

NAG Library Routine Document

E04UQF/E04UQA

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

To supply optional parameters to E04USF/E04USA from an external file. More precisely, E04UQF must be used to supply optional parameters to E04USF and E04UQA must be used to supply optional parameters to E04USA.

E04UQA is a version of E04UQF that has additional parameters in order to make it safe for use in multithreaded applications (see Section 5). The initialization routine E04WBF **must** have been called before calling E04UQA.

2 Specification

2.1 Specification for E04UQF

```
SUBROUTINE E04UQF (IOPTNS, INFORM)
```

```
INTEGER IOPTNS, INFORM
```

2.2 Specification for E04UQA

```
SUBROUTINE E04UQA (IOPTNS, LWSAV, IWSAV, RWSAV, INFORM)
```

```
INTEGER IOPTNS, IWSAV(610), INFORM
```

```
REAL (KIND=nag_wp) RWSAV(475)
```

```
LOGICAL LWSAV(120)
```

3 Description

E04UQF/E04UQA may be used to supply values for optional parameters to the corresponding routines E04USF/E04USA. E04UQF/E04UQA reads an external file and each line of the file defines a single optional parameter. It is only necessary to supply values for those parameters whose values are to be different from their default values.

Each optional parameter is defined by a single character string, of up to 72 characters, consisting of one or more items. The items associated with a given option must be separated by spaces, or equals signs [=]. Alphabetic characters may be upper or lower case. The string

```
Print Level = 1
```

is an example of a string used to set an optional parameter. For each option the string contains one or more of the following items:

- a mandatory keyword;
- a phrase that qualifies the keyword;
- a number that specifies an integer or real value. Such numbers may be up to 16 contiguous characters in Fortran's I, F, E or D formats, terminated by a space if this is not the last item on the line.

Blank strings and comments are ignored. A comment begins with an asterisk (*) and all subsequent characters in the string are regarded as part of the comment.

The file containing the options must start with `Begin` and must finish with `End`. An example of a valid options file is:

```
Begin * Example options file
  Print level = 5
End
```

For E04UQF each line of the file is normally printed as it is read, on the current advisory message unit (see X04ABF), but printing may be suppressed using the keyword **Nolist**. To suppress printing of `Begin`, **Nolist** must be the first option supplied as in the file:

```
Begin
  Nolist
  Print level = 5
End
```

Printing will automatically be turned on again after a call to E04USF/E04USA or E04UQF and may be turned on again at any time using the keyword **List**.

For E04UQA printing is turned off by default, but may be turned on at any time using the keyword **List**.

Optional parameter settings are preserved following a call to E04USF/E04USA and so the keyword **Defaults** is provided to allow you to reset all the optional parameters to their default values before a subsequent call to E04USF/E04USA.

A complete list of optional parameters, their abbreviations, synonyms and default values is given in Section 11 in E04USF/E04USA.

4 References

None.

5 Parameters

1: IOPTNS – INTEGER *Input*

On entry: the unit number of the options file to be read.

Constraint: $0 \leq \text{IOPTNS} \leq 99$.

2: INFORM – INTEGER *Output*

Note: for E04UQA, *INFORM* does not occur in this position in the parameter list. See the additional parameters described below.

On exit: contains zero if the options file has been successfully read and a value > 0 otherwise (see Section 6).

Note: the following are additional parameters for specific use with E04UQA. Users of E04UQF therefore need not read the remainder of this description.

2: LWSAV(120) – LOGICAL array *Communication Array*

3: IWSAV(610) – INTEGER array *Communication Array*

4: RWSAV(475) – REAL (KIND=nag_wp) array *Communication Array*

The arrays LWSAV, IWSAV and RWSAV **must not** be altered between calls to any of the routines E04UQA, E04URA, E04USA or E04WBF.

5: INFORM – INTEGER *Output*

Note: see the parameter description for *INFORM* above.

6 Error Indicators and Warnings

Errors or warnings detected by the routine:

INFORM = 1

IOPTNS is not in the range [0,99].

INFORM = 2

Begin was found, but end-of-file was found before End was found.

INFORM = 3

end-of-file was found before Begin was found.

INFORM = 4

Not used.

INFORM = 5

One or more lines of the options file is invalid. Check that all keywords are neither ambiguous nor misspelt.

7 Accuracy

Not applicable.

