NAG Fortran Library Routine Document

E04VJF

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

E04VJF may be used before E04VHF to determine the sparsity pattern for the Jacobian.

2 Specification

```
 SUBROUTINE E04VJF (NF, N, USRFUN, IAFUN, JAVAR, A, LENA, NEA, IGFUN,
1                   JGVAR, LENG, NEG, X, XLOW, XUPP, CW, LENCW, IW,
2                   LENIW, RW, LENRW, CUSER, IUSER, RUSER, IFAIL)

 INTEGER            NF, N, IAFUN(LENA), JAVAR(LENA), LENA, NEA,
1                   IGFUN(LENG), JGVAR(LENG), LENG, NEG, LENCW,
2                   IW(LENIW), LENIW, LENRW, IUSER(*), IFAIL
 double precision   A(LENA), X(N), XLOW(N), XUPP(N), RW(LENRW), RUSER(*)
 CHARACTER*8        CW(LENCW), CUSER(*)
 EXTERNAL           USRFUN
```

3 Description

When using E04VHF, if you set the optional parameter **Derivative Option** to 0 and your function routine USRFUN provides none of the derivatives, you may need to call E04VJF to determine the input arrays IAFUN, JAVAR, A, IGFUN and JGVAR. These arrays define the pattern of non-zeros in the Jacobian matrix. A typical sequence of calls could be

```
CALL E04VGF (CW, LENCW, ... )
CALL E04VJF (NF, N, ... )
CALL E04VLF ('Derivative Option' = 0, CW, ... )
CALL E04VHF (START, NF, ... )
```

E04VJF determines the sparsity pattern for the Jacobian and identifies the constant elements automatically. To do so, E04VJF approximates the problem functions, $F(x)$, at three random perturbations of the given initial point x. If an element of the approximate Jacobian is the same at all three points, then it is taken to be constant. If it is zero, it is taken to be identically zero. Since the random points are not chosen close together, the heuristic will correctly classify the Jacobian elements in the vast majority of cases. In general, E04VJF finds that the Jacobian can be permuted to the form:

$$\begin{pmatrix} G(x) & A_3 \\ A_2 & A_4 \end{pmatrix},$$

where A_2, A_3 and A_4 are constant. Note that $G(x)$ might contain elements that are also constant, but E04VJF must classify them as nonlinear. This is because E04VHF 'removes' linear variables from the calculation of F by setting them to zero before calling USRFUN. A knowledgeable user would be able to move such elements from $F(x)$ in USRFUN and enter them as part of IAFUN, JAVAR and A for E04VHF.

4 References

Hock W and Schittkowski K (1981) *Test Examples for Nonlinear Programming Codes. Lecture Notes in Economics and Mathematical Systems* **187** Springer–Verlag

5 Parameters

Note: all optional parameters are described in detail in Section 11.2 of the document for E04VHF.

1: NF – INTEGER *Input*

On entry: nf, the number of problem functions in $F(x)$, including the objective function (if any) and the linear and nonlinear constraints. Simple upper and lower bounds on x can be defined using the parameters XLOW and XUPP defined below and should not be included in F.

Constraint: NF > 0.

2: N – INTEGER *Input*

On entry: n, the number of variables.

Constraint: N > 0.

3: USRFUN – SUBROUTINE, supplied by the user. *External Procedure*

USRFUN must define the problem functions $F(x)$. This subroutine is passed to E04VJF as the external parameter USRFUN.

Its specification is:

```
      SUBROUTINE USRFUN (STATUS, N, X, NEEDF, NF, F, NEEDG, LENG, G,
     1                   CUSER, IUSER, RUSER)
      INTEGER            STATUS, N, NEEDF, NF, NEEDG, LENG, IUSER(*)
      double precision   X(N), F(NF), G(LENG), RUSER(*)
      CHARACTER*8        CUSER(*)
```

1: STATUS – INTEGER *Input/Output*

On entry: indicates the first call to USRFUN.

