NONSMOOTH OPTIMIZATION: USE OF THE CODE DYNEPS

Claude Lemarechal

March 1980
WP-80-36

INTERNATIONAL INSTITUTE FOR APPLIED SYSTEMS ANALYSIS
A-2361 Laxenburg, Austria

FOREWORD

One of the aims of the Optimization Task of the System and Decision Sciences Area is to provide computer codes that help to solve certain numerical problems.

This paper describes the use of such a code which is being used successfully on a number of IIASA problems, in particular for the Food and Agriculture Program and Human Settlements and Services.

NONSMOOTH OPTIMIZATION: USE OF THE CODE DYNEPS

Claude Lemarechal
INRIA-Rocquencourt, 78150 Le Chesnay, France

1. GENERALITIES

This code is a technical improvement of the code CONWOL, and its role is also to minimize a function f(x) without constraints, i.e.,

$$\text{find } x^* \text{ in the n-dimensional space } R^n \text{ such that} \qquad (1)$$
$$f(x^*) \leq f(x) \quad \text{for any} \quad X \text{ in } R^n \quad .$$

It is assumed that, given x, one can compute f(x) and the gradient g(x); however, g(x) is not assumed to vary continuously when x varies. Thus, the possible applications for DYNEPS could be:

- -- when f is known to be kinky
- -- when the differentiability properties of f are not exactly known
- -- when there are some constraints in the problem that are introduced in the objective function through a penalty term.

The code is only semiexperimental in the sense that dimensionments are static, printouts are schematic, etc. However,

it should be fairly reliable, and possible difficulties normally come from incorrect use of the code, rather than deficiencies of the code itself.

The method is iterative and constructs a sequence of "trial solutions" x_k, $k = 1,2,\ldots, K$, and returns some x_K which is hopefully a good approximation of x^*. More specifically, the algorithm aims at obtaining approximate optimality conditions of the type

$$\min f \geq f(x_k) - \varepsilon_k \quad . \tag{2}$$

It starts with big ε_1 (given by the user), reduces ε_k when an estimate such as (2) is obtained and stops when (2) is obtained with $\varepsilon_k \leq \varepsilon_0$, where ε_0 is the final tolerance, also given by the user.

Note that an estimate such as (2) supposes that f is convex. However, even in this case it cannot be obtained and the algorithm strives to approximate it by:

$$f(y) \geq f(x_k) - \varepsilon_k - \sqrt{\eta}\,\|y-x_k\| \quad , \quad \text{for any } y \text{ in } R^n, \tag{3}$$

where η is another tolerance given by the user; it plays the part of the squared norm of the gradient in the smooth case.

The computation of x_{k+1} from x_k is called an iteration and is done in two successive steps:

-- first, compute a direction d_k in R^n; this is done in the subroutine GAUCHE. It is a rather complicated process, which involves $g_1,\ldots, g_k$, the gradients computed in iterations $1,\ldots, k$.

-- second, compute a stepsize $t_k > 0$; this is done in the subroutine LIGNE.

Then two cases may occur. If LIGNE has found that $f(x_k+t_kd_k)$ is less than $f(x_k)$ by a definite amount, then x_k is normally updated to $x_{k+1} = x_k + t_kd_k$. Otherwise x_{k+1} is kept as x_k, and only a new gradient is used to compute d_{k+1}.

2. THE SUBROUTINE CALCUL

The first thing the user has to do when using DYNEPS is to provide a fortran subroutine to compute function and gradient values. This subroutine must have the following form:

```
SUBROUTINE CALCUL (X,G,F)

DIMENSION X(1), G(1)  .
```

X is the value of the vector of variables at which f and g must be computed, G is the value of the gradient at X, f is the value of the function.

Thus, other information essential for CALCUL (such as N, the number of variables) must be passed on through some COMMON block to be shared between the main program which calls DYNEPS and the (possibly many) subroutines which help characterize the problem to be solved.