8 Further Comments

E04URF/E04URA may also be used to supply optional parameters to E04USF/E04USA.

9 Example

This example solves the same problem as the example for E04USF/E04USA, but in addition illustrates the use of E04UQF/E04UQA and E04URF/E04URA to set optional parameters for E04USF/E04USA.

In this example the options file read by E04UQF/E04UQA is appended to the data file for the program (see Section 9.2). It would usually be more convenient in practice to keep the data file and the options file separate.

9.1 Program Text

the following program illustrates the use of E04UQF. An equivalent program illustrating the use of E04UQA is available with the supplied Library and is also available from the NAG web site.

```
! E04UQF Example Program Text
! Mark 24 Release. NAG Copyright 2012.

Module e04uqfe_mod

! E04UQF Example Program Module:
! Parameters and User-defined Routines

! .. Use Statements ..
Use nag_library, Only: nag_wp
! .. Implicit None Statement ..
Implicit None
! .. Parameters ..
Integer, Parameter :: iset = 1, nin = 5, ninopt = 7, &
nout = 6

Contains
Subroutine objfun(mode,m,n,ldfj,needfi,x,f,fjac,nstate,iuser,ruser)
! Routine to evaluate the subfunctions and their 1st derivatives.
```

```

! .. Parameters ..
Real (Kind=nag_wp), Parameter      ::
    a(44) = (/8.0E0_nag_wp,8.0E0_nag_wp, &
    10.0E0_nag_wp,10.0E0_nag_wp,10.0E0_nag_wp,10.0E0_nag_wp, &
    12.0E0_nag_wp,12.0E0_nag_wp,12.0E0_nag_wp,12.0E0_nag_wp, &
    14.0E0_nag_wp,14.0E0_nag_wp,14.0E0_nag_wp,16.0E0_nag_wp, &
    16.0E0_nag_wp,16.0E0_nag_wp,18.0E0_nag_wp,18.0E0_nag_wp, &
    20.0E0_nag_wp,20.0E0_nag_wp,20.0E0_nag_wp,22.0E0_nag_wp, &
    22.0E0_nag_wp,22.0E0_nag_wp,24.0E0_nag_wp,24.0E0_nag_wp, &
    24.0E0_nag_wp,26.0E0_nag_wp,26.0E0_nag_wp,26.0E0_nag_wp, &
    28.0E0_nag_wp,28.0E0_nag_wp,30.0E0_nag_wp,30.0E0_nag_wp, &
    30.0E0_nag_wp,32.0E0_nag_wp,32.0E0_nag_wp,34.0E0_nag_wp, &
    36.0E0_nag_wp,36.0E0_nag_wp,38.0E0_nag_wp,38.0E0_nag_wp, &
    40.0E0_nag_wp,42.0E0_nag_wp/)

! .. Scalar Arguments ..
Integer, Intent (In)                :: ldfj, m, n, needfi, nstate
Integer, Intent (Inout)             :: mode

! .. Array Arguments ..
Real (Kind=nag_wp), Intent (Out)    :: f(m)
Real (Kind=nag_wp), Intent (Inout)  :: fjac(ldfj,n), ruser(*)
Real (Kind=nag_wp), Intent (In)     :: x(n)
Integer, Intent (Inout)             :: iuser(*)

! .. Local Scalars ..
Real (Kind=nag_wp)                 :: ai, temp, x1, x2
Integer                             :: i
Logical                             :: mode02, mode12

! .. Intrinsic Procedures ..
Intrinsic                          :: exp

! .. Executable Statements ..
x1 = x(1)
x2 = x(2)

mode02 = (mode==0 .Or. mode==2)
mode12 = (mode==1 .Or. mode==2)

loop: Do i = 1, m

    If (needfi==i) Then
        f(i) = x1 + (0.49E0_nag_wp-x1)*exp(-x2*(a(i)-8.0E0_nag_wp))
        Exit loop
    End If

    ai = a(i)
    temp = exp(-x2*(ai-8.0E0_nag_wp))

    If (mode02) Then
        f(i) = x1 + (0.49E0_nag_wp-x1)*temp
    End If

    If (mode12) Then
        fjac(i,1) = 1.0E0_nag_wp - temp
        fjac(i,2) = -(0.49E0_nag_wp-x1)*(ai-8.0E0_nag_wp)*temp
    End If

End Do loop

Return

End Subroutine objfun
Subroutine confun(mode,ncnln,n,ldcj,needc,x,c,cjac,nstate,iuser,ruser)
! Routine to evaluate the nonlinear constraint and its 1st
! derivatives.

! .. Scalar Arguments ..
Integer, Intent (In)                :: ldcj, n, ncnln, nstate
Integer, Intent (Inout)             :: mode

! .. Array Arguments ..
Real (Kind=nag_wp), Intent (Out)    :: c(ncnln)
Real (Kind=nag_wp), Intent (Inout)  :: cjac(ldcj,n), ruser(*)
Real (Kind=nag_wp), Intent (In)     :: x(n)