If STATUS $= 0$, there is nothing special about the current call to USRFUN;
if STATUS $= 1$, E04VJF is calling your subroutine for the *first* time. Some data may need to be input or computed and saved.

On exit: may be used to indicate that you are unable to evaluate F at the current x. (For example, the problem functions may not be defined there).

E04VJF evaluates $F(x)$ at random perturbation of the initial point x, say x_p. If the functions cannot be evaluated at x_p, you can set STATUS $= -1$, E04VJF will use another random perturbation.

If for some reason you wish to terminate the current problem, set STATUS ≤ -2.

2: N – INTEGER *Input*

On entry: n, the number of variables, as defined in the call to E04VJF.

3: X(N) – ***double precision*** array *Input*

On entry: the variables x at which the problem functions are to be calculated. *The array x must not be altered.*

4: NEEDF – INTEGER *Input*

On entry: indicates if F must be assigned during the call to USRFUN (see F below).

5: NF – INTEGER *Input*

On entry: nf, the number of problem functions.

6: F(NF) – ***double precision*** array *Input/Output*

On entry: this will be set by E04VJF.

On exit: the computed $F(x)$ according to the setting of NEEDF.

If NEEDF=0, F is not required and is ignored.

If NEEDF > 0, the components of $F(x)$ must be calculated and assigned to F. E04VJF will always call USRFUN with NEEDF > 0.

To simplify the code, you may ignore the value of NEEDF and compute $F(x)$ on every entry to USRFUN.

7: NEEDG – INTEGER *Input*

On entry: E04VJF will call USRFUN with NEEDG $= 0$ to indicate that G is not required.

8: LENG – INTEGER *Input*

On entry: the dimension of the array G as declared in the (sub)program from which E04VJF is called.

9: G(LENG) – ***double precision*** array

E04VJF will always call USRFUN with NEEDG $= 0$. G will not be used but must be declared correctly.

10: CUSER$(*)$ – CHARACTER*8 array *User Workspace*
11: IUSER$(*)$ – INTEGER array *User Workspace*
12: RUSER$(*)$ – ***double precision*** array *User Workspace*

USRFUN is called from E04VJF with the parameters CUSER, IUSER and RUSER as supplied to E04VJF. These parameters are not touched by E04VJF and can be used as an alternative to COMMON.

USRFUN must be declared as EXTERNAL in the (sub)program from which E04VJF is called. Parameters denoted as *Input* must **not** be changed by this procedure.

4: IAFUN(LENA) – INTEGER array *Output*
5: JAVAR(LENA) – INTEGER array *Output*
6: A(LENA) – ***double precision*** array *Output*

On exit: define the coordinates (i, j) and values A_{ij} of the non-zero elements of the linear part A of the function $F(x) = f(x) + Ax$.

In particular, the NEA triples $(\mathrm{IAFUN}(k), \mathrm{JAVAR}(k), \mathrm{A}(k))$ define the row and column indices $i = \mathrm{IAFUN}(k)$ and $j = \mathrm{JAVAR}(k)$ of the element $A_{ij} = \mathrm{A}(k)$.

7: LENA – INTEGER *Input*

On entry: the dimension of the arrays IAFUN, JAVAR and A that hold $\left(i, j, A_{ij}\right)$ as declared in the (sub)program from which E04VJF is called. LENA should be an *overestimate* of the number of elements in the linear part of the Jacobian.

Constraint: LENA ≥ 1.

8: NEA – INTEGER *Output*

On exit: is the number of non-zero entries in A such that $F(x) = f(x) + Ax$.

9: IGFUN(LENG) – INTEGER array *Output*
10: JGVAR(LENG) – INTEGER array *Output*

On exit: define the coordinates (i, j) of the non-zero elements of G, the nonlinear part of the derivatives $J(x) = G(x) + A$ of the function $F(x) = f(x) + Ax$.

