APW78 METHOD SOLVING NONCONVEX NONLINEAR PROGRAMMING PROBLEMS--USER'S GUIDE

Zenon Fortuna

February 1980
WP-80-22

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

CONTENTS

APW78 METHOD SOLVING NONCONVEX NONLINEAR PROGRAMMING PROBLEMS--USER'S GUIDE

INTRODUCTION

In the last decade a rapid growth of the number of algorithms for mathematical programming can be observed. This reflects a growing interest in applications of computer aids to both research and project stages of scientific work, and certainly has a positive influence on the effectiveness and simplicity of these algorithms, but this development also has negative consequences. Many researchers are simply lost in the variety of existing methods of mathematical programming, and their attempts to verify some new suggestions by means of testing examples can very often be interrupted by difficulties which arise in preparing the problem, choosing a particular method (or, most often, methods) from an existing computer library, and then in discussing the outputs obtained.

This paper presents a new method, entirely developed at IIASA (Wierzbicki 1978) and implemented on the IIASA computer CDC 11. It is not the aim of this paper to develop further theoretical justificaiton of effectiveness of this new algorithm (the reader interested in details is directed to Wierzbicki 1978) but contains a description of the implemented algorithm and gives sufficient information for easy application of this algorithm. The reader not acquainted with numerical methods of mathematical programming can follow the general instructions given in Basic Instructions and, after confirming his understanding in An Example, can easily attempt to solve his problem. In some cases obtained outputs might not be sufficiently clear--then Extended Description of Output Messages should provide the reader with the desired information, and Detailed Description of In-Line Searching Procedure could help him in his own modifications of the program, which might be necessary in the case of particulary difficult problems.

BASIC INSTRUCTION

The algorithm solves a general problem of mathematical programming of the form:

minimize $F(X)$

with respect to *variable* $X \in X_0 \subset R^N$

under *constraints* $X_0 = \{X \in R^N : GH_i(X) \leq 0,\ i=1,\ldots,\ LG$

$GH_i(X)=0,\ i=LG+1,\ldots,\ LG+LH\}$.

This problem does not necessarily have a solution and usually some sufficient assumptions should be fulfilled to assure its existence. Also the algorithm is efficient only under some regularizing assumptions. The reader interested in these details is directed to Wierzbicki (1978). For practical applications, however, it is sufficient to deal with a problem with accessible first derivatives, and with (not explicitly given) continuous second derivatives.

The algorithm has been written in FORTRAN, and has been implemented under the password APW78. This algorithm is kept on a magnetic tape accessible in the System and Decision Sciences Area. The user is obliged to follow the following instruction:

1. Write the subroutine "SUB.f4p" described below containing a description of the problem.
2. Write data consisting of parameters, dimensions, starting points, etc.
3. Make a compilation of "SUB.f4p" obtaining "SUB.obj".
4. Make a linkage between "APW78.obj" and "SUB.obj", obtaining "APW78.out".
5. Run a program "APW78.out" with the written data.

ad.1

The file with a subroutine SUB can be obtained as follows:

```
% ed SUB. f4p          (+ carriage return)
? file                 (*)
f
SUB.f4p                (*)
a                      (+ c.r.)
```

```
[ contents of SUB.f4p

.               (+ c.r.)

nnn             (*)

q               (+ c.r.)

%               (*)
```

The above description, which repeats both user and computer (denoted by (*)) lines, is given to allow even the most inexperienced programmer to make use of APW78.

The contents of SUB.f4p (the part in brackets) has the form:

```
      SUBROUTINE sub(X,LB)
      IMPLICIT DOUBLE PRECISION (B,G,X)
      DIMENSION X(𝒩)
      COMMON /fgh/F, GH(ℒ𝒢 + ℒℋ)
      COMMON/bfgh/B(𝒩·(ℒ𝒢 + ℒℋ + 1))
      COMMON /counter/N1,N2
      IF(LB.EQ.-1) GO TO 1
      N1 = N1 + 1
      F = ... (arithm. expression for F(X))
      GH(1) = ... (arithm. expression for GH1(X))
      ⋮
      GH(ℒ𝒢 +ℒℋ) = ... (arithm. expression for GHLG+LH(X))
      IF(LB.EQ.Ø) RETURN
    1 N2 = N2 + 1
      B(1) = ... (arithm. expression for ∂F/∂X1 )
      ⋮
      B(𝒩)= ... (arighm. expression for ∂F/∂XN
      B(𝒩 + 1) = ... (arithm. expression for ∂GH1)/∂X1)
      ⋮
      ⋮
```

