NAG Toolbox for MATLAB

e04yb

1 Purpose

e04yb checks that a (sub)program for evaluating the second derivative term of the Hessian matrix of a sum of squares is consistent with a for calculating the corresponding first derivatives.

2 Syntax

```
[fvec, fjac, b, iw, w, ifail] = e04yb(m, lsqfun, lsqhes, x, lb, iw, w,
'n', n)
```

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 quantities

$$b_{jk} = \sum_{i=1}^{m} f_i \frac{\partial^2 f_i}{\partial x_j \partial x_k}$$

for $j = 1, 2, \ldots, n$ and $k = 1, 2, \ldots, j$. e04yb is designed to check the b_{jk} calculated by such user-supplied (sub)programs . As well as the function to be checked (**lsqhes**), you must supply a (sub)program (**lsqfun**) to evaluate the f_i and their first derivatives, and a point $x = (x_1, x_2, \ldots, x_n)^{\mathrm{T}}$ at which the checks will be made. Note that e04yb checks functions of the form required by e04he. e04yb is essentially identical to CHKLSH in the NPL Algorithms Library.

e04yb first calls user-supplied (sub)program **lsqfun** and user-supplied (sub)program **lsqhes** to evaluate the first derivatives and the b_{jk} at x. Let J denote the m by n matrix of first derivatives of the residuals. The Hessian matrix of the sum of squares,

$$G = J^{\mathrm{T}}J + B,$$

is calculated and projected onto two orthogonal vectors y and z to give the scalars $y^{\mathrm{T}}Gy$ and $z^{\mathrm{T}}Gz$ respectively. The same projections of the Hessian matrix are also estimated by finite differences, giving

$$p = \left(y^{\mathrm{T}}g(x + hy) - y^{\mathrm{T}}g(x)\right)/h \quad \text{and}$$
$$q = \left(z^{\mathrm{T}}g(x + hz) - z^{\mathrm{T}}g(x)\right)/h$$

respectively, where $g()$ denotes the gradient vector of the sum of squares at the point in brackets and h is a small positive scalar. If the relative difference between p and $y^{\mathrm{T}}Gy$ or between q and $z^{\mathrm{T}}Gz$ 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. (e04he gives you the option of resetting parameters of **lsqfun** to cause the minimization process to terminate immediately. e04yb 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 resets **iflag** to some negative number in **lsqfun** and return control to e04yb, 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**

The point x 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 e04he. **lsqfun** is called with e04yb's parameters **iw**, **liw**, **w**, **lw** as these parameters. If the recommendation in e04he 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 $5 \times \mathbf{n} + \mathbf{m} + \mathbf{m} \times \mathbf{n} + \mathbf{n} \times (\mathbf{n} - 1)/2$ (or $6 + 2 \times \mathbf{m}$ if $\mathbf{n} = 1$) elements of **w**.

Output Parameters

1: **iflag – int32 scalar**

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

If you resets **iflag** to some negative number in **lsqfun** and return control to e04yb, 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 x, 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 x, 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 e04he. **lsqfun** is called with e04yb's parameters **iw**, **liw**, **w**, **lw** as these parameters. If the recommendation in e04he 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 $5 \times \mathbf{n} + \mathbf{m} + \mathbf{m} \times \mathbf{n} + \mathbf{n} \times (\mathbf{n}-1)/2$ (or $6 + 2 \times \mathbf{m}$ if $\mathbf{n} = 1$) elements of **w**.

Note: e04ya should be used to check the first derivatives calculated by **lsqfun** before e04yb is used to check the b_{jk} since e04yb assumes that the first derivatives are correct.

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

lsqhes must calculate the elements of the symmetric matrix

$$B(x) = \sum_{i=1}^{m} f_i(x) G_i(x),$$

at any point x, where $G_i(x)$ is the Hessian matrix of $f_i(x)$. (As with user-supplied (sub)program **lsqfun**, a parameter can be set to cause immediate termination.)