11: LENG – INTEGER *Input*

On entry: the dimension of the arrays IGFUN and JGVAR that define the varying Jacobian elements (i, j, G_{ij}) as declared in the (sub)program from which E04VJF is called. LENG should be an *overestimate* of the number of elements in the nonlinear part of the Jacobian.

Constraint: LENG ≥ 1.

12: NEG – INTEGER *Output*

On exit: the number of non-zero entries in G.

13: X(N) – ***double precision*** array *Input*

On entry: an initial estimate of the variables x. The contents of x will be used by E04VJF in the call of USRFUN, and so each element of X should be within the bounds given by XLOW and XUPP.

14: XLOW(N) – ***double precision*** array *Input*
15: XUPP(N) – ***double precision*** array *Input*

On entry: contain the lower and upper bounds l_x and u_x on the variables x.

To specify a non-existent lower bound $[l_x]_j = -\infty$, set XLOW$(j) \leq -bigbnd$, where $bigbnd$ is the **Infinite Bound Size**. To specify a non-existent upper bound XUPP$(j) \geq bigbnd$.

To fix the jth variable (say $x_j = \beta$, where $|\beta| < bigbnd$), set XLOW$(j) =$ XUPP$(j) = \beta$.

16: CW(LENCW) – CHARACTER*8 array *Communication Array*
17: LENCW – INTEGER *Input*

On entry: the dimension of the array CW as declared in the (sub)program from which E04VJF is called.

Constraint: LENCW ≥ 600.

18: IW(LENIW) – INTEGER array *Communication Array*
19: LENIW – INTEGER *Input*

On entry: the dimension of the array IW as declared in the (sub)program from which E04VJF is called.

Constraint: LENIW ≥ 600.

20: RW(LENRW) – ***double precision*** array *Communication Array*
21: LENRW – INTEGER *Input*

On entry: the dimension of the array RW as declared in the (sub)program from which E04VJF is called.

Constraint: LENRW ≥ 600.

22: CUSER$(*)$ – CHARACTER*8 array *User Workspace*
23: IUSER$(*)$ – INTEGER array *User Workspace*
24: RUSER$(*)$ – ***double precision*** array *User Workspace*

Note: the dimension of the array CUSER, IUSER and RUSER must be at least 1.

CUSER, IUSER and RUSER are not used by E04VJF, but are passed directly to USRFUN and may be used to communicate with E04VJF.

25: IFAIL – INTEGER *Input/Output*

On initial entry: IFAIL must be set to 0, -1 or 1. Users who are unfamiliar with this parameter should refer to Chapter P01 for details.

On final exit: IFAIL $= 0$ unless the routine detects an error (see Section 6).

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, because for this routine the values of the output parameters may be useful even if $\mathrm{IFAIL} \neq 0$ on exit, the recommended value is -1. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

6 Error Indicators and Warnings

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

Errors or warnings detected by the routine:

$\mathrm{IFAIL} = 1$

The initialization routine E04VGF has not been called or at least one of LENCW, LENIW or LENRW is less than 600.

$\mathrm{IFAIL} = 2$

An input parameter is invalid. The output message provides more details of the invalid parameter.

$\mathrm{IFAIL} = 3$

Undefined user-supplied function.

The user has indicated that the problem functions are undefined by setting $\mathrm{STATUS} = -1$ on exit from USRFUN. This exit occurs if E04VJF is unable to find a point at which the functions are defined.

$\mathrm{IFAIL} = 4$

User requested termination.

The user has indicated the wish to terminate the call to E04VJF by setting STATUS to a value < -1 on exit from USRFUN.

$\mathrm{IFAIL} = 5$

Either LENA or LENG is too small, resulting in insufficient array to store the Jacobian information. Increase LENA and/or LENG

$\mathrm{IFAIL} = 6$

Unable to estimate the Jacobian structure.