B($\mathcal{N}$ · ($\mathcal{LG}$ + $\mathcal{LH}$ + 1)) = ... (arithm. expression for

$$\frac{\partial GH_{LG+LH}}{\partial X_N}$$

```
      N2 = N2 + 1

      RETURN

      END
```

Comment: Symbols $\mathcal{N}$, $\mathcal{LG}$ and $\mathcal{LH}$ denote explicitly written values of N, LG, and LH.

ad.2

Data can be supplied directly from a terminal after giving a command in point 5:

```
      % APW78.out          (+ c.r.)
```

but this data can also be prepared in a file with a name, say, DATA, and in point 5 we will write a command

```
      % APW78.out < DATA (+ c.r.)
```

An order in prepared data is sufficiently clarified by an explanation of how to prepare a file DATA:

```
      % ed DATA            (+ c.r.)

      ? file               (*)

      f                    (+ c.r.)

      DATA                 (*)

      a                    (+ c.r.)

      [ contents of DATA

      .                    (+ c.r.)

      w                    (+ c.r.)

      nnn                  (*)

      q                    (+ c.r.)

      %                    (*)
```

1st line:

$\mathcal{N}$, $\mathcal{MG}$, $\mathcal{MH}$, 500

explanation: These numbers are read in a format 4i3. Number 500 means that APW78 obtained working space of a 500 double precision number. The user cannot change this value without changing a proper value in a COMMON/space/×(500) in the APW78 program.

2nd line (ev. next lines):

x_1, x_2, ..., x_5
x_6, x_7,...
...
... $x_{\mathcal{N}}$

explanation: These numbers are read in a format 5f10.2. They define an initial evaluation of the solution (starting point), which should be given by the user (at least it is allowed to put 0., 0., ... there). It should be stressed that better or worse evaluation of this starting point can have a strong influence on both computation time and (in some cases) the effectiveness of an algorithm at all.

3rd line:

ip, itmax

explanation: These integer numbers are read in a format 2i3. "ip" should be equal to 1, 2, 3, 4, 5, or 6, and means an increasing level of outputs (during computations). "itmax" is a maximal number of iterations given by the user.

4th line:

n (or) y

explanation: In this line a single letter should be given indicating that we do not want to supply the program with additional information ("n" ≡ not), or we want to change some (usually given automatically) values.

4th-bis line:

ex, ey

explanation: This line should be given only in the case when a letter "y" has been given in line 4. These numbers are read in a format 2f10.5 and mean values of norms of Lagrangian's gradients with respect to x and y, respectively, which decide when the first iteration can be regarded as successful. The automatic choice of these values ("n" in the 4th line): 0.1, 0.1.

5th line:

n (or) y

explanation: If you accept the starting value of penalty coefficient $\rho = 0.2$, put "y". If not, write "n" and supply line 5th-bis.

5th-bis line:

ro

explanation: This number is read in a format f10.4.

6th line:

y

explanation: You told the program "yes, you can start".

ad.3

The compilation of SUB.f4p can be done by either an f4p or an ftn compiler. You can do this easily by taking a file called "for" from a tape on which APW78 is recorded, and writing

```
% for SUB.f4p          (+ c.r.)
```

You should obtain the answer:

```
none of the errors detected     (*)

Ø Errors Detected               (*)

%
```

ad.4

You can obtain a linkage between APW78.obj and SUB.obj by giving a command:

```
% linker -i APW78.obj SUB.obj -l -c      (+ c.r.)
```

and you should obtain the answer:

```
CULC Linkage Editor       (*)

Ø Errors Detected         (*)

%                         (*)
```

ad.5

Now you have an executable program APW78.out and you will obtain a solution by writing

```
% APW78.out < DATA        (+ c.r.)
```

or giving DATA from a terminal after writing

```
% APW78.out        (+ c.r.)
```

AN EXAMPLE

minimize $(x_1^2 + x_2^2)$

under constraints $x_1 + x_2 \leq -2$, $x_1 = x_2$.