Its specification is:

```
[iflag, b, iw, w] = lsqhes(iflag, m, n, fvecc, xc, lb, iw, liw, w,
lw)
```

Input Parameters

1: **iflag – int32 scalar**

Is set to a nonnegative number.

If **lsqhes** resets **iflag** to some negative number, e04yb 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: **fvecc(m) – double array**

The value of the residual f_i at the point x, for $i = 1,2,\ldots,m$, so that the values of the f_i can be used in the calculation of the elements of **b**.

5: **xc(n) – double array**

The point x at which the elements of **b** are to be evaluated.

6: **lb – int32 scalar**

Gives the length of the array **b**.

7: **iw**(**liw**) – **int32 array**
8: **liw** – **int32 scalar**
9: **w**(**lw**) – **double array**
10: **lw** – **int32 scalar**

As in user-supplied (sub)program **lsqfun**, these parameters correspond to the parameters **iw**, **liw**, **w**, **lw** of e04yb. **lsqhes must not change** the first $5 \times \mathbf{n} + \mathbf{m} \times \mathbf{n} + \mathbf{n} \times (\mathbf{n} - 1)/2$ (or $6 + 2 \times \mathbf{m}$ if $\mathbf{n} = 1$) elements of **w**.

Output Parameters

1: **iflag** – **int32 scalar**

Is set to a nonnegative number.

If **lsqhes** resets **iflag** to some negative number, e04yb will terminate immediately, with **ifail** set to your setting of **iflag**.

2: **b**(**lb**) – **double array**

Unless **iflag** is reset to a negative number **b** must contain the lower triangle of the matrix $B(x)$, evaluated at the point in **xc**, stored by rows. (The upper triangle is not needed because the matrix is symmetric.) More precisely, $\mathbf{b}(j(j-1)/2 + k)$ must contain $\sum_{i=1}^{m} f_i \frac{\partial^2 f_i}{\partial x_j \partial x_k}$ evaluated at the point x, for $j = 1, 2, \ldots, n$ and $k = 1, 2, \ldots, j$.

3: **iw**(**liw**) – **int32 array**
4: **w**(**lw**) – **double array**

As in user-supplied (sub)program **lsqfun**, these parameters correspond to the parameters **iw**, **liw**, **w**, **lw** of e04yb. **lsqhes must not change** the first $5 \times \mathbf{n} + \mathbf{m} \times \mathbf{n} + \mathbf{n} \times (\mathbf{n} - 1)/2$ (or $6 + 2 \times \mathbf{m}$ if $\mathbf{n} = 1$) elements of **w**.

4: **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 b_{jk} calculated by user-supplied (sub)program **lsqhes**. '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 could go undetected. For a similar reason, it is preferable that no two elements of **x** should have the same value.

5: **lb** – **int32 scalar**

Constraint: $\mathbf{lb} \geq (\mathbf{n} + 1) \times \mathbf{n}/2$.

6: **iw**(**liw**) – **int32 array**

This array appears in the parameter list purely so that, if e04yb is called by another library function, the library function can pass quantities to user-supplied (sub)program **lsqfun** and user-supplied (sub)program **lsqhes** via **iw**. **iw** is not examined or changed by e04yb. In general you must provide an array **iw**, but are advised not to use it.

7: **w**(**lw**) – **double array**

The actual length of **w** as declared in the (sub)program from which e04yb is called.

Constraints:

if $\mathbf{n} > 1$, $\mathbf{lw} \geq 5 \times \mathbf{n} + \mathbf{m} + \mathbf{m} \times \mathbf{n} + \mathbf{n} \times (\mathbf{n} - 1)/2$;
if $\mathbf{n} = 1$, $\mathbf{lw} \geq 6 + 2 \times \mathbf{m}$.

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

5.3 Input Parameters Omitted from the MATLAB Interface

ldfjac, liw, lw

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 $\dfrac{\partial f_i}{\partial x_j}$ at the point given in **x**, as calculated by **lsqfun**, for $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$.