3. THE CALLING SEQUENCE

```
CALL DYNEPS (X, F, EPS, EPSO, ETA, ZERO, FMIN, IMP,
 N, G, NMAX, ITMAX, NAPMAX)
```

where the parameters are:

X (Input-Output), a vector of dimension n.
  Input: the initial values of the variables given by the user when calling DYNEPS.
  Output: the final variables returned by DYNEPS.

F (I-O), a scalar. Same meaning but concerning function values.

EPS (I-O), an initial guess to get (2). A fraction of $f(x_1)$ - min f is a reasonable value. The choice of EPS affects only the first iteration. EPS is modified by DYNEPS.

EPSO (I), the final value wished by the user in the bound (3).

ETA (I), the tolerance in (3). It is homogenous to the square norm of the gradient, and a peculiarity of DYNEPS is

that very small values for ETA are acceptable. If Δf and Δx are of the same order of magnitude (i.e., gradients close to unity) it is not unreasonable to ask for ETA in the range 10^{-12} on PDP11.

ZERO (I). The machine precision; because the program is written in single precision, it is approximately 10^{-6} on PDP11.

FMIN (I). A safeguard to prevent unbounded solutions. The program stops if some X is found such that $F(x) \leq$ FMIN.

IMP (I). Controls the printouts. The amount of printouts is an increasing function of IMP. If IMP $\leq$

0 nothing is printed

1 something very short is printed at each iteration

2 some more information is printed at each iteration (mainly useful for the designer of the algorithm).

3 information is printed during executions of LIGNE; very useful to check the computation of the gradient (see Section 4).

IMP > 3 dumps the execution of GAUCHE and should never be used.

N (I), number of variables.

G (I-O), a vector of dimension n.

Input: the gradient of f at the initial value of x.

Output: no meaning.

NMAX (I), controls the core requirement. Because GAUCHE uses g_i, i = 1,..., k at iteration k, the amount of core required by the algorithm is theoretically infinite. Therefore, when the number of gradients is going to exceed NMAX, a cleaning up is made to keep a number of gradients no larger than NMAX. NMAX should be reasonably large (say at least 10).

ITMAX (I). Maximum number of iterations, i.e., DYNEPS stops when k = ITMAX.

NAPMAX (I). Maximum number of calls to the subroutine CALCUL.

4. WARNINGS AND HINTS

Do not forget to call CALCUL before entering DYNEPS, in order to properly initialize F, G, and possibly EPS.

Check that the internal dimensions are sufficient. One must have:

in DYNEPS	Dim. of Q $\geq$ N*(NMAX - 1)
	Dim. of S $\geq$ N
	Dims of EPSN, AL, JC $\geq$ NMAX
in LIGNE	Dim. of x $\geq$ N
in GAUCHE	Dim. of R $\geq$ (NMAX, NMAX)
	Dims. of RR, x, y, w1, w2, A, E, JC, IC $\geq$ NMAX

In its present form, the program accepts N $\leq$ 50 and NMAX $\leq$ 20.

In case of difficulty, if the calling sequence is correct and if all the DIMENSION statements are large enough, then there is a 99% probability that the gradient is badly computed in CALCUL. To check it, run with IMP = 3. Then, at each iteration, a line is printed at each call of CALCUL. The following notations are used:

FK is $f(x_k)$, the initial value, at o-stepsize, for the line-search.

F is $f(x_k+td_k)$, the objective function at the current stepsize t. The printed F - FK gives the change in f when x is changed from x_k to $x_k + td_k$.

D is the direction d_k, and (D,G) is the derivative with respect to t of the one-dimensional function $f(x_k + td_k)$.

Then, drawing the observed points of the graph of f and of its tangents should indicate if the derivative seems to agree with the function.

The standard cause of failure is when a sequence of stepsizes is produced going to zero (from the right), with $F - F_k$ decreasing down to zero, whereas the derivative (D,G) is constantly negative. The user must then judge whether this is due to round off errors or to gross blunders in CALCUL.

