SVF,

– the example in Section 10 in E04RJF solves LP/QP problems read in from an MPS file by E04MXF,

– an application for statistics, E optimal design, solved as an SDP problem is shown in Section 10 in E04RNF,

– the example in Section 10 in E04RPF reads BMI-SDP problem from a file which might be modified by users, in this case it solves Static Output Feedback (SOF) problem,

– the example in Section 10 in E04RYF walks through the life cycle of the handle in which a BMI-SDP problem is formulated and solved,

– an example in Section 10 in E04STF is a small test from Hock and Schittkowski set to show how to call the NLP solver,

– the simple example in Section 10 in E04SVF demonstrates on the LovÄsz ϑ function eigenvalue optimization problem formulated as SDP.