```

```

Integer, Intent (Inout)          :: iuser(*)
Integer, Intent (In)             :: needc(ncnln)
! .. Executable Statements ..
If (nstate==1) Then

!     First call to CONFUN. Set all Jacobian elements to zero.
!     Note that this will only work when 'Derivative Level = 3'
!     (the default; see Section 11.2).

    cjac(1:ncnln,1:n) = 0.0E0_nag_wp
End If

If (needc(1)>0) Then

    If (mode==0 .Or. mode==2) Then
        c(1) = -0.09E0_nag_wp - x(1)*x(2) + 0.49E0_nag_wp*x(2)
    End If

    If (mode==1 .Or. mode==2) Then
        cjac(1,1) = -x(2)
        cjac(1,2) = -x(1) + 0.49E0_nag_wp
    End If

End If

Return

End Subroutine confun
End Module e04uqfe_mod
Program e04uqfe

! E04UQF Example Main Program

! .. Use Statements ..
Use nag_library, Only: e04uqf, e04urf, e04usf, nag_wp, x04abf, x04acf, &
    x04baf
Use e04uqfe_mod, Only: confun, iset, nin, ninopt, nout, objfun
! .. Implicit None Statement ..
Implicit None
! .. Parameters ..
Character (*), Parameter          :: fname = 'e04uqfe.opt'
! .. Local Scalars ..
Real (Kind=nag_wp)               :: objf
Integer                           :: i, ifail, inform, iter, lda, &
    ldcj, ldfj, ldr, liwork, lwork, &
    m, mode, n, nclin, ncnln, &
    outchn, sda, sdcjac
Character (80)                    :: rec
! .. Local Arrays ..
Real (Kind=nag_wp), Allocatable  :: a(:,,:), bl(:), bu(:), c(:), &
    cjac(:,,:), clamda(:), f(:), &
    fjac(:,,:), r(:,,:), work(:), &
    x(:), y(:)
Real (Kind=nag_wp)               :: user(1)
Integer, Allocatable             :: istate(:), iwork(:)
Integer                           :: iuser(1)
! .. Intrinsic Procedures ..
Intrinsic                        :: max
! .. Executable Statements ..
Write (rec,99998) 'E04UQF Example Program Results'
Call x04baf(nout,rec)

! Skip heading in data file
Read (nin,*)

Read (nin,*) m, n
Read (nin,*) nclin, ncnln
liwork = 3*n + nclin + 2*ncnln
lda = max(1,nclin)

If (nclin>0) Then

```

```

      sda = n
    Else
      sda = 1
    End If

    ldcj = max(1,ncnln)

    If (ncnln>0) Then
      sdcjac = n
    Else
      sdcjac = 1
    End If

    ldfj = m
    ldr = n

    If (ncnln==0 .And. nclin>0) Then
      lwork = 2*n**2 + 20*n + 11*nclin + m*(n+3)
    Else If (ncnln>0 .And. nclin>=0) Then
      lwork = 2*n**2 + n*nclin + 2*n*ncnln + 20*n + 11*nclin + 21*ncnln + &
        m*(n+3)
    Else
      lwork = 20*n + m*(n+3)
    End If

    Allocate (istate(n+nclin+ncnln),iwork(liwork),a(lda,sda), &
      bl(n+nclin+ncnln),bu(n+nclin+ncnln),y(m),c(max(1, &
      ncnln)),cjac(ldcj,sdcjac),f(m),fjac(ldfj,n),clamda(n+nclin+ncnln), &
      r(ldr,n),x(n),work(lwork))

    If (nclin>0) Then
      Read (nin,*)(a(i,1:sda),i=1,nclin)
    End If

    Read (nin,*) y(1:m)
    Read (nin,*) bl(1:(n+nclin+ncnln))
    Read (nin,*) bu(1:(n+nclin+ncnln))
    Read (nin,*) x(1:n)

!   Set the unit number for advisory messages to OUTCHN

    outchn = nout
    Call x04abf(iset,outchn)

!   Set three options using E04URF

    Call e04urf(' Infinite Bound Size = 1.0D+25 ')
    Call e04urf(' Print Level = 1 ')
    Call e04urf(' Verify Level = -1 ')

!   Open the options file for reading

    mode = 0

    ifail = 0
    Call x04acf(ninopt,fname,mode,ifail)

!   Read the options file for the remaining options

    Call e04uqf(ninopt,inform)

    If (inform/=0) Then
      Write (rec,99999) 'E04UQF terminated with INFORM = ', inform
      Call x04baf(nout,rec)
      Go To 100
    End If

!   Solve the problem

