NAG Library Routine Document

E04JCF

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

E04JCF is an easy-to-use algorithm that uses methods of quadratic approximation to find a minimum of an objective function F over $\mathbf{x} \in R^n$, subject to fixed lower and upper bounds on the independent variables $x_1, x_2, \ldots, x_n$. Derivatives of F are not required.

The routine is intended for functions that are continuous and that have continuous first and second derivatives (although it will usually work even if the derivatives have occasional discontinuities). Efficiency is maintained for large n.

2 Specification

```
SUBROUTINE E04JCF (OBJFUN, N, NPT, X, BL, BU, RHOBEG, RHOEND, MONFUN,      &
                   MAXCAL, F, NF, IUSER, RUSER, IFAIL)

INTEGER             N, NPT, MAXCAL, NF, IUSER(*), IFAIL
REAL (KIND=nag_wp) X(N), BL(N), BU(N), RHOBEG, RHOEND, F, RUSER(*)
EXTERNAL            OBJFUN, MONFUN
```

3 Description

E04JCF is applicable to problems of the form:

$$\underset{\mathbf{x}\in R^n}{\text{minimize}} F(\mathbf{x}) \quad \text{subject to} \quad \boldsymbol{\ell} \leq \mathbf{x} \leq \mathbf{u} \quad \text{and} \quad \boldsymbol{\ell} \leq \mathbf{u},$$

where F is a nonlinear scalar function whose derivatives may be unavailable, and where the bound vectors are elements of R^n. Relational operators between vectors are interpreted elementwise.

Fixing variables (that is, setting $\ell_i = u_i$ for some i) is allowed in E04JCF.

You must supply a subroutine to calculate the value of F at any given point $\mathbf{x}$.

The method used by E04JCF is based on BOBYQA, the method of Bound Optimization BY Quadratic Approximation described in Powell (2009). In particular, each iteration of E04JCF generates a quadratic approximation Q to F that agrees with F at m automatically chosen interpolation points. The value of m is a constant prescribed by you. Updates to the independent variables mostly occur from approximate solutions to trust-region subproblems, using the current quadratic model.

4 References

Powell M J D (2009) The BOBYQA algorithm for bound constrained optimization without derivatives *Report DAMTP 2009/NA06* University of Cambridge http://www.damtp.cam.ac.uk/user/na/NA_papers/NA2009_06.pdf

5 Arguments

1: OBJFUN – SUBROUTINE, supplied by the user. *External Procedure*

OBJFUN must evaluate the objective function F at a specified vector $\mathbf{x}$.

The specification of OBJFUN is:

```
SUBROUTINE OBJFUN (N, X, F, IUSER, RUSER, INFORM)
```

```
INTEGER            N, IUSER(*), INFORM
REAL (KIND=nag_wp) X(N), F, RUSER(*)
```

1: N – INTEGER *Input*

On entry: n, the number of independent variables.

2: X(N) – REAL (KIND=nag_wp) array *Input*

On entry: **x**, the vector at which the objective function is to be evaluated.

3: F – REAL (KIND=nag_wp) *Output*

On exit: must be set to the value of the objective function at **x**, unless you have specified termination of the current problem using INFORM.

4: IUSER(∗) – INTEGER array *User Workspace*

5: RUSER(∗) – REAL (KIND=nag_wp) array *User Workspace*

OBJFUN is called with the arguments IUSER and RUSER as supplied to E04JCF. You should use the arrays IUSER and RUSER to supply information to OBJFUN.

6: INFORM – INTEGER *Output*

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

OBJFUN must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub) program from which E04JCF is called. Arguments denoted as *Input* must **not** be changed by this procedure.

2: N – INTEGER *Input*

On entry: n, the number of independent variables.

Constraint: N ≥ 2 and $n_r \geq 2$, where n_r denotes the number of non-fixed variables.

3: NPT – INTEGER *Input*

On entry: m, the number of interpolation conditions imposed on the quadratic approximation at each iteration.

Suggested value: NPT $= 2 \times n_r + 1$, where n_r denotes the number of non-fixed variables.

Constraint: $n_r + 2 \leq$ NPT $\leq \frac{(n_r+1)\times(n_r+2)}{2}$, where n_r denotes the number of non-fixed variables.