We have:

N = 2, LG = 1, LH = 1, and

```
      SUBROUTINE sub(X,LB)
      IMPLICIT DOUBLE PRECISION (B,G,X)
      DIMENSION X(2)
      COMMON /fgh/F, GH(2)
      COMMON /bfgh/B(6)
      COMMON .counter/N1, N2
      IF (LB.EQ.-1) GO TO 1
      N1 = N1 = 1
      F = X(1)**2 + X(2)**2
      GH(1) = X(1) + X(2) + 2
      GH(2) = X(1) - X(2)
      IF(LB.EQ.Ø) RETURN
    1 B(1) = X(1) + X(1)
      B(2) = X(2) + X(2)
      B(3) = 1.
      B(4) = 1.
      B(5) = 1.
```

```
      B(6) = -1.

      N2 = N2 + 1

      RETURN

      END
```

DATA:

```
      2, 1, 1, 500

      -10., 10.

      1, 5

      n

      n

      y
```

Obtained outputs:

```
      WELL, NOW GIVE ME NUMBERS N, LG, LH, INI
      IN A FORMAT 4I3, EG.:
      9, 1, 1, 5ØØ

      AND NOW GIVE ME INITIAL VECTOR X IN A FORMAT 5F10.2 EG.:
      1Ø., 2., Ø., Ø.1, 1.
      3., Ø., Ø,

      I HAVE TO ASK YOU ALSO HOW MANY ITERATIONS SHOULD
      I GIVE YOU AND WHAT IS THE MAXIMAL NUMBER OF ITERATIONS

      GIVE ME THIS INFORMATION AS NUMBERS IP AND ITMAX IN A
      FORMAT 2I3

      YOU MAY ALSO HAVE SOME OPTIONS.
      I SHALL LIST THEM NOW BELOW AND YOU WILL DECIDE
      WHETHER YOU WANT TO DECLARE THESE VALUES.
      IF YOU DO NOT, PRESS BUTTON "N", PLEASE.
      IF YES, PRESS "Y". AND I SHALL ASK YOU FOR DATA

           EX, EY?

      DO YOU WANT TO CHOOSE RØØ?
      PRESS BUTTON "Y" OR "N"

      I REPEAT:
      YOU GAVE ME
      N = 2  LG = 1  LH = 1  INI = 5ØØ
      EX = Ø.ØØE + ØØ  EY = Ø.ØØE + ØØ  RO = Ø.ØØE + ØØ
      IP = 1   ITMAX = 5
      X = -1Ø.ØØØØ   1Ø.ØØØØ
     NOW PUSH ME PRESSING "Y"  (Now begin by pressing "Y")
```

```
1  ITERATION AS PHASE I WITH EX = Ø.1ØE + ØØ  EY = Ø.1ØE + ØØ
     T = Ø.1ØØØE - Ø1
     T = Ø.4167E + ØØ

   RO INCREASED TO VALUE Ø.1ØE + Ø1

2  ITERATION AS PHASE I WITH EX = Ø.1ØE + ØØ  EY = Ø.10E + ØØ
     T = Ø.1ØØØE + Ø1
     T = Ø.27Ø3E + ØØ

3  ITERATION STARTS AS PHASE II
     EX CHANGED TO Ø.1ØE - Ø1  EY CHANGED TO Ø.1ØE - Ø1

S T O P
AFTER 3 ITERATIONS

      XFIN             GHFIN      C.VIOLFIN          YFIN
-Ø.1ØØØØØØØE + Ø1   Ø.ØØE + ØØ         Ø     Ø.2ØØØØØØØE + Ø1
-Ø.1ØØØØØØØE + Ø1   Ø.ØØE + ØØ        Ø!!    Ø.ØØØØØØØØE + ØØ

F = Ø.2ØØØØØØØE + Ø1
MAX CONSTR. VIOL. = Ø.ØØØØØØØØE + ØØ
ROØ Ø.1ØE + Ø1

     DURING COMPUTATIONS I NEEDED 7 VALUES OF F AND GH
     AND 5 VALUES OF DERIVATIVES B
```

Comments: The first part of the output messages (including the line "NOW PUSH...") has a meaning when we run a program giving data directly from a terminal. Numbers denoted by T are the step coefficients which are used during in-line minimization. THe two exclamation marks which are standing in column C.VIOLFIN (final constraints violations) indicate a constraint which is violated the most. In a case when there are some constraints with the same violation, these marks stand by the last of these constraints.

