NAG Library Routine Document

e04rmf

1 Purpose

e04rmf is a part of the NAG optimization modelling suite and defines the number of residuals in a sum of squares objective function (nonlinear least squares problems) and, optionally, the sparsity structure of their first derivatives.

2 Specification

Fortran Interface

```
Subroutine e04rmf (handle, nres, isparse, nnzrd, irowrd, icolrd, ifail)
Integer, Intent (In)      :: nres, isparse, nnzrd, irowrd(nnzrd),          &
                             icolrd(nnzrd)
Integer, Intent (Inout)   :: ifail
Type (c_ptr), Intent (In) :: handle
```

3 Description

After the initialization routine **e04raf** has been called and unless the objective function has already been defined, **e04rmf** may be used to declare the objective function of the optimization problem as a sum of squares. It will typically be used in data fitting or calibration problems of the form

$$\begin{aligned} \underset{x \in R^n}{\text{minimize}} \quad & f(x) = \sum_{j=1}^{m_r} r_j(x)^2 \\ \text{subject to} \quad & l_x \leq x \leq u_x, \end{aligned}$$

where x is an n-dimensional variable vector and $r_i(x)$ are nonlinear residuals (see Section 2.2.3 in the E04 Chapter Introduction). The values of the residuals, and possibly their derivatives, will be communicated to the solver by a user-supplied function. **e04rmf** also allows the user to declare the structured first derivative matrix

$$\left[\frac{\partial r_j(x)}{\partial x_i}\right]_{i=1,\ldots,n,\ j=1,\ldots,m_r}$$

as being dense or sparse. If declared as sparse, its sparsity structure must be specified here.

See **e04raf** for more details.

4 References

None.

5 Arguments

1: **handle** – Type (c_ptr) *Input*

On entry: the handle to the problem. It needs to be initialized by **e04raf** and **must not** be changed.

2: **nres** – Integer *Input*

On entry: m_r, the number of residuals in the objective function.

If **nres** $= 0$, no objective function will be defined and **irowrd** and **icolrd** will not be referenced.

Constraint: **nres** ≥ 0.

3: **isparse** – Integer *Input*

On entry: is a flag indicating if the nonzero structure of the first derivative matrix is dense or sparse.

isparse $= 0$
The first derivative matrix is considered dense and **irowrd** and **icolrd** will not be referenced. The ordering is assumed to be column-wise, namely the routine will behave as if **nnzrd** $= n \times m_r$ and the vectors **irowrd** and **icolrd** filled as:

$$\textbf{irowrd} = (1, 2, \ldots, n, 1, 2, \ldots, n, \ldots, 1, 2, \ldots, n);$$

$$\textbf{icolrd} = (1, 1, \ldots, 1, 2, 2, \ldots, 2, \ldots, m_r, m_r, \ldots, m_r).$$

isparse $= 1$
The sparsity structure of the first derivative matrix will be supplied by **nnzrd**, **irowrd** and **icolrd**.

Constraint: **isparse** $= 0$ or 1.

4: **nnzrd** – Integer *Input*

On entry: the number of nonzeros in the first derivative matrix.

Constraint: if **nres** > 0, **nnzrd** > 0.

5: **irowrd**(**nnzrd**) – Integer array *Input*
6: **icolrd**(**nnzrd**) – Integer array *Input*

On entry: arrays **irowrd** and **icolrd** store the sparsity structure (pattern) of the first derivative matrix as **nnzrd** nonzeros in coordinate storage (CS) format (see Section 2.1.1 in the F11 Chapter Introduction). The matrix has dimensions $n \times m_r$. **irowrd** specifies one-based row indices and **icolrd** specifies one-based column indices. No particular order of elements is expected, but elements should not repeat and the same order should be used when the first derivative matrix is evaluated for the solver.

Constraints:

$1 \leq \textbf{irowrd}(l) \leq n$, for $l = 1, 2, \ldots,$ **nnzrd**;
$1 \leq \textbf{icolrd}(l) \leq$ **nres**, for $l = 1, 2, \ldots,$ **nnzrd**.

7: **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, the recommended value is -1. **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 $= 1$

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 **e04raf** or it has been corrupted.

ifail $= 2$

The Hessians of nonlinear functions have already been defined, a nonlinear objective cannot be added.

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

ifail $= 3$

The objective function has already been defined.

ifail $= 6$

On entry, **isparse** $= \langle value \rangle$.
Constraint: **isparse** $= 0$ or 1.

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

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

ifail $= 8$

On entry, $i = \langle value \rangle$, **icolrd**$(i) = \langle value \rangle$ and **nres** $= \langle value \rangle$.
Constraint: $1 \leq$ **icolrd**$(i) \leq$ **nres**.

On entry, $i = \langle value \rangle$, **irowrd**$(i) = \langle value \rangle$ and $n = \langle value \rangle$.
Constraint: $1 \leq$ **irowrd**$(i) \leq n$.

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

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

e04rmf is not threaded in any implementation.