4: X(N) – REAL (KIND=nag_wp) array *Input/Output*

On entry: an estimate of the position of the minimum. If any component is out-of-bounds it is replaced internally by the bound it violates.

On exit: the lowest point found during the calculations. Thus, if IFAIL $= 0$ on exit, X is the position of the minimum.

5: BL(N) – REAL (KIND=nag_wp) array *Input*

6: BU(N) – REAL (KIND=nag_wp) array *Input*

On entry: the fixed vectors of bounds: the lower bounds $\boldsymbol{\ell}$ and the upper bounds **u**, respectively. To signify that a variable is unbounded you should choose a large scalar r appropriate to your problem, then set the lower bound on that variable to $-r$ and the upper bound to r. For well-scaled problems $r = r_{\max}^{\frac{1}{4}}$ may be suitable, where $r_{\max}$ denotes the largest positive model number (see X02ALF).

Constraints:

if X(i) is to be fixed at BL(i), then BL(i) = BU(i);
otherwise BU(i) − BL(i) ≥ 2.0 × RHOBEG, for $i = 1, 2, \ldots,$ N.

7: RHOBEG – REAL (KIND=nag_wp) *Input*

On entry: an initial lower bound on the value of the trust-region radius.

Suggested value: RHOBEG should be about one tenth of the greatest expected overall change to a variable: the initial quadratic model will be constructed by taking steps from the initial X of length RHOBEG along each coordinate direction.

Constraints:

RHOBEG > 0.0;
RHOBEG ≥ RHOEND.

8: RHOEND – REAL (KIND=nag_wp) *Input*

On entry: a final lower bound on the value of the trust-region radius.

Suggested value: RHOEND should indicate the absolute accuracy that is required in the final values of the variables.

Constraint: RHOEND ≥ $macheps$, where $macheps$ = X02AJF(), the ***machine precision***..

9: MONFUN – SUBROUTINE, supplied by the NAG Library or the user. *External Procedure*

MONFUN may be used to monitor the optimization process. It is invoked every time a new trust-region radius is chosen.

If no monitoring is required, MONFUN may be the dummy monitoring routine E04JCP supplied by the NAG Library.

The specification of MONFUN is:

```
SUBROUTINE MONFUN (N, NF, X, F, RHO, IUSER, RUSER, INFORM)
INTEGER             N, NF, IUSER(*), INFORM
REAL (KIND=nag_wp) X(N), F, RHO, RUSER(*)
```

1: N – INTEGER *Input*

On entry: n, the number of independent variables.

2: NF – INTEGER *Input*

On entry: the cumulative number of calls made to OBJFUN.

3: X(N) – REAL (KIND=nag_wp) array *Input*

On entry: the current best point.

4: F – REAL (KIND=nag_wp) *Input*

On entry: the value of OBJFUN at X.

5: RHO – REAL (KIND=nag_wp) *Input*

On entry: a lower bound on the current trust-region radius.

6: IUSER(∗) – INTEGER array *User Workspace*
7: RUSER(∗) – REAL (KIND=nag_wp) array *User Workspace*

MONFUN is called with the arguments IUSER and RUSER as supplied to E04JCF. You should use the arrays IUSER and RUSER to supply information to MONFUN.

8: INFORM – INTEGER *Output*

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

MONFUN must either be a module subprogram USEd by, or declared as EXTERNAL in, the (sub)program from which E04JCF is called. Arguments denoted as *Input* must **not** be changed by this procedure.

10: MAXCAL – INTEGER *Input*

On entry: the maximum permitted number of calls to OBJFUN.

Constraint: MAXCAL ≥ 1.

11: F – REAL (KIND=nag_wp) *Output*

On exit: the function value at the lowest point found (X).

12: NF – INTEGER *Output*

On exit: unless IFAIL $= 1$ or -999 on exit, the total number of calls made to OBJFUN.

13: IUSER$(*)$ – INTEGER array *User Workspace*

14: RUSER$(*)$ – REAL (KIND=nag_wp) array *User Workspace*

IUSER and RUSER are not used by E04JCF, but are passed directly to OBJFUN and MONFUN and should be used to pass information to these routines.

15: 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).

E04JCF returns with IFAIL $= 0$ if the final trust-region radius has reached its lower bound RHOEND.

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 $= 1$

On entry, MAXCAL $= \langle value \rangle$.
Constraint: MAXCAL ≥ 1.