EXTENDED DESCRIPTION OF OUTPUT MESSAGES

Given a higher level of outputs (ip = 2,3,4,5, or 6) you will obtain the extended information about the minimization process. This information will be listed below and explained briefly.

1.
BEFORE UFO AUGLAG = ...
This is a value of augmented Lagrangian before unconstrained minimization.

2.
NGRADX = ... NGRADY = ...

These are l^2 norms of $\frac{\partial L}{\partial x}$ and $\frac{\partial L}{\partial y}$, where x is a primal and y is a dual variable.

3.
TAUØ = ...
This is the initial value of the step coefficient for the procedure MINIX (see next chapter).

4.
COS(D,G) = ...
This is a cosinus of an angle beyween a direction D and a gradient G. This value should be equal to -1. at the beginning (the steepest descent direction).

5
YØ = ... , PO = ... , DELTA = ...
These are values of a minimized functional, a scalar product <D,G> and a parameter δ in MINIX (see next chapter).

6.
D = ...
This a vector being the current direction of minimization.

7.
T = ...
This is a value of step coefficient.

8.
X = ...
This is a current point at which a value of a minimized functional is evaluated.

9.
Y, YOPT = ...
These are values of a minimized functional at the current point X which have already obtained the best (optimal) value in the current direction D.

10.
Z1, Z2 = ...
These are numbers corresponding to the two-side Goldstein test in MINIX (see next chapter).

11.
IT, INTO
These are indicators of what is a current position of the value of a minimized functional with respect to the Goldstein test (see next chapter).

12.
TMIN, TMAX, TOPT, KL
These are lower bound, upper bound, and current optimal values for the step coefficient T. Indicator KL might be helpful in following the algorithm of MINIX (with no importance for a user). Negative values of TMIN or TMAX mean that they are not yet known.

13.
TAU1 = ...
This is the best value of step coefficint T chosen by MINIX. TAU1 = 0. means that on given direction D MINIX did not find a better solution.

14.
X1, G1 = ...
X2, G2 = ...
These are coordinates of points and gradients taken for variable metric evaluation.

15.
DX, DG = ...
These are differences of X1 - X2 and G1 - G2.

16.
<XS - XN, GS - GS> = ..., KEY = ...
The first number should be positive--it is a necessary condition for the positive definitness of a variable metric.
KEY = 1 means that the variable metric will be used in further minimizations.
KEY = 0 means that the variable metric will be used as a part of the augmented Lagrangian's hessian.

17.
AL, A, BE, A = ...
These numbers correspond to details of the variable metric method which will not be described here.

18.
VARIABLE METRIC ON ix VECTORS
The integer number ix gives information on which number of differences DX and DG has been used for construction of a variable matrix.

19.
A NEW DATA NOT ACCEPTED
This means that obtained DX and DG cannot be used for actualizing the variable matrix.

20.
MATRIX = I
This message appears when the program starts again with steepest descent. This happens when, e.g., due to rounding errors
$\langle D,G \rangle \geq 0$.

21.
DIRMIN FAILED
This message indicates that MINIX cannot find a better solution even on the sttepest descent direction. Usually it happens in a very small neighborhood of the solution where some protections in MINIX (see next chapter) do not allow further searching. It can also happen, however, far from the optimum for strongly curved problems. In such cases a change in the starting point XØ is recommended.

