NAG Toolbox for MATLAB

e04zc

1 Purpose

e04zc checks that user-supplied (sub)programs for evaluating an objective function, constraint functions and their first derivatives, produce derivative values which are consistent with the function and constraint values calculated.

2 Syntax

```
[c, cjac, objf, objgrd, user, ifail] = e04zc(ncnln, confun, objfun, x,
'n', n, 'user', user)
```

3 Description

Routines for minimizing a function of several variables subject to general equality and/or inequality constraints may require you to provide (sub)programs to evaluate the objective function $F(x_1, x_2, \ldots, x_n)$, constraint functions $c_i(x_1, x_2, \ldots, x_n)$, for $i = 1, 2, \ldots, m$, and their first derivatives. e04zc is designed to check the derivatives calculated by such user-supplied (sub)programs . As well as the functions to be checked (**confun** and **objfun**), you must supply a point $x = (x_1, x_2, \ldots, x_n)^{\mathrm{T}}$ at which the checks will be made.

To check the first derivatives of F, the function first calls user-supplied (sub)program **objfun** to evaluate F and its first derivatives $g_j = \frac{\partial F}{\partial x_j}$, for $j = 1, 2, \ldots, n$ at x. The components of the user-supplied derivatives along two orthogonal directions (defined by unit vectors p_1 and p_2, say) are then calculated; these will be $g^{\mathrm{T}}p_1$ and $g^{\mathrm{T}}p_2$ respectively. The same components are also estimated by finite differences, giving quantities

$$v_k = \frac{F(x + hp_k) - F(x)}{h}, \qquad k = 1, 2$$

where h is a small positive scalar. If the relative difference between v_1 and $g^{\mathrm{T}}p_1$ or between v_2 and $g^{\mathrm{T}}p_2$ is 'judged' too large, the error indicator **ifail** (see Section 6) is set to 2.

When $n = 1$ only p_1 and v_1 are generated.

Similar checks are made of whether components of the first derivatives

$$\frac{\partial c_i}{\partial x_j}, \qquad i = 1, 2, \ldots, m \text{ and } j = 1, 2, \ldots, n$$

(as calculated by (sub)program **confun** at x) are consistent with difference approximations to the same quantities.

4 References

Gill P E, Murray W and Wright M H 1981 *Practical Optimization* Academic Press

5 Parameters

5.1 Compulsory Input Parameters

1: **ncnln – int32 scalar**

The number m of constraint functions.

Constraint: **ncnln** ≥ 0.

2: **confun – string containing name of m-file**

confun must calculate the vector $c(x)$ of nonlinear constraint functions and its Jacobian for a specified n-vector x. If there are no nonlinear constraints (**ncnln** $= 0$), **confun** will not be called by e04zc and **confun** may be the string `'e54vdm'`. If there are nonlinear constraints, e04zc always calls **confun** and user-supplied (sub)program **objfun** together, in that order.

```
[mode, c, cjac, user] = confun(mode, ncnln, n, ldcjac, x, nstate,
user)
```

Input Parameters

1: **mode – int32 scalar**

mode is a flag that you may set within **confun** to indicate a failure in the evaluation of the nonlinear constraints.

Is always nonnegative.

If **mode** is negative on exit from **confun**, then execution of e04zc will be terminated with **ifail** containing the negative value of **mode**.

2: **ncnln – int32 scalar**

The number m of nonlinear constraints, as input to e04zc.

3: **n – int32 scalar**

The number n of variables, as input to e04zc.

4: **ldcjac – int32 scalar**

The first dimension of the array **cjac** and the length of the array **c**, as input to e04zc.

5: **x(n) – double array**

The vector x of variables at which the constraint functions are to be evaluated.

6: **nstate – int32 scalar**

Will be 1 on the first call to **confun** by e04zc, and is 0 for the two subsequent calls. Thus, if you wish, **nstate** may be tested within **confun** in order to perform certain calculations once only. For example, you may read data or initialize global variables when **nstate** $= 1$. In addition, the constant elements of **cjac** can be set in **confun** when **nstate** $= 1$, and need not be defined on subsequent calls.

7: **user – Any MATLAB object**

confun is called from e04zc with **user** as supplied to e04zc

Output Parameters

1: **mode – int32 scalar**

mode is a flag that you may set within **confun** to indicate a failure in the evaluation of the nonlinear constraints.

Is always nonnegative.

If **mode** is negative on exit from **confun**, then execution of e04zc will be terminated with **ifail** containing the negative value of **mode**.