On entry, RHOBEG $= \langle value \rangle$, BL$(i) = \langle value \rangle$, BU$(i) = \langle value \rangle$ and $i = \langle value \rangle$.
Constraint: if BL$(i) \neq$ BU(i) in coordinate i, then BU$(i) -$ BL$(i) \geq 2 \times$ RHOBEG.

On entry, RHOBEG $= \langle value \rangle$.
Constraint: RHOBEG > 0.0.

On entry, RHOBEG $= \langle value \rangle$ and RHOEND $= \langle value \rangle$.
Constraint: RHOEND $\leq$ RHOBEG.

On entry, RHOEND $= \langle value \rangle$.
Constraint: RHOEND $\geq macheps$, where $macheps =$ X02AJF(), the ***machine precision***.

There were $n_r = \langle value \rangle$ unequal bounds.
Constraint: $n_r \geq 2$.

There were $n_r = \langle value \rangle$ unequal bounds and NPT $= \langle value \rangle$ on entry.
Constraint: $n_r + 2 \leq$ NPT $\leq \frac{(n_r+1)\times(n_r+2)}{2}$.

IFAIL $= 2$

The function evaluations limit was reached: OBJFUN has been called MAXCAL times.

IFAIL $= 3$

The predicted reduction in a trust-region step was non-positive. Check your specification of OBJFUN and whether the function needs rescaling. Try a different initial X.

IFAIL $= 4$

A rescue procedure has been called in order to correct damage from rounding errors when computing an update to a quadratic approximation of F, but no further progess could be made. Check your specification of OBJFUN and whether the function needs rescaling. Try a different initial X.

IFAIL $= 5$

User-supplied monitoring routine requested termination.

User-supplied objective function requested termination.

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

Experience shows that, in many cases, on successful termination the ∞-norm distance from the best point **x** to a local minimum of F is less than $10 \times$ RHOEND, unless RHOEND is so small that such accuracy is unattainable.

8 Parallelism and Performance

E04JCF 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 routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

For each invocation of E04JCF, local workspace arrays of fixed length are allocated internally. The total size of these arrays amounts to $(\mathrm{NPT}+6)\times(\mathrm{NPT}+n_r)+\frac{n_r\times(3n_r+21)}{2}$ real elements and n_r integer elements, where n_r denotes the number of non-fixed variables; that is, the total size is $\mathcal{O}\left(n_r^4\right)$. If you follow the recommendation for the choice of NPT on entry, this total size reduces to $\mathcal{O}\left(n_r^2\right)$.

Usually the total number of function evaluations (NF) is substantially less than $\mathcal{O}\left(n_r^2\right)$, and often, if $\mathrm{NPT}=2\times n_r+1$ on entry, NF is only of magnitude n_r or less.

10 Example

This example involves the minimization of

$$F=(x_1+10x_2)^2+5(x_3-x_4)^2+(x_2-2x_3)^4+10(x_1-x_4)^4$$

subject to

$$\begin{aligned}1\leq x_1\leq 3\\-2\leq x_2\leq 0\\1\leq x_4\leq 3,\end{aligned}$$

starting from the initial guess $(3,-1,0,1)$.