$\mathrm{IFAIL} = 7$

Internal memory allocation failed when attempting to obtain the required workspace. Please contact NAG.

$\mathrm{IFAIL} = 8$

Internal memory allocation insufficient. Please contact NAG.

7 Accuracy

Not applicable.

8 Further Comments

None.

9 Example

This example shows how to call E04VJF to determine the sparsity pattern of the Jacobian before calling E04VJF to solve a sparse nonlinear proramming problem without providing the Jacobian information in USRFUN.

This is a reformulation of Problem 74 from Hock and Schittkowski (1981) and involves the minimization of the nonlinear function

$$f(x) = 10^{-6}x_3^3 + \tfrac{2}{3} \times 10^{-6}x_4^3 + 3x_3 + 2x_4$$

subject to the bounds

$$\begin{aligned} -0.55 \leq x_1 \leq 0.55, \\ -0.55 \leq x_2 \leq 0.55, \\ 0 \leq x_3 \leq 1200, \\ 0 \leq x_4 \leq 1200, \end{aligned}$$

to the nonlinear constraints

$$\begin{array}{lll} 1000\sin(-x_1 - 0.25) & +1000\sin(-x_2 - 0.25) - x_3 & = -894.8, \\ 1000\sin(x_1 - 0.25) & +1000\sin(x_1 - x_2 - 0.25) - x_4 & = -894.8, \\ 1000\sin(x_2 - 0.25) & +1000\sin(x_2 - x_1 - 0.25) & = -1294.8, \end{array}$$

and to the linear constraints

$$\begin{aligned} -x_1 + x_2 \geq -0.55, \\ x_1 - x_2 \geq -0.55. \end{aligned}$$

The initial point, which is infeasible, is

$$x_0 = (\,0, \quad 0, \quad 0, \quad 0\,)^T,$$

and $f(x_0) = 0$.

The optimal solution (to five figures) is

$$x^* = (0.11887, -0.39623, 679.94, 1026.0)^T,$$

and $f(x^*) = 5126.4$. All the nonlinear constraints are active at the solution.

The formulation of the problem combines the constraints and the objective into a single vector (F).

$$F = \begin{pmatrix} 1000\sin(-x_1 - 0.25) & +1000\sin(-x_2 - 0.25) - x_3 \\ 1000\sin(x_1 - 0.25) & +1000\sin(x_1 - x_2 - 0.25) - x_4 \\ 1000\sin(x_2 - 0.25) & +1000\sin(x_2 - x_1 - 0.25) \\ & -x_1 + x_2 \\ & x_1 - x_2 \\ & 10^{-6}x_3^3 + \tfrac{2}{3} \times 10^{-6}x_4^3 + 3x_3 + 2x_4 \end{pmatrix}$$

9.1 Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     E04VJF Example Program Text
*     Mark 21 Release. NAG Copyright 2004.
      IMPLICIT         NONE
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, NFMAX, LENAMX, LENGMX
      PARAMETER        (NMAX=100,NFMAX=100,LENAMX=300,LENGMX=300)
      INTEGER          LENCW, LENIW, LENRW
      PARAMETER        (LENCW=600,LENIW=600,LENRW=600)
*     .. Local Scalars ..
      DOUBLE PRECISION OBJADD, SINF
```

