NAG Toolbox for MATLAB

e04ya

1 Purpose

e04ya checks that a (sub)program for evaluating a vector of functions and the matrix of their first derivatives produces derivative values which are consistent with the function values calculated.

2 Syntax

```
[fvec, fjac, iw, w, ifail] = e04ya(m, lsqfun, x, iw, w, 'n', n, 'liw',
liw, 'lw', lw)
```

3 Description

Routines for minimizing a sum of squares of m nonlinear functions (or 'residuals'), $f_i(x_1, x_2, \ldots, x_n)$, for $i = 1, 2, \ldots, m$; $m \geq n$, may require you to supply a (sub)program to evaluate the f_i and their first derivatives. e04ya checks the derivatives calculated by such user-supplied (sub)programs , e.g., functions of the form required for e04gb, e04gd and e04he. As well as the function to be checked (**lsqfun**), you must supply a point $x = (x_1, x_2, \ldots, x_n)^{\mathrm{T}}$ at which the check will be made. e04ya is essentially identical to CHKLSJ in the NPL Algorithms Library.

e04ya first calls user-supplied (sub)program **lsqfun** to evaluate the $f_i(x)$ and their first derivatives, and uses these to calculate the sum of squares $F(x) = \sum_{i=1}^{m} [f_i(x)]^2$, and its first derivatives $g_j = \frac{\partial F}{\partial x_j}_x$, for $j = 1, 2, \ldots, n$. The components of g 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, an error indicator is set.

4 References

None.

5 Parameters

5.1 Compulsory Input Parameters

1: **m – int32 scalar**

the number m of residuals, $f_i(x)$, and the number n of variables, x_j.

Constraint: $1 \leq \mathbf{n} \leq \mathbf{m}$.

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

lsqfun must calculate the vector of values $f_i(x)$ and their first derivatives $\frac{\partial f_i}{\partial x_j}$ at any point x. (The minimization functions mentioned in Section 3 give you the option of resetting a parameter to terminate immediately. e04ya will also terminate immediately, without finishing the checking process, if the parameter in question is reset.)

Its specification is:

```
[iflag, fvecc, fjacc, iw, w] = lsqfun(iflag, m, n, xc, ljc, iw, liw,
w, lw)
```

Input Parameters

1: **iflag – int32 scalar**

To **lsqfun**, **iflag** will be set to 2.

If you reset **iflag** to some negative number in **lsqfun** and return control to e04ya, the function will terminate immediately with **ifail** set to your setting of **iflag**.

2: **m – int32 scalar**

3: **n – int32 scalar**

The numbers m and n of residuals and variables, respectively.

4: **xc(n) – double array**

x, the point at which the values of the f_i and the $\frac{\partial f_i}{\partial x_j}$ are required.

5: **ljc – int32 scalar**

6: **iw(liw) – int32 array**

7: **liw – int32 scalar**

8: **w(lw) – double array**

9: **lw – int32 scalar**

These parameters are present so that **lsqfun** will be of the form required by the minimization functions mentioned in Section 3. **lsqfun** is called with the same parameters **iw**, **liw**, **w**, **lw** as in the call to e04ya. If the recommendation in the minimization function document is followed, you will have no reason to examine or change the elements of **iw** or **w**. In any case, **lsqfun must not change** the first $3 \times \mathbf{n} + \mathbf{m} + \mathbf{m} \times \mathbf{n}$ elements of **w**.

Output Parameters

1: **iflag – int32 scalar**

To **lsqfun**, **iflag** will be set to 2.

If you reset **iflag** to some negative number in **lsqfun** and return control to e04ya, the function will terminate immediately with **ifail** set to your setting of **iflag**.

2: **fvecc(m) – double array**

Unless **iflag** is reset to a negative number, **fvecc**(i) must contain the value of f_i at the point in **xc**, for $i = 1, 2, \ldots, m$.

3: **fjacc(ljc,n) – double array**

Unless **iflag** is reset to a negative number, **fjacc**(i,j) must contain the value of $\frac{\partial f_i}{\partial x_j}$ at the point in **xc**, for $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$.

4: **iw(liw) – int32 array**

5: **w(lw) – double array**

These parameters are present so that **lsqfun** will be of the form required by the minimization functions mentioned in Section 3. **lsqfun** is called with the same parameters **iw**, **liw**, **w**, **lw** as in the call to e04ya. If the recommendation in the minimization function document is followed, you will have no reason to examine or change the elements of **iw** or **w**. In any case, **lsqfun must not change** the first $3 \times \mathbf{n} + \mathbf{m} + \mathbf{m} \times \mathbf{n}$ elements of **w**.

3: **x(n) – double array**

$\mathbf{x}(j)(j = 1, 2, \ldots, n)$ must be set to the co-ordinates of a suitable point at which to check the derivatives calculated by user-supplied (sub)program **lsqfun**. 'Obvious' settings, such as 0 or 1, should not be used since, at such particular points, incorrect terms may take correct values (particularly zero), so that errors can go undetected. For a similar reason, it is preferable that no two elements of **x** should have the same value.

4: **iw(liw) – int32 array**

This array appears in the parameter list purely so that, if e04ya is called by another library function, the library function can pass quantities to user-supplied (sub)program **lsqfun** via **iw**. **iw** is not examined or changed by e04ya. In general you must provide an array **iw**, but are advised not to use it.

5: **w(lw) – double array**

Constraint: $\mathbf{lw} \geq 3 \times \mathbf{n} + \mathbf{m} + \mathbf{m} \times \mathbf{n}$.

5.2 Optional Input Parameters

1: **n – int32 scalar**

Default: For **n**, the dimension of the array **x**.

the number m of residuals, $f_i(x)$, and the number n of variables, x_j.