5. AN ILLUSTRATIVE EXAMPLE

For demonstrative purposes, we will show the printout of a run where the function to be minimized is MAXQUAD, as described in "A set of nonsmooth optimization test-problems" (in "Nonsmooth optimization" Lemarechal and Mifflin, eds., IIASA Proceedings Series Volume 3, Pergamon Press).

The subroutine CALCUL contains a mistake that has been purposely introduced in the computation of the gradient. Instead of

$$Z = z + 2. * a(k_0,i,j) + x(j) \quad ,$$

we have written

$$Z = z + a(k_0,i,j) + x(j) \quad .$$

The printout with IMP = 1 is given below.

```
  1     1    f=  0.5337068e 04    eps= 0.100e 02
  2     2    f=  0.1623051e 03    eps= 0.100e 02
  3     5    f=  0.9297476e 02    eps= 0.100e 02
  4     6    f=  0.4032729e 02    eps= 0.100e 02
  5     9    f=  0.1132498e 02    eps= 0.100e 02
  6    12    f=  0.5169246e 01    eps= 0.100e 02
  7    16    f=  0.4694672e 01    eps= 0.134e 01
  8    17    f=  0.4299792e 01    eps= 0.949e 00
  9    20    f=  0.1433475e 01    eps= 0.316e 01
 10    23    f=  0.1433475e 01    eps= 0.316e 01
f= 0.14330558e 01     ...fin anormale
```

It gives for each iteration: the number of iteration, the number of calls to CALCUL made so far, the current value of the objective function, and of the convergence parameter EPS (which is supposed to reduce down to EPSO).