```
      INTEGER          I, IFAIL, LENA, LENG, N, NEA, NEG, NF, NFNAME,
     +                 NINF, NS, NXNAME, OBJROW, START
      CHARACTER*8      PROB
*     .. Local Arrays ..
      DOUBLE PRECISION A(LENAMX), F(NFMAX), FLOW(NFMAX), FMUL(NFMAX),
     +                 FUPP(NFMAX), RUSER(1), RW(LENRW), X(NMAX),
     +                 XLOW(NMAX), XMUL(NMAX), XUPP(NMAX)
      INTEGER          FSTATE(NFMAX), IAFUN(LENAMX), IGFUN(LENGMX),
     +                 IUSER(1), IW(LENIW), JAVAR(LENAMX),
     +                 JGVAR(LENGMX), XSTATE(NMAX)
      CHARACTER*8      CUSER(1), CW(LENCW), FNAMES(NFMAX), XNAMES(NMAX)
*     .. External Subroutines ..
      EXTERNAL         E04VGF, E04VHF, E04VJF, E04VLF, E04VMF, USRFUN
*     .. Executable Statements ..
      WRITE (NOUT,*) 'E04VJF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NF
*
      IF (N.LE.NMAX .AND. NF.LE.NFMAX) THEN
*
*        Call E04VGF to initialise E04VJF.
         IFAIL = -1
         CALL E04VGF(CW,LENCW,IW,LENIW,RW,LENRW,IFAIL)
*
         LENA = LENAMX
         LENG = LENGMX
*
*        Read the bounds on the variables.
         DO 20 I = 1, N
            READ (NIN,*) XLOW(I), XUPP(I)
   20    CONTINUE
*
         DO 40 I = 1, N
            X(I) = 0.0D0
   40    CONTINUE
*
*        Determine the Jacobian structure.
         IFAIL = 0
         CALL E04VJF(NF,N,USRFUN,IAFUN,JAVAR,A,LENA,NEA,IGFUN,JGVAR,
     +               LENG,NEG,X,XLOW,XUPP,CW,LENCW,IW,LENIW,RW,LENRW,
     +               CUSER,IUSER,RUSER,IFAIL)
*
*        Print the Jacobian structure.
         WRITE (NOUT,*)
         WRITE (NOUT,99999) NEA
         WRITE (NOUT,99998)
         WRITE (NOUT,99997)
         DO 60 I = 1, NEA
            WRITE (NOUT,99996) I, IAFUN(I), JAVAR(I), A(I)
   60    CONTINUE
         WRITE (NOUT,*)
         WRITE (NOUT,99995) NEG
         WRITE (NOUT,99994)
         WRITE (NOUT,99993)
         DO 80 I = 1, NEG
            WRITE (NOUT,99992) I, IGFUN(I), JGVAR(I)
   80    CONTINUE
*
*        Now that we have the determined the structure of the
*        Jacobian, set up the information necessary to solve
*        the optimization problem.
         START = 0
         NFNAME = 1
         NXNAME = 1
         PROB = 'E04VJFE'
         OBJADD = 0.0
         DO 100 I = 1, N
            X(I) = 0.0D0
            XSTATE(I) = 0
            XMUL(I) = 0.0D0
```

```
  100    CONTINUE
         DO 120 I = 1, NF
            F(I) = 0.0D0
            FSTATE(I) = 0
            FMUL(I) = 0.0D0
  120    CONTINUE
*
*        The row containing the objective function.
         READ (NIN,*) OBJROW
*
*        Read the bounds on the functions.
         DO 140 I = 1, NF
            READ (NIN,*) FLOW(I), FUPP(I)
  140    CONTINUE
*
*        By default E04VHF does not print monitoring
*        information. Set the print file unit or the summary
*        file unit to get information.
         CALL E04VMF('Print file',NOUT,CW,IW,RW,IFAIL)
*
*        Tell E04VHF that we supply no derivatives in USRFUN.
         CALL E04VLF('Derivative option 0',CW,IW,RW,IFAIL)
*
*        Solve the problem.
         IFAIL = -1
         CALL E04VHF(START,NF,N,NXNAME,NFNAME,OBJADD,OBJROW,PROB,USRFUN,
     +               IAFUN,JAVAR,A,LENA,NEA,IGFUN,JGVAR,LENG,NEG,XLOW,
     +               XUPP,XNAMES,FLOW,FUPP,FNAMES,X,XSTATE,XMUL,F,
     +               FSTATE,FMUL,NS,NINF,SINF,CW,LENCW,IW,LENIW,RW,
     +               LENRW,CUSER,IUSER,RUSER,IFAIL)
*
         WRITE (NOUT,*)
         WRITE (NOUT,99991) IFAIL
         IF (IFAIL.EQ.0 .OR. IFAIL.EQ.4) THEN
            WRITE (NOUT,99990) F(OBJROW)
            WRITE (NOUT,99989) (X(I),I=1,N)
         END IF
      END IF
*
      STOP
*
99999 FORMAT (1X,'NEA (the number of non-zero entries in A) = ',I3)
99998 FORMAT (1X,'  I     IAFUN(I)   JAVAR(I)          A(I)')
99997 FORMAT (1X,'----    --------   --------   -----------')
99996 FORMAT (1X,I3,2I10,1P,E18.4)
99995 FORMAT (1X,'NEG (the number of non-zero entries in G) = ',I3)
99994 FORMAT (1X,'  I     IGFUN(I)   JGVAR(I)')
99993 FORMAT (1X,'----    --------   --------')
99992 FORMAT (1X,I3,2I10)
99991 FORMAT (1X,'On exit from E04VHF, IFAIL = ',I5)
99990 FORMAT (1X,'Final objective value = ',F11.1)
99989 FORMAT (1X,'Optimal X = ',7F9.2)
      END