2: **c**(**ldcjac**) **– double array**

Must contain **ncnln** nonlinear constraint values, with the value of the *j*th nonlinear constraint in **c**(j).

3: **cjac**(**ldcjac,n**) **– double array**

Must contain the Jacobian of the nonlinear constraint functions with the *i*th row of **cjac** containing the gradient of the *i*th nonlinear constraint, i.e., **cjac**(i,j) must contain the partial derivative of c_i with respect to x_j. If **cjac** contains any constant elements, a saving in computation can be made by setting them once only, when **nstate** $= 1$.

4: **user – Any MATLAB object**

confun is called from e04zc with **user** as supplied to e04zc

3: **objfun – string containing name of m-file**

objfun must calculate the objective function $F(x)$ and its gradient for a specified n-element vector x.

```
[mode, objf, objgrd, user] = objfun(mode, n, x, nstate, user)
```

Input Parameters

1: **mode – int32 scalar**

mode is a flag that you may set within **objfun** to indicate a failure in the evaluation of the objective function.

Is always nonnegative.

If **mode** is negative on exit from **objfun**, then execution of e04zc will be terminated with **ifail** set to **mode**.

2: **n – int32 scalar**

The number n of variables as input to e04zc.

3: **x**(**n**) **– double array**

The vector x of variables at which the objective function is to be evaluated.

4: **nstate – int32 scalar**

Will be 1 on the first call to **objfun** by e04zc, and is 0 on the two subsequent calls. Thus, if you wish, **nstate** may be tested in order to perform certain calculations only on the first call of **objfun**, e.g., read data or initialize global variables. Note that if there are any nonlinear constraints, (sub)program **confun** and **objfun** are called together, in that order.

5: **user – Any MATLAB object**

objfun is called from e04zc with **user** as supplied to e04zc

Output Parameters

1: **mode – int32 scalar**

mode is a flag that you may set within **objfun** to indicate a failure in the evaluation of the objective function.

Is always nonnegative.

If **mode** is negative on exit from **objfun**, then execution of e04zc will be terminated with **ifail** set to **mode**.

2: **objf – double scalar**

Must be set to the value of the objective function.

3: **objgrd(n) – double array**

Must contain the gradient vector of the objective function, with **objgrd**(j) containing the partial derivative of F with respect to x_j.

4: **user – Any MATLAB object**

objfun is called from e04zc with **user** as supplied to e04zc

4: **x(n) – double array**

x(j), for $j = 1, 2, \ldots, n$ must be set to the co-ordinates of a suitable point x at which to check the derivatives calculated by (sub)program **confun** and user-supplied (sub)program **objfun**. ‘Obvious’ settings such as 0 or 1, should not be used since, at such points, incorrect terms may take correct values (particularly zero), so that errors could go undetected. Similarly, it is preferable that no two elements of x should be equal.

5.2 Optional Input Parameters

1: **n – int32 scalar**

Default: The dimension of the arrays **cjac**, **objgrd**, **x**. (An error is raised if these dimensions are not equal.)

the number n of independent variables in the objective and constraint functions.

Constraint: $\mathbf{n} \geq 1$.

2: **user – Any MATLAB object**

user is not used by e04zc, but is passed to **confun** and **objfun**. Note that for large objects it may be more efficient to use a global variable which is accessible from the m-files than to use **user**.

5.3 Input Parameters Omitted from the MATLAB Interface

ldcjac, work, lwork

5.4 Output Parameters

1: **c(ldcjac) – double array**

Unless you set **mode** < 0 in the first call of (sub)program **confun**, **c**(i) contains the value of $c_i(x)$ at the point given in **x**, for $i = 1, 2, \ldots, m$.

If **ncnln** $= 0$, **c** is not referenced.

2: **cjac(ldcjac,n) – double array**

Unless you set **mode** < 0 in the first call of (sub)program **confun**, **cjac**(i,j) contains the value of the derivative $\frac{\partial c_i}{\partial x_j}$ at the point given in **x**, as calculated by **confun**, for $j = 1, 2, \ldots, n$ and $i = 1, 2, \ldots, m$.

If **ncnln** $= 0$, **cjac** is not referenced.

3: **objf – double scalar**

Unless you set **mode** < 0 in the first call of user-supplied (sub)program **objfun**, **objf** contains the value of the objective function $F(x)$ at the point given in **x**.

4: **objgrd(n) – double array**

Unless you set **mode** < 0 in the first call of user-supplied (sub)program **objfun**, **objgrd**(j) contains the value of the derivative $\frac{\partial F}{\partial x_j}$ at the point given in **x**, as calculated by **objfun**, for $j = 1, 2, \ldots, n$.