DETAILED DESCRIPTION OF AN IN-LINE SEARCHING PROCEDURE

The subroutine which finds a subsequent point X in a given (by means of variable metric method) direction D is called MINIX. The principles of this algorighm are based on (Wierzbicki 1978). Generally, the computed step coefficient T should satisfy the two-side Goldstein test:

$$f(X) + T \cdot \langle D,G \rangle \cdot (1-\delta) \leq f(X+T \cdot D) \leq f(X) + T \cdot \langle D,G \rangle \cdot \delta \ ,$$

where δ is a given number from the interval $(\emptyset, \frac{1}{2})$. It has been assumed that $\delta = \emptyset.3$. The initial coefficint TAU$\emptyset$ is estimated by means of the rule

$$\text{TAU } \emptyset = (\min \frac{\Delta_0}{||G||} , 1.) \ ,$$

where Δ_0 is a step obtained in the last minimization. In the first iteration TAU$\emptyset$ = $\emptyset.\emptyset 1$. After evaluating f(X+TAU$\emptyset$·D), a quadratic approximation is performed (indicated by INTO = 2 in output messages). Afterwards, the step coefficient is either decreased or increased by an adapted factor and in a case when a condition (*) is satisfied, T is regarded as the final (optimal) value of a step coefficient. During computations the following numbers are evaluated:

$$Z1 = Y\emptyset - Y + T \cdot PO \cdot (1-\delta) \ ,$$

$$Z2 = Y\emptyset - Y + T \cdot PO \cdot \delta \ .$$

When two subsequent values TMIN and TMAX satisfy conditions Z1(TMIN) > 0 and Z2(TMAX) > 0, then T is computed as a geometrical mean value:

$$T = (TMIN \cdot TMAX)^{\frac{1}{2}} \ .$$

The searching process is stopped when one of the following conditions occurs:

1. Condition (*) is satisfied,
2. $T < 10^{20}$
3. $T < 10^{-20}$
4. $\Delta < 10^{-5} \cdot \Delta_0$, $\Delta = T \cdot ||DN||$
5. $|f_{i+1}-f_i| < 10^{-5} \cdot |f_i-f_{i-1}|$,
   where f_{i-1}, f_i, f_{i+1} are the last three values of f.

As a final value of T this value is given for which the value of f is minimal.

PLANNED DEVELOPMENTS

The proposed algorithm is comparable in its effectiveness with the methods described in Coope and Fletcher (1979) and Powell (1977). Several tests have been applied to check the features of the algorithm and the results are satisfactory. Similarly as in Coope and Fletcher (1979) and Powell (1977) it has been noticed that a quadratic programming algorithm used for solving an approximate problem (PHASE II) is a weak point of the whole algorithm. Actually APW78 uses a certain adaptation of an algorithm described in Panne (1975), but it is planned to replace it by a more efficient algorithm. Also a method used for constructing the approximation of a Lagrangian's hessian may probably be chosen more robustly (now is a certain version of symmetric rank-one correction), and, e.g., a method described in Fortuna (1978) will be tested.

An extended "conversational" version of the algorithm will be prepared. To make this possible the user interrupts and after analyzing plotter drawings characterizing the already known properties of the problem, he starts the method with changed coefficients. This version wil help the user to obtain a solution even for very irregular problems.

It should be stressed that the algorithm has not been thoroughly tested and is not error free. Any comments and suggestions directed to the author will help to improve the algorithm, and might make it very useful in many IIASA applications.

REFERENCES

Coope, I.D., and R. Flethcher (1979) Some Numerical Experience with a Globally Convergent Algorithm for Nonlinearly Constrained Optimization. NA/30 Report. University of Dundee.

Fortuna, Z. (1978) Superlinearly Convergent Secant Method. Math. Opt. Conference, Eisenach, GDR.

v.d. Panne, C. (1975) Methods for Linear and Quadratic Programming. Amsterdam: North Holland.

Powell, M.J.D. (1977) A Fast Algorithm for Nonlinearly Constrained Optimization Calculations. 1977 Dundee Conference on Numerical Analysis.

Wierzbicki, A.P. (1978) A Quadratic Approximation Method Based on Augmented Lagrangian Functions for Nonconvex Nonlinear Programming Problems. WP-78-61. Laxenburg, Austria: International Institute for Applied System Analysis.

Wierzbicki, A.P. (1978) Lagrangian Functions and Nondifferentiable Optimization. WP-78-63. Laxenburg, Austria: International Institute for Applied System Analysis.