      SUBROUTINE USRFUN(STATUS,N,X,NEEDF,NF,F,NEEDG,LENG,G,CUSER,IUSER,
     +                  RUSER)
      IMPLICIT          NONE
*     .. Scalar Arguments ..
      INTEGER           LENG, N, NEEDF, NEEDG, NF, STATUS
*     .. Array Arguments ..
      DOUBLE PRECISION  F(NF), G(LENG), RUSER(*), X(N)
      INTEGER           IUSER(*)
      CHARACTER*8       CUSER(*)
*     .. Intrinsic Functions ..
      INTRINSIC         SIN
*     .. Executable Statements ..
      IF (NEEDF.GT.0) THEN
         F(1) = 1000.0D+0*SIN(-X(1)-0.25D+0) + 1000.0D+0*SIN(-X(2)
     +          -0.25D+0) - X(3)
         F(2) = 1000.0D+0*SIN(X(1)-0.25D+0) + 1000.0D+0*SIN(X(1)-X(2)
     +          -0.25D+0) - X(4)
```

```
          F(3) = 1000.0D+0*SIN(X(2)-X(1)-0.25D+0) + 1000.0D+0*SIN(X(2)
     +           -0.25D+0)
          F(4) = -X(1) + X(2)
          F(5) = X(1) - X(2)
          F(6) = 1.0D-6*X(3)**3 + 2.0D-6*X(4)**3/3.0D+0 + 3*X(3) + 2*X(4)
       END IF
*
       END
```

9.2 Program Data

```
E04VJF Example Program Data
 4   6             : Values of N and NF
-0.55D0    0.55D0  : Bounds on the variables, XLOW(i), XUPP(i), for i = 1 to N
-0.55D0    0.55D0
 0.0D0   1200.0D0
 0.0D0   1200.0D0

 6                 : Value of OBJROW
 -894.8D0 -894.8D0 : Bounds on the functions, FLOW(i), FUPP(i), for i = 1 to NF
 -894.8D0 -894.8D0
-1294.8D0 -1294.8D0
-0.55D0    1.0D25
-0.55D0    1.0D25
-1.0D25    1.0D25
```

9.3 Program Results

```
 E04VJF Example Program Results

 NEA (the number of non-zero entries in A) =   4
   I     IAFUN(I)   JAVAR(I)          A(I)
 ----    --------   --------   -----------
   1         4         1      -1.0000E+00
   2         5         1       1.0000E+00
   3         4         2       1.0000E+00
   4         5         2      -1.0000E+00