Then we show the printout with IMP = 3. At the tenth iteration we see that, when the stepsize is close to the optimal stepsize, the derivative is frankly negative. This is enough to stop the algorithm.

```
1     1    f=  0.5337068e 04   eps= 0.100e 02
          (d,d)=  0.164e 09  extra cout= 0.000e 00
                                              -(d,g0)= 0.164e 09
          t initial 0.78074365e-04  f-fk=-0.517e 04   (d,g)= 0.196e 07
           t=  0.781e-04   logic= 3

2     2    f=  0.1623051e 03   eps= 0.100e 02
          (d,d)=  0.218e 05  extra cout= 0.231e 03
                                              -(d,g0)= 0.241e 05
          t initial 0.42885983e 00  f-fk= 0.773e 06   (d,g)= 0.182e 07
          t diminue 0.42885985e-01  f-fk= 0.672e 05   (d,g)= 0.175e 07
          t diminue 0.42885984e-02  f-fk=-0.693e 02   (d,g)=-0.106e 03
           t=  0.429e-02   logic= 3

3     5    f=  0.9297476e 02   eps= 0.100e 02
          (d,d)=  0.929e 03  extra cout= 0.000e 00
                                              -(d,g0)= 0.929e 03
          t initial 0.14928854e 00  f-fk=-0.526e 02   (d,g)= 0.596e 03
           t=  0.149e 00   logic= 3

4     6    f=  0.4032729e 02   eps= 0.100e 02
          (d,d)=  0.351e 03  extra cout= 0.979e 01
                                              -(d,g0)= 0.449e 03
          t initial 0.23454417e 00  f-fk= 0.162e 03   (d,g)= 0.145e 04
          t diminue 0.36990784e-01  f-fk=-0.245e 02   (d,g)=-0.268e 03
           interpol 0.56746125e-01  f-fk=-0.290e 02   (d,g)= 0.197e 02
           t=  0.567e-01   logic= 3

5     9    f=  0.1132498e 02   eps= 0.100e 02
          (d,d)=  0.835e 02  extra cout= 0.794e 01
                                              -(d,g0)= 0.163e 03
          t initial 0.35599217e 00  f-fk= 0.172e 03   (d,g)= 0.679e 03
          t diminue 0.35599217e-01  f-fk=-0.589e 01   (d,g)=-0.158e 03
           interpol 0.67638516e-01  f-fk=-0.616e 01   (d,g)= 0.486e 02
           t=  0.676e-01   logic= 3

6    12    f=  0.5169246e 01   eps= 0.100e 02
          (d,d)=  0.135e 02  extra cout= 0.000e 00
                                              -(d,g0)= 0.135e 02
          t initial 0.91076344e 00  f-fk= 0.132e 03   (d,g)= 0.141e 03
          t diminue 0.91076344e-01  f-fk=-0.547e 00   (d,g)=-0.532e 01
           interpol 0.17304507e 00  f-fk= 0.467e 01   (d,g)= 0.489e 02
           interpol 0.99273220e-01  f-fk=-0.475e 00   (d,g)= 0.109e 02
           t=  0.993e-01   logic= 3

7    16    f=  0.4694672e 01   eps= 0.134e 01
          (d,d)=  0.179e 02  extra cout= 0.567e 01
                                              -(d,g0)= 0.255e 02
          t initial 0.37278481e-01  f-fk=-0.395e 00   (d,g)= 0.250e 02
           t=  0.373e-01   logic= 3
```

```
 8    17    f=  0.4299792e 01    eps= 0.949e 00
          (d,d)=  0.273e 02  extra cout= 0.978e 01
                                                  -(d,g0)= 0.366e 02
          t initial 0.21601800e-01  f-fk=-0.138e 01   (d,g)=-0.316e 02
          t grandit 0.43203600e-01  f-fk=-0.229e 01   (d,g)=-0.125e 02
          t grandit 0.86407200e-01  f-fk=-0.287e 01   (d,g)=-0.403e 01
           t=  0.864e-01   logic= 3

 9    20    f=  0.1433475e 01    eps= 0.316e 01
          (d,d)=  0.280e 00  extra cout= 0.000e 00
                                                  -(d,g0)= 0.280e 00
          t initial 0.20460793e 02  f-fk= 0.103e 04   (d,g)= 0.497e 02
          t diminue 0.20460794e 01  f-fk= 0.117e 02   (d,g)= 0.518e 01
          t diminue 0.20460795e 00  f-fk= 0.145e 00   (d,g)= 0.110e 00
           t=  0.205e 00   logic= 2

10    23    f=  0.1433475e 01    eps= 0.316e 01
          (d,d)=  0.179e 00  extra cout= 0.000e 00
                                                  -(d,g0)= 0.179e 00
          t initial 0.25611925e 00  f-fk= 0.678e-01   (d,g)=-0.176e 00
          t diminue 0.25611926e-01  f-fk=-0.391e-03   (d,g)=-0.456e 00
          t diminue 0.35686791e-02  f-fk=-0.151e-03   (d,g)=-0.483e 00
           interpol 0.84861079e-02  f-fk=-0.306e-03   (d,g)=-0.477e 00
           interpol 0.12222219e-01  f-fk=-0.385e-03   (d,g)=-0.472e 00
           interpol 0.15094112e-01  f-fk=-0.422e-03   (d,g)=-0.469e 00
           interpol 0.17321056e-01  f-fk=-0.439e-03   (d,g)=-0.466e 00
           interpol 0.19058086e-01  f-fk=-0.443e-03   (d,g)=-0.464e 00
           interpol 0.20420330e-01  f-fk=-0.440e-03   (d,g)=-0.462e 00
           interpol 0.21493299e-01  f-fk=-0.435e-03   (d,g)=-0.461e 00
           interpol 0.22340419e-01  f-fk=-0.430e-03   (d,g)=-0.460e 00
           interpol 0.23010075e-01  f-fk=-0.423e-03   (d,g)=-0.459e 00
           interpol 0.23541436e-01  f-fk=-0.419e-03   (d,g)=-0.459e 00
           interpol 0.23119440e-01  f-fk=-0.422e-03   (d,g)=-0.459e 00
           interpol 0.23206325e-01  f-fk=-0.421e-03   (d,g)=-0.459e 00
           interpol 0.23275629e-01  f-fk=-0.421e-03   (d,g)=-0.459e 00
           interpol 0.23330217e-01  f-fk=-0.420e-03   (d,g)=-0.459e 00
           interpol 0.23373697e-01  f-fk=-0.420e-03   (d,g)=-0.459e 00
           interpol 0.23407992e-01  f-fk=-0.420e-03   (d,g)=-0.459e 00
           interpol 0.23435395e-01  f-fk=-0.420e-03   (d,g)=-0.459e 00
           interpol 0.23457082e-01  f-fk=-0.419e-03   (d,g)=-0.459e 00
           interpol 0.23439452e-01  f-fk=-0.420e-03   (d,g)=-0.459e 00
           interpol 0.23442416e-01  f-fk=-0.420e-03   (d,g)=-0.459e 00
           interpol 0.23445072e-01  f-fk=-0.419e-03   (d,g)=-0.459e 00
           t=  0.234e-01   logic= 1
f= 0.14330558e 01    ...fin anormale
```