3: **b(lb) – double array**

Unless you set **iflag** negative in user-supplied (sub)program **lsqhes**, **b**$(j \times (j-1)/2 + k)$ contains the value of b_{jk} at the point given in **x** as calculated by **lsqhes**, for $j = 1, 2, \ldots, n$ and $k = 1, 2, \ldots, j$.

4: **iw(liw) – int32 array**

This array appears in the parameter list purely so that, if e04yb is called by another library function, the library function can pass quantities to user-supplied (sub)program **lsqfun** and user-supplied (sub)program **lsqhes** via **iw**. **iw** is not examined or changed by e04yb. In general you must provide an array **iw**, but are advised not to use it.

5: **w(lw) – double array**

6: **ifail – int32 scalar**

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

6 Error Indicators and Warnings

Note: e04yb 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 e04yb because you have set **iflag** negative in user-supplied (sub)programs **lsqfun** or **lsqhes**. The setting of **ifail** will be the same as your setting of **iflag**. The check on **lsqhes** will not have been completed.

ifail $= 1$

On entry, $\mathbf{m} < \mathbf{n}$,
or $\mathbf{n} < 1$,
or **ldfjac** $<$ **m**,
or $\mathbf{lb} < (\mathbf{n}+1) \times \mathbf{n}/2$,

or **liw** < 1,
or **lw** $< 5 \times \mathbf{n} + \mathbf{m} + \mathbf{m} \times \mathbf{n} + \mathbf{n} \times (\mathbf{n} - 1)/2$, if $\mathbf{n} > 1$,
or **lw** $< 6 + 2 \times \mathbf{m}$, if $\mathbf{n} = 1$.

ifail $= 2$

You should check carefully the derivation and programming of expressions for the b_{jk}, because it is very unlikely that user-supplied (sub)program **lsqhes** is calculating them correctly.

7 Accuracy

ifail is set to 2 if

$$\left|y^{\mathrm{T}}Gy - p\right| \geq \sqrt{h}\left(\left|y^{\mathrm{T}}Gy\right| + 1.0\right) \quad \text{or}$$
$$\left|z^{\mathrm{T}}Gz - q\right| \geq \sqrt{h}\left(\left|z^{\mathrm{T}}Gz\right| + 1.0\right)$$

where h is set equal to $\sqrt{\epsilon}$ (ϵ being the ***machine precision*** as given by x02aj) and other quantities are defined as in Section 3.

8 Further Comments

e04yb calls user-supplied (sub)program **lsqhes** once and user-supplied (sub)program **lsqfun** three times.

9 Example

e04yb_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);
    fvecc(i) = xc(1) + t(i,1)/denom - y(i);
    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
```

e04yb_lsqhes.m

```
function [iflag, b] = lsqhes(iflag, m, n, fvecc, xc, lb)
  global y t;
  b = zeros(lb, 1);

  sum22 = 0;
  sum32 = 0;
  sum33 = 0;
  for i = 1:m
     dummy = 2*t(i,1)/(xc(2)*t(i,2)+xc(3)*t(i,3))^3;
     sum22 = sum22 + fvecc(i)*dummy*t(i,2)^2;
     sum32 = sum32 + fvecc(i)*dummy*t(i,2)*t(i,3);
     sum33 = sum33 + fvecc(i)*dummy*t(i,3)^2;
  end
  b(3) = sum22;
  b(5) = sum32;
  b(6) = sum33;
```

```
m = int32(15);
x = [0.19;
     -1.34;
     0.88];
lb = int32(6);
iw = zeros(1,1,'int32');
w = zeros(78,1);
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, b, iwOut, wOut, ifail] = ...
    e04yb(m, 'e04yb_lsqfun', 'e04yb_lsqhes', x, lb, 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
b =
   1.0e+04 *
         0
         0
    1.5715
```

```
         0
    1.5712
    1.5710
iwOut =
           0
wOut =
     array elided
ifail =
           0
```