 NEG (the number of non-zero entries in G) =  10
   I     IGFUN(I)   JGVAR(I)
 ----    --------   --------
   1         1         1
   2         2         1
   3         3         1
   4         1         2
   5         2         2
   6         3         2
   7         6         3
   8         6         4
   9         1         3
  10         2         4

 Parameters
 ==========

 Files
 -----
 Solution file.....         0      Old basis file .....         0      (Print file)...........         6
 Insert file.......         0      New basis file .....         0      (Summary file).........         0
 Punch file........         0      Backup basis file......      0
 Load file.........         0      Dump file..............      0

 Frequencies
 -----------
 Print frequency.....       100    Check frequency.....        60    Save new basis map.....       100
 Summary frequency...       100    Factorization frequency     50    Expand frequency.......     10000

 QP subproblems
 --------------
 QPsolver Cholesky......
```

```
Scale tolerance........     0.900       Minor feasibility tol..  1.00E-06       Iteration limit........     10000
Scale option...........         0       Minor optimality  tol..  1.00E-06       Minor print level......         1
Crash tolerance........     0.100       Pivot tolerance........  1.11E-15       Partial price..........         1
Crash option...........         3       Elastic weight.........  1.00E+04       Prtl price section ( A)         4
                                        New superbasics........        99       Prtl price section (-I)         5

The SQP Method
--------------
Minimize...............                 Cold start.............                 Proximal Point method..         1
Nonlinear objectiv vars         2       Objective Row..........         6       Function precision.....  1.72E-13
Unbounded step size....  1.00E+20       Superbasics limit......         4       Difference interval....  4.15E-07
Unbounded objective....  1.00E+15       Hessian dimension......         4       Central difference int.  5.57E-05
Major step limit.......  2.00E+00       Nonderiv.  linesearch..                 Derivative option......         0
Major iterations limit.      1000       Linesearch tolerance...   0.90000       Verify level...........         0
Minor iterations limit.       500       Penalty parameter......  0.00E+00       Major Print Level......         1
                                        Major optimality tol...  2.00E-06

Hessian Approximation
---------------------
Full-Memory Hessian....                 Hessian updates........  99999999       Hessian frequency......  99999999
                                                                               Hessian flush..........  99999999

Nonlinear constraints
---------------------
Nonlinear constraints..         3       Major feasibility tol..  1.00E-06       Violation limit........  1.00E+06
Nonlinear Jacobian vars         4

Miscellaneous
-------------
LU factor tolerance....      3.99       LU singularity tol.....  1.05E-08       Timing level...........         0
LU update tolerance....      3.99       LU swap tolerance......  1.03E-04       Debug level............         0
LU partial  pivoting...                 eps (machine precision)  1.11E-16       System information.....        No

Nonlinear constraints       3     Linear constraints       2
Nonlinear variables         4     Linear variables         0
Jacobian  variables         4     Objective variables      2
Total constraints           5     Total variables          4