10.1 Program Text

```
!   E04JCF Example Program Text
!   Mark 26 Release. NAG Copyright 2016.
    Module e04jcfe_mod

!     E04JCF Example Program Module:
!            Parameters and User-defined Routines

!     .. Use Statements ..
      Use nag_library, Only: nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Accessibility Statements ..
      Private
      Public                           :: monfun, objfun
!     .. Parameters ..
      Integer, Parameter, Public       :: nout = 6
    Contains
      Subroutine objfun(n,x,f,iuser,ruser,inform)

!       .. Parameters ..
        Real (Kind=nag_wp), Parameter  :: five = 5.0_nag_wp
        Real (Kind=nag_wp), Parameter  :: ten = 1.0E1_nag_wp
        Real (Kind=nag_wp), Parameter  :: two = 2.0_nag_wp
!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (Out) :: f
        Integer, Intent (Out)          :: inform
        Integer, Intent (In)           :: n
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Inout) :: ruser(*)
        Real (Kind=nag_wp), Intent (In) :: x(n)
        Integer, Intent (Inout)        :: iuser(*)
!       .. Executable Statements ..
        inform = 0

        f = (x(1)+ten*x(2))**2 + five*(x(3)-x(4))**2 + (x(2)-two*x(3))**4 +    &
```

```
          ten*(x(1)-x(4))**4

        Return

      End Subroutine objfun
      Subroutine monfun(n,nf,x,f,rho,iuser,ruser,inform)

!       .. Scalar Arguments ..
        Real (Kind=nag_wp), Intent (In) :: f, rho
        Integer, Intent (Out)          :: inform
        Integer, Intent (In)           :: n, nf
!       .. Array Arguments ..
        Real (Kind=nag_wp), Intent (Inout) :: ruser(*)
        Real (Kind=nag_wp), Intent (In) :: x(n)
        Integer, Intent (Inout)        :: iuser(*)
!       .. Local Scalars ..
        Logical                        :: verbose_output
!       .. Executable Statements ..
        inform = 0

        Write (nout,Fmt=99999) 'Monitoring: new trust region radius =', rho

!       Set this to .True. to get more detailed output
        verbose_output = .False.

        If (verbose_output) Then
          Write (nout,Fmt=99998) 'Number of function calls =', nf
          Write (nout,Fmt=99997) 'Current function value =', f
          Write (nout,Fmt=99996) 'The corresponding X is:', x(1:n)
        End If

        Return
99999   Format (/,4X,A,1P,E13.3)
99998   Format (4X,A,I16)
99997   Format (4X,A,1P,E12.4)
99996   Format (4X,A,/,(4X,5E12.4))
      End Subroutine monfun
    End Module e04jcfe_mod
    Program e04jcfe

!     Example problem for E04JCF.

!     .. Use Statements ..
      Use nag_library, Only: e04jcf, nag_wp, x02alf
      Use e04jcfe_mod, Only: monfun, nout, objfun
!     .. Implicit None Statement ..
      Implicit None
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: f, infbnd, rhobeg, rhoend
      Integer                          :: ifail, maxcal, n, nf, npt
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: bl(:), bu(:), x(:)
      Real (Kind=nag_wp)               :: ruser(1)
      Integer                          :: iuser(1)
!     .. Executable Statements ..
      Write (nout,*) 'E04JCF Example Program Results'

      maxcal = 500
      rhobeg = 1.0E-1_nag_wp
      rhoend = 1.0E-6_nag_wp
      n = 4
      npt = 2*n + 1

!     x(3) is unconstrained, so we're going to set bl(3) to a large
!     negative number and bu(3) to a large positive number.

      infbnd = x02alf()**0.25_nag_wp

      Allocate (bl(n),bu(n),x(n))

      bl(1:n) = (/1.0_nag_wp,-2.0_nag_wp,-infbnd,1.0_nag_wp/)
```

```
      bu(1:n) = (/3.0_nag_wp,0.0_nag_wp,infbnd,3.0_nag_wp/)
      x(1:n) = (/3.0_nag_wp,-1.0_nag_wp,0.0_nag_wp,1.0_nag_wp/)

      ifail = -1
      Call e04jcf(objfun,n,npt,x,bl,bu,rhobeg,rhoend,monfun,maxcal,f,nf,iuser, &
        ruser,ifail)

      Select Case (ifail)
      Case (0,2:5)

        If (ifail==0) Then
          Write (nout,Fmt=99999) 'Successful exit from E04JCF.',               &
            'Function value at lowest point found =', f
        Else
          Write (nout,Fmt=99998)                                               &
            'On exit from E04JCF, function value at lowest point found =', f
        End If

        Write (nout,Fmt=99997) 'The corresponding X is:', x(1:n)
      End Select

99999 Format (2(/,1X,A),1P,E13.3)
99998 Format (/,1X,A,1P,E13.3)
99997 Format (1X,A,/,(2X,5E13.3))
    End Program e04jcfe
```

10.2 Program Data

None.

10.3 Program Results

```
 E04JCF Example Program Results

    Monitoring: new trust region radius =  1.000E-02

    Monitoring: new trust region radius =  1.000E-03

    Monitoring: new trust region radius =  1.000E-04

    Monitoring: new trust region radius =  1.000E-05

    Monitoring: new trust region radius =  1.000E-06

 Successful exit from E04JCF.
 Function value at lowest point found =    2.434E+00
 The corresponding X is:
      0.100E+01   -0.852E-01    0.409E+00    0.100E+01
```