5: **user – Any MATLAB object**

user is not used by e04zc, but is passed to **confun** and **objfun**. Note that for large objects it may be more efficient to use a global variable which is accessible from the m-files than to use **user**.

6: **ifail – int32 scalar**

0 unless the function detects an error (see Section 6).

6 Error Indicators and Warnings

Note: e04zc may return useful information for one or more of the following detected errors or warnings.

ifail < 0

A negative value of **ifail** indicates an exit from e04zc because you have set **mode** negative in the user-supplied (sub)programs **objfun** or **confun**. The value of **ifail** will be the same as your setting of **mode**. The checks on **objfun** and **confun** will not have been completed.

ifail $= 1$

On entry, **n** < 1,
or **ncnln** < 0,
or **ldcjac** $< \max(1, \textbf{ncnln})$,
or **lwork** $< 4 \times \textbf{n} + \textbf{ncnln} + \textbf{n} \times \textbf{ldcjac}$.

ifail $= 2$

You should check carefully the derivation and programming of expressions for the derivatives of $F(x)$, because it is very unlikely that user-supplied (sub)program **objfun** is calculating them correctly.

ifail $= 2 + i$, for $i = 1, 2, \ldots,$ **ncnln**

You should check carefully the derivation and programming of expressions for the derivatives of $c_i(x)$, because it is very unlikely that (sub)program **confun** is calculating them correctly. See also Section 7.

7 Accuracy

ifail is set to 2 if

$$\left(v_k - g^{\mathrm{T}} p_k\right)^2 \geq h \times \left(\left(g^{\mathrm{T}} p_k\right)^2 + 1\right)$$

for $k = 1$ or 2. (See Section 3 for definitions of the quantities involved.) The scalar h is set equal to $\sqrt{\epsilon}$, where ϵ is the ***machine precision*** (see x02aj).

ifail is set to $2 + i$ if a relation analogous to that given above holds for c_i and its calculated derivatives.

8 Further Comments

The user-supplied (sub)programs **confun** and **objfun** are both called three times unless **ncnln** $= 0$, in which case **confun** is not called.

Before using e04zc to check the calculation of first derivatives, you should be confident that (sub)program **confun** and user-supplied (sub)program **objfun** are calculating F and the c_i correctly. The usual way of checking the calculation of these function values is to compare values of $F(x)$ and the $c_i(x)$ calculated by **objfun** and **confun** at nontrivial points x with values calculated independently. ('Non-trivial' means that, as when setting x before calling e04zc, co-ordinates such as 0 or 1 should be avoided.)

9 Example

```
e04zc_confun.m
```

```
function [mode, c, cjac, user] = confun(mode, ncnln, n, nrowj, x,
nstate, user)
  c = zeros(nrowj, 1);
  cjac = zeros(nrowj, n);

  c(1) = x(1)^2 + x(6)^2;
  cjac(1,1) = 2*x(1);
  cjac(1,6) = 2*x(6);
  c(2) = (x(2)-x(1))^2 + (x(7)-x(6))^2;
  cjac(2,1) = -2*(x(2)-x(1));
  cjac(2,2) = 2*(x(2)-x(1));
  cjac(2,6) = -2*(x(7)-x(6));
  cjac(2,7) = 2*(x(7)-x(6));
  c(3) = (x(3)-x(1))^2 + x(6)^2;
  cjac(3,1) = -2*(x(3)-x(1));
  cjac(3,3) = 2*(x(3)-x(1));
  cjac(3,6) = 2*x(6);
  c(4) = (x(1)-x(4))^2 + (x(6)-x(8))^2;
  cjac(4,1) = 2*(x(1)-x(4));
  cjac(4,4) = -2*(x(1)-x(4));
  cjac(4,6) = 2*(x(6)-x(8));
  cjac(4,8) = -2*(x(6)-x(8));
  c(5) = (x(1)-x(5))^2 + (x(6)-x(9))^2;
  cjac(5,1) = 2*(x(1)-x(5));
  cjac(5,5) = -2*(x(1)-x(5));
  cjac(5,6) = 2*(x(6)-x(9));
  cjac(5,9) = -2*(x(6)-x(9));
  c(6) = x(2)^2 + x(7)^2;
  cjac(6,2) = 2*x(2);
  cjac(6,7) = 2*x(7);
  c(7) = (x(3)-x(2))^2 + x(7)^2;
  cjac(7,2) = -2*(x(3)-x(2));
  cjac(7,3) = 2*(x(3)-x(2));
  cjac(7,7) = 2*x(7);
  c(8) = (x(4)-x(2))^2 + (x(8)-x(7))^2;
  cjac(8,2) = -2*(x(4)-x(2));
  cjac(8,4) = 2*(x(4)-x(2));
  cjac(8,7) = -2*(x(8)-x(7));
  cjac(8,8) = 2*(x(8)-x(7));
  c(9) = (x(2)-x(5))^2 + (x(7)-x(9))^2;
  cjac(9,2) = 2*(x(2)-x(5));
  cjac(9,5) = -2*(x(2)-x(5));
  cjac(9,7) = 2*(x(7)-x(9));
  cjac(9,9) = -2*(x(7)-x(9));
  c(10) = x(3)^2;
  cjac(10,3) = 2*x(3);
  c(11) = (x(4)-x(3))^2 + x(8)^2;
  cjac(11,3) = -2*(x(4)-x(3));
  cjac(11,4) = 2*(x(4)-x(3));
  cjac(11,8) = 2*x(8);
  c(12) = (x(5)-x(3))^2 + x(9)^2;
  cjac(12,3) = -2*(x(5)-x(3));
  cjac(12,5) = 2*(x(5)-x(3));
  cjac(12,9) = 2*x(9);
  c(13) = x(4)^2 + x(8)^2;
  cjac(13,4) = 2*x(4);
  cjac(13,8) = 2*x(8);
  c(14) = (x(4)-x(5))^2 + (x(9)-x(8))^2;
  cjac(14,4) = 2*(x(4)-x(5));
  cjac(14,5) = -2*(x(4)-x(5));
  cjac(14,8) = -2*(x(9)-x(8));
  cjac(14,9) = 2*(x(9)-x(8));
  c(15) = x(5)^2 + x(9)^2;
  cjac(15,5) = 2*x(5);
  cjac(15,9) = 2*x(9);
```