The user has defined       0   out of      10   first  derivatives

  Itns Major Minors    Step   nCon Feasible  Optimal  MeritFunction     L+U BSwap     nS  condHz Penalty
     4     0      4             1  8.0E+02  1.0E+00  0.0000000E+00      14            1 3.0E+07         _  r
     6     1      2 1.2E-03     2  4.0E+02  1.0E+00  1.7331709E+06      13            1 1.3E+07 5.1E+00 _n rl
     7     2      1 1.3E-03     3  2.7E+02  5.5E-01  1.7301152E+06      13                      5.1E+00 _s  l
     7     3      0 7.5E-03     4  8.8E+01  5.4E-01  8.8193384E+05      13                      2.8E+00 _   l
     7     4      0 2.3E-02     5  2.9E+01  5.3E-01  8.4262007E+05      13                      2.8E+00 _   l
     7     5      0 6.9E-02     6  8.9E+00  5.2E-01  7.3075570E+05      13                      2.8E+00 _   l
     8     6      1 2.2E-01     7  2.3E+00  8.0E+01  4.4817383E+05      13            1 1.2E+04 2.8E+00 _   l
     9     7      1 8.3E-01     8  1.7E-01  9.2E+00  2.4331006E+04      13            1 9.5E+03 2.8E+00 _   l
    10     8      1 1.0E+00     9  6.5E-03  4.0E+01  5.3126219E+03      13     1      1 1.3E+02 2.8E+00 _
    11     9      1 1.0E+00    10  4.6E-03  1.2E+01  5.1602362E+03      13            1 9.4E+01 2.8E+00 _
    12    10      1 1.0E+00    11  2.3E-04  6.2E-02  5.1265653E+03      13            1 9.6E+01 2.8E+00 _
    13    11      1 1.0E+00    12 (1.3E-08) 2.9E-04  5.1264981E+03      13            1 1.2E+02 2.8E+00 _      c
    14    11      2 1.0E+00    12 (1.3E-08) 2.7E-04  5.1264981E+03      13            1 1.2E+02 2.8E+00 _      c
    15    12      1 1.0E+00    13 (5.6E-13) 7.1E-05  5.1264981E+03      13            1 9.5E+01 2.8E+00 _      c
    16    13      1 1.0E+00    14 (1.8E-14)(1.0E-09) 5.1264981E+03      13            1 9.5E+01 2.8E+00 _      c

E04VHF EXIT   0 -- finished successfully
E04VHF INFO   1 -- optimality conditions satisfied

Problem name                 E04VJFE
No. of iterations                16   Objective value      5.1264981096E+03
No. of major iterations          13   Linear objective     0.0000000000E+00
Penalty parameter          2.780E+00  Nonlinear objective  5.1264981096E+03
No. of calls to funobj         105    No. of calls to funcon          105
Calls with modes 1,2 (known g)  14    Calls with modes 1,2 (known g)   14
Calls for forward differencing  48    Calls for forward differencing   48
Calls for central differencing  24    Calls for central differencing   24
No. of superbasics               1    No. of basic nonlinears           3
No. of degenerate steps          0    Percentage                     0.00
Max x                    2 1.0E+03    Max pi                   3 5.5E+00
```

```
 Max Primal infeas           0 0.0E+00   Max Dual infeas             1 1.0E-08
 Nonlinear constraint violn    1.5E-11

 Name           E04VJFE                  Objective Value      5.1264981096E+03

 Status         Optimal Soln             Iteration     16    Superbasics     1

 Objective               (Min)
 RHS
 Ranges
 Bounds

 Section 1 - Rows

  Number  ...Row.. State  ...Activity...  Slack Activity  ..Lower Limit.  ..Upper Limit.  .Dual Activity    ..i

       5  r      1    EQ      -894.80000         0.00000      -894.80000      -894.80000        -4.38698      1
       6  r      2    EQ      -894.80000         0.00000      -894.80000      -894.80000        -4.10563      2
       7  r      3    EQ     -1294.80000         0.00000     -1294.80000     -1294.80000        -5.46328      3
       8  r      4    BS        -0.51511         0.03489        -0.55000            None               .      4
       9  r      5    BS         0.51511         1.06511        -0.55000            None               .      5

 Section 2 - Columns

  Number  .Column. State  ...Activity...  .Obj Gradient.  ..Lower Limit.  ..Upper Limit.  Reduced Gradnt    m+j

       1  x      1    BS         0.11888               .         -0.55000         0.55000         0.00000      6
       2  x      2    BS        -0.39623               .         -0.55000         0.55000         0.00000      7
       3  x      3   SBS       679.94532         4.38698               .       1200.00000         0.00000      8
       4  x      4    BS      1026.06713         4.10563               .       1200.00000         0.00000      9

 On exit from E04VHF, IFAIL =     0
 Final objective value =      5126.5
 Optimal X =      0.12    -0.40   679.95  1026.07
```