

Use of the Code N1CV2

Claude Lemaréchal and Claudia Sagastizábal

Inria – BP 105 – 78153 Le Chesnay; Tel. 01 39 63 55 11; Fax – – 57 86

e-mail Claude.Lemarechal@inria.fr, Claudia.Sagastizabal@inria.fr

Reading these notes supposes that the notes “Using a MODULOPT minimization code” have been read and understood, and that SIMUL has been written.

The code N1CV2 minimizes on the whole space a function without the usual differentiability assumptions. The objective function f must be convex. The user must compute (in SIMUL) the function value $f(x)$ and some (arbitrary) subgradient $g(x)$, for given x . Thus, the situation is formally the same as in classical (smooth) unconstrained optimization; the code is therefore a “minimizer of class 1”. Note that it is written in double precision version: the single precision version does not exist.

The algorithm is described in C.Lemaréchal, C.Sagastizábal: ‘Variable metric bundle methods: from conceptual to implementable forms’, *Mathematical Programming* 76,3(1997) 393-410.

Calling sequence

```
CALL N1CV2 (SIMUL,PROSCA,N,X,F,G,DX,DF1,EPS,ZERO,IMP,IO,MODE,  
           NBUN,ITER,NSIM,MEMAX,IZ,NIZ,DZ,NDZ,IZS,RZS,DZS)
```

SIMUL (External) Entry to the simulator computing function-gradient (see MODULOPT norms). N1CV2 always requires function *and* gradient together; in other words, it never calls the simulator with INDIC=2 nor 3.

PROSCA (External) Entry to subroutine computing scalar product (see below).

- N** (Input) Number of variables.
- X** (Input-Output) Vector of variables.
- F** (Input-Output) Corresponding values of $f(x)$. Note that **SIMUL** must be called prior to **N1CV2**, so that **F** is initialized, as well as **G** below.
- G** (Input-Output)
- Input: gradient at **X**-input
- Output: usually the last convex combination of subgradients, that gave the last direction (this may not be true in case of abnormal end).
- DX** (Input) Minimal resolution on **X**. This means that the user considers two points x and y as indistinguishable if they have $|x_i - y_i| \leq \text{DX}$ for all i .
- DF1** (Input) a positive number: the expected change in f at the first iteration. Used to estimate the initial stepsize and an appropriate stopping criterion. Gross estimate is enough; set it for example to (an estimate of) the total decrease from starting point to optimum.
- EPS** (Input) Required accuracy on f (in absolute value).
- ZERO** (Input) The machine precision (say 10^{-12} since the code is written in double precision).
- IMP** (Input) Controls the printouts; they are an increasing function of **IMP**
- = 0 nothing is printed
 - ≤ 1 initial and final printouts
 - ≤ 2 printout every time the bundle is reduced
 - ≤ 3 printout at each iteration
 - ≤ 4 prints detail of the line-search
 - > 4 debugging for the authors; maximal value is **IMP** = 8
 - < 0 **SIMUL** is called with **INDIC** = 1 every -**IMP** iteration
- IO** (Input) The i.o. channel for **IMP**.

MODE (Output) Status of the return

< 0 **SIMUL** has answered **INDIC** <0 in such a way that the algorithm could not proceed further

= 0 **SIMUL** has answered **INDIC** = 0 at the **X** output

= 1 normal end, the stopping criterion has been met

= 2 something wrong in the calling sequence: **DX** or **DF1** is ≤ 0 etc.

= 3 not used

= 4 maximal number of iterations reached

= 5 maximal number of calls to **SIMUL** reached

= 6 minimal resolution **DX** reached

= 7 something wrong with the quadratic program computing the direction (usually, this means that **EPS** and **DX** are too small, in view of the machine precision).

= 9 **MEMAX**=1, why don't you use steepest descent?

NBUN (Output) Number of elements in the final bundle.

ITER (Input-Output)

Input: maximal number of iterations.

Output: actual number of iterations done when returning.

NSIM (Input-Output)

Input: maximal number of calls to **SIMUL** (cumulated along iterations).

Output: actual number of such calls to **SIMUL** done (cumulated).

MEMAX (Input) maximal number of subgradients to store in the bundle. Must be ≥ 1 (for **MEMAX** = 1, we obtain the gradient method; for **MEMAX** = 2, it is a sort of conjugate gradient).

Normally, efficiency should be an increasing function of **MEMAX**.

IZ, NIZ, DZ, NDZ Integer and Real*8 working spaces for **M1CV2**. **NIZ** and **NDZ** are the dimensions allotted respectively to **IZ** (integer array) and **DZ** (real*8 array) in the calling program. They must be as follows:

At least MEMAX for IZ; this means that the calling program must include the statement

```
DIMENSION IZ(NIZ)
```

where NIZ is an integer at least equal to MEMAX.

As for DZ, the value MEMAX*(MEMAX+N+8)+5*N+10 is enough; in case of emergency, the really minimal value is (with $k = \min\{\text{MEMAX}, N+1\}$):

$$\frac{\text{MEMAX}*(\text{MEMAX}+2*N+5)}{2} + \frac{k(k+11)}{2} + 10.$$

IZS,RZS,DZS Working space for the simulator (see MODULOPT norms).

The subroutine PROSCA

Computes the scalar product $\langle \cdot, \cdot \rangle$ of the space in which the gradient is defined. This means that, if g is computed by SIMUL, and if dx is a differential of x , then the corresponding differential of the objective function, as computed by PROSCA, is $df = \langle g, dx \rangle$.

This subroutine must be of the form

```
SUBROUTINE PROSCA(N,X,Y,PS,IZS,RZS,DZS)
```

in which the notation is self-understanding.

When G is the ordinary vector of partial derivatives, i.e. when $\langle \cdot, \cdot \rangle$ is the ordinary dot-product, then PROSCA can have the following form:

```

SUBROUTINE FUCLID (N,X,Y,PS,IZS,RZS,DZS)
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
DIMENSION X(N),Y(N),IZS(*),RZS(*),DZS(*)
REAL RZS(*)
PS = 0.DO
DO 10 I = 1,N
10 PS = PS + X(I)*Y(I)
RETURN
END
```

which is included in the N1CV2 package. Note that FUCLID must be declared EXTERNAL.