NAG Toolbox for MATLAB

e04ly

1 Purpose

e04ly is an easy-to-use modified-Newton algorithm for finding a minimum of a function, $F(x_1, x_2, \ldots, x_n)$ subject to fixed upper and lower bounds on the independent variables, $x_1, x_2, \ldots, x_n$ when first and second derivatives of F are available. It is intended for functions which are continuous and which have continuous first and second derivatives (although it will usually work even if the derivatives have occasional discontinuities).

2 Syntax

```
[bl, bu, x, f, g, user, ifail] = e04ly(ibound, funct2, hess2, bl, bu, x,
'n', n, 'user', user)
```

3 Description

e04ly is applicable to problems of the form:

$$\text{Minimize } F(x_1, x_2, \ldots, x_n) \qquad \text{subject to} \qquad l_j \leq x_j \leq u_j, \qquad j = 1, 2, \ldots, n$$

when first and second derivatives of $F(x)$ are available.

Special provision is made for problems which actually have no bounds on the x_j, problems which have only non-negativity bounds and problems in which $l_1 = l_2 = \cdots = l_n$ and $u_1 = u_2 = \cdots = u_n$. You must supply a (sub)program to calculate the values of $F(x)$ and its first derivatives at any point x and a (sub)program to calculate the second derivatives.

From a starting point you supplied there is generated, on the basis of estimates of the curvature of $F(x)$, a sequence of feasible points which is intended to converge to a local minimum of the constrained function.

4 References

Gill P E and Murray W 1976 Minimization subject to bounds on the variables *NPL Report NAC 72* National Physical Laboratory

5 Parameters

5.1 Compulsory Input Parameters

1: **ibound – int32 scalar**

Indicates whether the facility for dealing with bounds of special forms is to be used. It must be set to one of the following values:

ibound $= 0$

If youare supplying all the l_j and u_j individually.

ibound $= 1$

If there are no bounds on any x_j.

ibound $= 2$

If all the bounds are of the form $0 \leq x_j$.

ibound $= 3$

If $l_1 = l_2 = \cdots = l_n$ and $u_1 = u_2 = \cdots = u_n$.

Constraint: $0 \leq$ **ibound** ≤ 3.

2: **funct2 – string containing name of m-file**

You must supply this function to calculate the values of the function $F(x)$ and its first derivatives $\dfrac{\partial F}{\partial x_j}$ at any point x. It should be tested separately before being used in conjunction with e04ly (see the E04 Chapter Introduction).

Its specification is:

```
[fc, gc, user] = funct2(n, xc, user)
```

Input Parameters

1: **n – int32 scalar**

The number n of variables.

2: **xc**(**n**) **– double array**

The point x at which the function and its derivatives are required.

3: **user – Any MATLAB object**

funct2 is called from e04ly with **user** as supplied to e04ly

Output Parameters

1: **fc – double scalar**

The value of the function F at the current point x.

2: **gc**(**n**) **– double array**

gc(j) must be set to the value of the first derivative $\dfrac{\partial F}{\partial x_j}$ at the point x, for $j = 1, 2, \ldots, n$.

3: **user – Any MATLAB object**

funct2 is called from e04ly with **user** as supplied to e04ly

3: **hess2 – string containing name of m-file**

You must supply this function to evaluate the elements $H_{ij} = \dfrac{\partial^2 F}{\partial x_i \partial x_j}$ of the matrix of second derivatives of $F(x)$ at any point x. It should be tested separately before being used in conjunction with e04ly (see the E04 Chapter Introduction).

Its specification is:

```
[heslc, hesdc, user] = hess2(n, xc, lh, user)
```

Input Parameters

1: **n – int32 scalar**

The number n of variables.

2: **xc(n) – double array**

The point x at which the derivatives are required.

3: **lh – int32 scalar**

The length of the array **heslc**.

4: **user – Any MATLAB object**

hess2 is called from e04ly with **user** as supplied to e04ly

Output Parameters

1: **heslc(lh) – double array**

hess2 must place the strict lower triangle of the second derivative matrix H in **heslc**, stored by rows, i.e., set $\mathbf{heslc}((i-1)(i-2)/2+j) = \dfrac{\partial^2 F}{\partial x_i \partial x_j}$, for $i = 2, 3, \ldots, n$ and $j = 1, 2, \ldots, i-1$. (The upper triangle is not required because the matrix is symmetric.)