Finally we show the printout with IMP = 1, when the mistake in CALCUL is removed.

```
 1    1   f=  0.5337068e 04   eps= 0.100e 02
 2    2   f=  0.1622138e 03   eps= 0.100e 02
 3    5   f=  0.8726418e 02   eps= 0.100e 02
 4    6   f=  0.7060697e 01   eps= 0.100e 02
 5    9   f=  0.5644329e 01   eps= 0.644e 01
 6   18   f=  0.1785346e 01   eps= 0.434e 01
 7   20   f=  0.6918660e 00   eps= 0.284e 01
 8   21   f=  0.4191631e 00   eps= 0.219e 01
 9   22   f=  0.2481052e 00   eps= 0.545e 00
10   24   f=  0.1365424e 00   eps= 0.127e 00
11   26   f= -0.5789071e 00   eps= 0.100e 01
12   30   f= -0.7109213e 00   eps= 0.211e 00
13   35   f= -0.7159269e 00   eps= 0.991e-02
14   38   f= -0.7197158e 00   eps= 0.301e-01
15   44   f= -0.7979828e 00   eps= 0.887e-01
16   46   f= -0.8346893e 00   eps= 0.611e-01
17   48   f= -0.8346893e 00   eps= 0.611e-01
18   49   f= -0.8360314e 00   eps= 0.365e-02
19   57   f= -0.8395631e 00   eps= 0.382e-02
20   61   f= -0.8401815e 00   eps= 0.806e-03
21   65   f= -0.8402191e 00   eps= 0.100e-03
22   68   f= -0.8411544e 00   eps= 0.178e-02
23   71   f= -0.8411846e 00   eps= 0.100e-03
24   74   f= -0.8412031e 00   eps= 0.131e-03
25   79   f= -0.8412591e 00   eps= 0.127e-03
```

```
 26    95    f= -0.8413643e 00    eps= 0.156e-03

 27    97    f= -0.8413853e 00    eps= 0.100e-03

 28   100    f= -0.8413895e 00    eps= 0.100e-03

 29   101    f= -0.8413895e 00    eps= 0.100e-03

 30   102    f= -0.8413895e 00    eps= 0.100e-03

 31   103    f= -0.8413895e 00    eps= 0.100e-03

 32   104    f= -0.8413895e 00    eps= 0.100e-03
error from gauche. at entry, the old solution is optimal
f=-0.84138954e 00     ...fin anormale
```

Now some trouble appears in the computation of the direction. Because the subprogram that computes this direction is fairly reliable, the trouble must be due to rounding off. This is confirmed by the fact that we have used ETA = 10^{-10}, whereas the squared norm of the gradient in the neighborhood of the solution is in the range 10^{4}.