e04zc_objfun.m

```
function [mode, objf, objgrd, user] = objfun(mode, n, x, nstate, user)
  objgrd = zeros(n, 1);

   objf = x(2)*x(6) - x(1)*x(7) + x(3)*x(7) + x(5)*x(8) - x(4)*x(9) -
x(3)*x(8);
  objf = -objf;
  objgrd(1) = x(7);
  objgrd(2) = -x(6);
  objgrd(3) = -x(7) + x(8);
  objgrd(4) = x(9);
  objgrd(5) = -x(8);
  objgrd(6) = -x(2);
  objgrd(7) = -x(3) + x(1);
  objgrd(8) = -x(5) + x(3);
  objgrd(9) = x(4);
```

```
ncnln = int32(15);
x = [1.1;
     1.2;
     1.3;
     1.4;
     1.5;
     1.6;
     1.7;
     1.8;
     1.9];
[c, cjac, objf, objgrd, user, ifail] = e04zc(ncnln, 'e04zc_confun',
'e04zc_objfun', x)
```

```
c =
    3.7700
    0.0200
    2.6000
    0.1300
    0.2500
    4.3300
    2.9000
    0.0500
    0.1300
    1.6900
    3.2500
    3.6500
    5.2000
    0.0200
    5.8600
cjac =
  Columns 1 through 7
    2.2000         0         0         0         0    3.2000         0
   -0.2000    0.2000         0         0         0   -0.2000    0.2000
   -0.4000         0    0.4000         0         0    3.2000         0
   -0.6000         0         0    0.6000         0   -0.4000         0
   -0.8000         0         0         0    0.8000   -0.6000         0
         0    2.4000         0         0         0         0    3.4000
         0   -0.2000    0.2000         0         0         0    3.4000
         0   -0.4000         0    0.4000         0         0   -0.2000
         0   -0.6000         0         0    0.6000         0   -0.4000
         0         0    2.6000         0         0         0         0
         0         0   -0.2000    0.2000         0         0         0
         0         0   -0.4000         0    0.4000         0         0
         0         0         0    2.8000         0         0         0
         0         0         0   -0.2000    0.2000         0         0
         0         0         0         0    3.0000         0         0
  Columns 8 through 9
         0         0
         0         0
         0         0
    0.4000         0
         0    0.6000
         0         0
         0         0
    0.2000         0
         0    0.4000
         0         0
    3.6000         0
         0    3.8000
    3.6000         0
   -0.2000    0.2000
         0    3.8000
objf =
    0.0400
objgrd =
    1.7000
   -1.6000
    0.1000
    1.9000
   -1.8000
   -1.2000
   -0.2000
   -0.2000
    1.4000
user =
     0
ifail =
           0
```