2: **hesdc(n) – double array**

Must contain the diagonal elements of the second derivative matrix, i.e., set $\mathbf{hesdc}(j) = \dfrac{\partial^2 F}{\partial x_j^2}$, for $j = 1, 2, \ldots, n$.

3: **user – Any MATLAB object**

hess2 is called from e04ly with **user** as supplied to e04ly

4: **bl(n) – double array**

The lower bounds l_j.

If **ibound** is set to 0, $\mathbf{bl}(j)$ must be set to l_j, for $j = 1, 2, \ldots, n$. (If a lower bound is not specified for any x_j, the corresponding $\mathbf{bl}(j)$ should be set to -10^6.)

If **ibound** is set to 3, you must set $\mathbf{bl}(1)$ to l_1; e04ly will then set the remaining elements of **bl** equal to $\mathbf{bl}(1)$.

5: **bu(n) – double array**

The upper bounds u_j.

If **ibound** is set to 0, $\mathbf{bu}(j)$ must be set to u_j, for $j = 1, 2, \ldots, n$. (If an upper bound is not specified for any x_j the corresponding $\mathbf{bu}(j)$ should be set to 10^6.)

If **ibound** is set to 3, you must set $\mathbf{bu}(1)$ to u_1; e04ly will then set the remaining elements of **bu** equal to $\mathbf{bu}(1)$.

6: **x(n) – double array**

$\mathbf{x}(j)$ must be set to a guess at the jth component of the position of the minimum, for $j = 1, 2, \ldots, n$. The function checks the gradient and the Hessian matrix at the starting point, and is more likely to detect any error in your programming if the initial $\mathbf{x}(j)$ are nonzero and mutually distinct.

5.2 Optional Input Parameters

1: **n – int32 scalar**

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

the number n of independent variables.

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

2: **user – Any MATLAB object**

user is not used by e04ly, but is passed to **funct2** and **hess2**. 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

iw, liw, w, lw

5.4 Output Parameters

1: **bl(n) – double array**

The lower bounds actually used by e04ly.

2: **bu(n) – double array**

The upper bounds actually used by e04ly.

3: **x(n) – double array**

The lowest point found during the calculations. Thus, if **ifail** $= 0$ on exit, $\mathbf{x}(j)$ is the jth component of the position of the minimum.

4: **f – double scalar**

The value of $F(x)$ corresponding to the final point stored in **x**.

5: **g(n) – double array**

The value of $\dfrac{\partial F}{\partial x_j}$ corresponding to the final point stored in **x**, for $j = 1, 2, \ldots, n$; the value of $\mathbf{g}(j)$ for variables not on a bound should normally be close to zero.

6: **user – Any MATLAB object**

user is not used by e04ly, but is passed to **funct2** and **hess2**. 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**.

7: **ifail – int32 scalar**

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

6 Error Indicators and Warnings

Note: e04ly may return useful information for one or more of the following detected errors or warnings.

ifail $= 1$

On entry, **n** < 1,
or **ibound** < 0,
or **ibound** > 3,
or **ibound** $= 0$ and **bl**$(j) >$ **bu**(j) for some j,
or **ibound** $= 3$ and **bl**$(1) >$ **bu**(1),
or **liw** $<$ **n** $+ 2$,
or **lw** $< \max(10, \mathbf{n} \times (\mathbf{n} + 7))$.

ifail $= 2$

There have been $50 \times$ **n** function evaluations, yet the algorithm does not seem to be converging. The calculations can be restarted from the final point held in **x**. The error may also indicate that $F(x)$ has no minimum.

ifail $= 3$

The conditions for a minimum have not all been met but a lower point could not be found and the algorithm has failed.

ifail $= 4$

Not used. (This value of the parameter is included so as to make the significance of **ifail** $= 5$ etc. consistent in the easy-to-use functions.)

ifail $= 5$
ifail $= 6$
ifail $= 7$
ifail $= 8$

There is some doubt about whether the point x found by e04ly is a minimum. The degree of confidence in the result decreases as **ifail** increases. Thus, when **ifail** $= 5$ it is probable that the final x gives a good estimate of the position of a minimum, but when **ifail** $= 8$ it is very unlikely that the function has found a minimum.

ifail $= 9$

In the search for a minimum, the modulus of one of the variables has become very large $(\sim 10^6)$. This indicates that there is a mistake in user-supplied (sub)programs **funct2** or **hess2**, that your problem has no finite solution, or that the problem needs rescaling (see Section 8).

ifail $= 10$

It is very likely that you have made an error in forming the gradient.

ifail $= 11$

It is very likely that you have made an error in forming the second derivatives.