9 Further Comments

None.

10 Example

In this example, we demonstrate how to declare a least squares problem through **e04rmf** and solve it with **e04fff** on a very simple example. Here $n = 2$, $m_r = 3$ and the residuals are computed by:

$$\begin{aligned} r_1(x) &= x(1) + x(2) - 0.9 \\ r_2(x) &= 2x(1) + x(2) - 1.9 \\ r_3(x) &= 3x(1) + x(2) - 3.0 \end{aligned}$$

The expected result is:

$$x = (0.95, 0.10)$$

with an objective value of 0.015.

10.1 Program Text

```
!   e04rmf Example Program Text

!   Mark 26.1 Release. NAG Copyright 2017.

    Module e04rmfe_mod

!     .. Use Statements ..
      Use iso_c_binding, Only: c_ptr
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Accessibility Statements ..
      Private
      Public                           :: objfun

    Contains

      Subroutine objfun(nvar,x,nres,rx,inform,iuser,ruser,cpuser)

!       .. Scalar Arguments ..
        Type (c_ptr), Intent (In)      :: cpuser
        Integer, Intent (Inout)        :: inform
        Integer, Intent (In)           :: nres, nvar
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Inout) :: ruser(*)
        Real (Kind=nag_wp), Intent (Out) :: rx(nres)
        Real (Kind=nag_wp), Intent (In) :: x(nvar)
        Integer, Intent (Inout)        :: iuser(*)
!       .. Executable Statements ..

!       Interrupt solver if the dimensions are incorrect
        If (nres/=3 .Or. nvar/=2) Then
          inform = -1
          Go To 100
        End If

        rx(1) = x(1) + x(2) - 1.1_nag_wp
        rx(2) = 2.0_nag_wp*x(1) + x(2) - 1.9_nag_wp
        rx(3) = 3.0_nag_wp*x(1) + x(2) - 3.0_nag_wp

100     Continue
        Return
```

```
      End Subroutine objfun

    End Module e04rmfe_mod


    Program e04rmfe

!     .. Use Statements ..
      Use e04rmfe_mod, Only: objfun
      Use iso_c_binding, Only: c_null_ptr, c_ptr
      Use nag_library, Only: e04fff, e04ffu, e04raf, e04rmf, e04rzf, e04zmf,   &
                             nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nout = 6
!     .. Local Scalars ..
      Type (c_ptr)                     :: cpuser, handle
      Integer                          :: ifail, isparse, nnzrd, nres, nvar
!     .. Local Arrays ..
      Real (Kind=nag_wp)               :: rinfo(100), ruser(1), stats(100)
      Real (Kind=nag_wp), Allocatable  :: rx(:), x(:)
      Integer                          :: icolrd(6), irowrd(6), iuser(1)
!     .. Executable Statements ..

      Write (nout,*) 'E04RMF Example Program Results'
      Write (nout,*)
      Flush (nout)

      nvar = 2
      nres = 3
      handle = c_null_ptr

!     Initialize handle
      ifail = 0
      Call e04raf(handle,nvar,ifail)

!     Define residuals structure with e04rmf
      isparse = 1
      nnzrd = 6
      icolrd(1:6) = (/1,1,2,2,3,3/)
      irowrd(1:6) = (/1,2,1,2,1,2/)
      Call e04rmf(handle,nres,isparse,nnzrd,irowrd,icolrd,ifail)

!     Set options for the e04fff solver
!     relax the main convergence criteria a bit
      Call e04zmf(handle,'DFLS Trust Region Tolerance = 1.0e-03',ifail)
!     Deactivate the slow iterations detection
      Call e04zmf(handle,'DFLS Maximum Slow Steps = 0',ifail)
!     Turn off option printing
      Call e04zmf(handle,'Print Options = NO',ifail)
!     Print the solution
      Call e04zmf(handle,'Print Solution = YES',ifail)
!     Deactivate iteration log
      Call e04zmf(handle,'Print Level = 1',ifail)

!     Define starting point
      Allocate (x(nvar),rx(nres))
      x(1:2) = (/2.0_nag_wp,2.0_nag_wp/)

!     Call the solver
      ifail = -1
      cpuser = c_null_ptr
      Call e04fff(handle,objfun,e04ffu,nvar,x,nres,rx,rinfo,stats,iuser,ruser, &
        cpuser,ifail)
```

```
!     Free the handle memory
      ifail = 0
      Call e04rzf(handle,ifail)

    End Program e04rmfe
```

10.2 Program Data

None.

10.3 Program Results

```
 E04RMF Example Program Results

  ---------------------------------------------
  E04FF, Derivative free solver for data fitting
         (nonlinear least-squares problems)
  ---------------------------------------------

  Status: Converged, small trust region size.

  Value of the objective                    1.50000E-02
  Computed Solution:
    idx   Lower bound        Value      Upper bound
      1        -inf        9.50000E-01         inf
      2        -inf        1.00000E-01         inf
```