Constraint: $1 \leq \mathbf{n} \leq \mathbf{m}$.

2: **liw – int32 scalar**

Default: The dimension of the array **iw**.

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

3: **lw – int32 scalar**

Default: The dimension of the array **w**.

Constraint: $\mathbf{lw} \geq 3 \times \mathbf{n} + \mathbf{m} + \mathbf{m} \times \mathbf{n}$.

5.3 Input Parameters Omitted from the MATLAB Interface

ldfjac

5.4 Output Parameters

1: **fvec(m) – double array**

Unless you set **iflag** negative in the first call of user-supplied (sub)program **lsqfun**, **fvec**(i) contains the value of f_i at the point given by you in **x**, for $i = 1, 2, \ldots, m$.

2: **fjac(ldfjac,n) – double array**

Unless you set **iflag** negative in the first call of user-supplied (sub)program **lsqfun**, **fjac**(i,j) contains the value of the first derivative $\frac{\partial f_i}{\partial x_j}$ at the point given in **x**, as calculated by **lsqfun**, for $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$.

3: **iw(liw) – int32 array**

This array appears in the parameter list purely so that, if e04ya is called by another library function, the library function can pass quantities to user-supplied (sub)program **lsqfun** via **iw**. **iw** is not examined or changed by e04ya. In general you must provide an array **iw**, but are advised not to use it.

4: **w(lw) – double array**

5: **ifail – int32 scalar**

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

6 Error Indicators and Warnings

Note: e04ya 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 e04ya because you have set **iflag** negative in user-supplied (sub)program **lsqfun**. The setting of **ifail** will be the same as your setting of **iflag**. The check on **lsqfun** will not have been completed.

ifail $= 1$

On entry, $\mathbf{m} < \mathbf{n}$,
or $\mathbf{n} < 1$,
or $\mathbf{ldfjac} < \mathbf{m}$,
or $\mathbf{liw} < 1$,
or $\mathbf{lw} < 3 \times \mathbf{n} + \mathbf{m} + \mathbf{m} \times \mathbf{n}$.

ifail $= 2$

You should check carefully the derivation and programming of expressions for the $\frac{\partial f_i}{\partial x_j}$, because it is very unlikely that user-supplied (sub)program **lsqfun** is calculating them correctly.

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*** as given by x02aj.

8 Further Comments

e04ya calls user-supplied (sub)program **lsqfun** three times.

Before using e04ya to check the calculation of the first derivatives, you should be confident that user-supplied (sub)program **lsqfun** is calculating the residuals correctly.

e04ya only checks the derivatives calculated by a when $\mathbf{iflag} = 2$. So, if user-supplied (sub)program **lsqfun** is intended for use in conjunction with a minimization function which may set **iflag** to 1, you must check that, for given settings of the $\mathbf{xc}(j)$, **lsqfun** produces the same values for the $\frac{\partial f_i}{\partial x_j}$ when **iflag** is set to 1 as when **iflag** is set to 2.

9 Example

e04ya_lsqfun.m

```
function [iflag, fvecc, fjacc] = lsqfun(iflag, m, n, xc, ljc)
  global y t;

  fvecc = zeros(m, 1);
  fjacc = zeros(ljc, n);

  for i = 1:m
    denom = xc(2)*t(i,2) + xc(3)*t(i,3);
    if (iflag ~= 1)
      fvecc(i) = xc(1) + t(i,1)/denom - y(i);
    end
    if (iflag ~= 0)
      fjacc(i,1) = 1;
      dummy = -1/(denom*denom);
      fjacc(i,2) = t(i,1)*t(i,2)*dummy;
      fjacc(i,3) = t(i,1)*t(i,3)*dummy;
    end
  end
```

```
m = int32(15);
x = [0.19;
     -1.34;
     0.88];
iw = zeros(0,0,'int32');
w = zeros(69,1);
global y t;
y=[0.14,0.18,0.22,0.25,0.29,0.32,0.35,0.39,0.37,0.58,0.73,0.96,
1.34,2.10,4.39];

t = [1.0, 15.0, 1.0;
     2.0, 14.0, 2.0;
     3.0, 13.0, 3.0;
     4.0, 12.0, 4.0;
     5.0, 11.0, 5.0;
     6.0, 10.0, 6.0;
     7.0, 9.0, 7.0;
     8.0, 8.0, 8.0;
     9.0, 7.0, 7.0;
     10.0, 6.0, 6.0;
     11.0, 5.0, 5.0;
     12.0, 4.0, 4.0;
     13.0, 3.0, 3.0;
     14.0, 2.0, 2.0;
     15.0, 1.0, 1.0];

[fvec, fjac, iwOut, wOut, ifail] = e04ya(m, 'e04ya_lsqfun', x, iw, w)
```

```
fvec =
   -0.0020
   -0.1076
   -0.2330
   -0.3785
   -0.5836
   -0.8689
   -1.3464
   -2.3739
   -2.9750
   -4.0132
   -5.3226
   -7.2917
  -10.5703
  -17.1274
```

```
  -36.8087
fjac =
    1.0000   -0.0406   -0.0027
    1.0000   -0.0969   -0.0138
    1.0000   -0.1785   -0.0412
    1.0000   -0.3043   -0.1014
    1.0000   -0.5144   -0.2338
    1.0000   -0.9100   -0.5460
    1.0000   -1.8098   -1.4076
    1.0000   -4.7259   -4.7259
    1.0000   -6.0762   -6.0762
    1.0000   -7.8765   -7.8765
    1.0000  -10.3970  -10.3970
    1.0000  -14.1777  -14.1777
    1.0000  -20.4789  -20.4789
    1.0000  -33.0813  -33.0813
    1.0000  -70.8885  -70.8885
iwOut =
wOut =
     array elided
ifail =
           0
```