```

```

ifail = 0
Call e04usf(m,n,ncnln,ncnln,lda,ldcj,ldfj,ldr,a,bl,bu,y,confun,objfun, &
  iter,istate,c,cjac,f,fjac,clamda,objf,r,x,iwork,liwork,work,lwork, &
  iuser,user,ifail)

100 Continue

99999 Format (1X,A,I5)
99998 Format (1X,A)
End Program e04uqfe

```

9.2 Program Data

```

Begin Example options file for E04UQF
  Major Iteration Limit = 15 * (Default = 50)
  Minor Iteration Limit = 10 * (Default = 50)
End

E04UQF Example Program Data
44 2 :Values of M and N
1 1 :Values of NCLIN and NCNLN
1.0 1.0 :End of matrix A
0.49 0.49 0.48 0.47 0.48 0.47 0.46 0.46 0.45 0.43 0.45
0.43 0.43 0.44 0.43 0.43 0.46 0.45 0.42 0.42 0.43 0.41
0.41 0.40 0.42 0.40 0.40 0.41 0.40 0.41 0.41 0.40 0.40
0.40 0.38 0.41 0.40 0.40 0.41 0.38 0.40 0.40 0.39 0.39 :End of Y
0.4 -4.0 1.0 0.0 :End of BL
1.0E+25 1.0E+25 1.0E+25 1.0E+25 :End of BU
0.4 0.0 :End of X

```

9.3 Program Results

E04UQF Example Program Results

Calls to E04URF

```

Infinite Bound Size = 1.0D+25
Print Level = 1
Verify Level = -1

```

OPTIONS file

```

Begin Example options file for E04UQF
  Major Iteration Limit = 15 * (Default = 50)
  Minor Iteration Limit = 10 * (Default = 50)
End

```

*** E04USF

Parameters

Linear constraints.....	1	Variables.....	2
Nonlinear constraints..	1	Subfunctions.....	44
Infinite bound size....	1.00E+25	COLD start.....	
Infinite step size.....	1.00E+25	EPS (machine precision)	1.11E-16
Step limit.....	2.00E+00	Hessian.....	NO
Linear feasibility.....	1.05E-08	Crash tolerance.....	1.00E-02
Nonlinear feasibility..	1.05E-08	Optimality tolerance...	3.26E-12
Line search tolerance..	9.00E-01	Function precision.....	4.37E-15
Derivative level.....	3	Monitoring file.....	-1
Verify level.....	-1		

```

Major iterations limit.      15      Major print level.....      1
Minor iterations limit.     10      Minor print level.....      0

J'J initial Hessian.....      Reset frequency.....      2

Workspace provided is      IWORK(      9), WORK(      306).
To solve problem we need IWORK(      9), WORK(      306).

Exit from NP problem after      6 major iterations,
                               8 minor iterations.

```

Varbl	State	Value	Lower Bound	Upper Bound	Lagr Mult	Slack
V	1	FR	0.419953	0.400000	None	1.9953E-02
V	2	FR	1.28485	-4.00000	None	5.285

L	Con State	Value	Lower Bound	Upper Bound	Lagr Mult	Slack
L	1	FR	1.70480	1.00000	None	0.7048

N	Con State	Value	Lower Bound	Upper Bound	Lagr Mult	Slack
N	1	LL	-9.767742E-13	.	None	3.3358E-02 -9.7677E-13

Exit E04USF - Optimal solution found.

Final objective value = 0.1422983E-01