If you are dissatisfied with the result (e.g., because **ifail** $=$ 5, 6, 7 or 8), it is worth restarting the calculations from a different starting point (not the point at which the failure occurred) in order to avoid the region which caused the failure.

7 Accuracy

When a successful exit is made then, for a computer with a mantissa of t decimals, one would expect to get about $t/2 - 1$ decimals accuracy in x, and about $t - 1$ decimals accuracy in F, provided the problem is reasonably well scaled.

8 Further Comments

The number of iterations required depends on the number of variables, the behaviour of $F(x)$ and the distance of the starting point from the solution. The number of operations performed in an iteration of e04ly is roughly proportional to $n^3 + O(n^2)$. In addition, each iteration makes one call of user-supplied (sub)program **hess2** and at least one call of user-supplied (sub)program **funct2**. So, unless $F(x)$, the gradient vector and the matrix of second derivatives can be evaluated very quickly, the run time will be dominated by the time spent in user-supplied (sub)programs **funct2** and **hess2**.

Ideally the problem should be scaled so that at the solution the value of $F(x)$ and the corresponding values of $x_1, x_2, \ldots x_n$ are each in the range $(-1, +1)$, and so that at points a unit distance away from the solution, F is approximately a unit value greater than at the minimum. It is unlikely that you will be able to follow these recommendations very closely, but it is worth trying (by guesswork), as sensible scaling will reduce the difficulty of the minimization problem, so that e04ly will take less computer time.

9 Example

e04ly_funct2.m

```
function [fc, gc, user] = funct2(n, xc, user)
  gc = zeros(n, 1);
  fc = (xc(1)+10*xc(2))^2 + 5*(xc(3)-xc(4))^2 + (xc(2)-2*xc(3))^4 + ...
       10*(xc(1)-xc(4))^4;
  gc(1) = 2*(xc(1)+10*xc(2)) + 40*(xc(1)-xc(4))^3;
  gc(2) = 20*(xc(1)+10*xc(2)) + 4*(xc(2)-2*xc(3))^3;
  gc(3) = 10*(xc(3)-xc(4)) - 8*(xc(2)-2*xc(3))^3;
  gc(4) = 10*(xc(4)-xc(3)) - 40*(xc(1)-xc(4))^3;
```

e04ly_hess2.m

```
function [heslc, hesdc, user] = hess2(n, xc, lh, user)
  heslc = zeros(lh, 1);
  hesdc = zeros(n, 1);

  hesdc(1) = 2 + 120*(xc(1)-xc(4))^2;
  hesdc(2) = 200 + 12*(xc(2)-2*xc(3))^2;
  hesdc(3) = 10 + 48*(xc(2)-2*xc(3))^2;
  hesdc(4) = 10 + 120*(xc(1)-xc(4))^2;
  heslc(1) = 20;
  heslc(2) = 0;
  heslc(3) = -24*(xc(2)-2*xc(3))^2;
  heslc(4) = -120*(xc(1)-xc(4))^2;
  heslc(5) = 0;
  heslc(6) = -10;
```

```
ibound = int32(0);
bl = [1;
     -2;
     -1000000;
     1];
bu = [3;
     0;
     1000000;
     3];
x = [3;
     -1;
     0;
     1];
[blOut, buOut, xOut, f, g, user, ifail] = ...
    e04ly(ibound, 'e04ly_funct2', 'e04ly_hess2', bl, bu, x)
```

```
Warning: e04ly returned a non-zero warning or error indicator (5)
```

```
blOut =
           1
          -2
    -1000000
           1
buOut =
           3
           0
     1000000
           3
xOut =
    1.0000
   -0.0852
    0.4093
    1.0000
f =
    2.4338
g =
    0.2953
   -0.0000
    0.0000
    5.9070
user =
     0
ifail =
           5
```