JUAN AMUNATEGUI, M. Sc.
VINCENTIU DUMITRU, M. Sc.

Computer Experiments with an Unconstrained Nonlinear Program

I do not know what is untried and afterward,
But I know it will in its turn prove sufficient, and cannot fail

Walt Whitman-Song of myself *)

1. INTRODUCTION **)

Unconstrained non linear programming problems have, recently, attracted many mathematicians either as an absolute problem in itself or in relation with the solution of constrained non linear problems by sequential methods. The purpose of this paper is to give, in a first part, a brief presentation of some of the available methods for solving unconstrained non linear problems. In the second part a computer program, using the Fletcher and Reeves formula and a modification of this formula is described, as well as some results obtained with different functions. This program was submitted as partial fulfillment of the requirement for the course of Non Linear Programming given by Professor P. Wolfe at Columbia University. We feel that it might be also of interest, from the academic point of view, to show some example of an assignment given in a graduate course for the degree of Master of Science and Doctor of Engineering Science in an American university.

2. UNCONSTRAINED OPTIMIZATION METHODS

All the methods, described in this section, have in common the characteristic of using as step in the algorithm, a one-dimensional optimization.

---

*) We think that the Song of Walt Whitman is the song of every programmer

**) Acknoledgments: We are indepted to Professor P. Wolfe for his clarifying and penetrating presentation of these methods in his course at Columbia University. We are gratefull to Miss M. Lenard, Professor Wolfe's assistant, for granting us the permission of reproducing the explanatory notes on her subroutine STEP and the listing of the subroutine itself.

Considering the following notations:

x^k : given point at iteration k, of components $(x_1^k, x_2^k, ..., x_n^k)$

p_k : direction of the one-dimensional optimization

$f(x)$: function to be optimized

$\Delta f(x)$: gradient, vector of components $\frac{\partial f}{\partial x_1^k}, ..., \frac{\partial f}{\partial x_n^k}$

t : positive parameter

the algorithm in the case of a maximization will be:

0 — choose an initial x_0 and if necessary an initial direction p_0. set up $k = 0$

1 — choose $t = t_m$ such that:

$$max\ (f(x^k + t\ p_k) : t \geqq 0) \tag{1}$$

2 — compute:

$$x^{k+1} = x^k + t_m \cdot p_k \tag{2}$$

3 — compute $f(x^{k+1})$. If optimal, STOP; if not, choose a new direction p_{k+1} and go to 1.

2.1. **Optimal gradient method** (ref. 1 and 2)

This method presented first by Cauchy, chooses as direction p the gradient $\nabla f(x^k)$.

If the function f of (1) is differentiated with respect to t and set equal to zero:

$$\frac{\mathrm{d}\, f(x^k + tp_k)}{\mathrm{d}\, t} = \sum_{i=1}^{n} \frac{\partial f(x^k + t\, p_k)}{\partial x_i^k} \cdot (p_k)_i$$

$$= \nabla f(x^k + t\, p_k) \cdot p_k = 0$$

or

$$\nabla f(x^{k+1})\, \nabla f(x^k) = 0 \tag{3}$$

the two successive directions are therefore at right angle.

The convergence of the procedure has been proved by Curry for continuously differentiable functions.

If the function is a quadratic function and λ and Λ its smallest and largest eigen values the rate of convergence is bounded by:

$$\frac{f(x^{k+1})}{f(x^k)} \left(\frac{\Lambda - \lambda}{\Lambda + \lambda}\right)^2 \tag{4}$$

which is very slow if Λ is large compared to λ.

2.2. **Subrelaxation method** (ref. 7)

In step 2 of the algorithm the expression (2) is replaced by

$$x^{k+1} = x^k + r \cdot t_m \cdot p_k \tag{5}$$

where $0 < r < 1$ (from experiments 0.9 seems to be a good value)
This method is an improvement over the optimal gradient method.

2.3. **Diagonal method** (ref. 8)

Choose

$$p_k = x^k - x^{k-2} \tag{6}$$

after having used the optimal gradient method during the $k - 1$ previous iterations and the optimal gradient method is resumed for the next $k - 1$ iterations after the k th until (6) is used again. Experiments have given better results than the previous method.

2.4. **Method of Fletcher and Reeves** (ref. 4 and 9)

Choose the new direction as:

$$p_k = \nabla f(x^k) + \frac{|\nabla f(x^k)|^2}{|\nabla f(x^{k-1})|^2} \cdot p_{k-1} \tag{7}$$

p_k and p_{k-1} are then said conjugate with respect to some quadratic form Q since:

$$p_k \cdot Q \cdot p_{k-1} = 0$$

There cannot be more than n vectors in such system and therefore the optimal solution will be reached at least at interation $n + 1$.

2. 5. **Davidon-Fletcher-Powell method** (ref. 3, 5 and 6)

It is said to be the best method, until now, for unconstrained optimization. The direction p_k is chosen such that:

$$p_k = H_k \nabla f(x^k) \tag{8}$$

where

$$H_k = H_{k-1} - \frac{(H_{k-1}\, y)\,(H_{k-1}\, y)^T}{y^T H_{k-1}\, y} + \frac{s\, s^T}{s^T y} \tag{9}$$

and

$$y = \nabla f(x^k) - \nabla f(x^{k-1})$$
$$s = x^k - x^{k-1}$$

The idea is to approximate the inverse of the Hessian by formula (9) since H_{k-1} is not likely to be equal to it, at least for non-quadratic function. We can check that, similarly to the Hessian, we have the following relationship;

$$H_k\, y = s$$

We can notice that the two other matrices that are added to and substracted from H_{k-1} are of rank 1 and therefore H_k will keep the maximum of informations from H_{k-1}. Furtheremore if H_{k-1} is positive definite so is H_k. Usually H_0 is chosen as I since for the quadratic case the p_k's are conjugate and as we have seen above, the procedure converges in a finite number of steps.

3. COMPUTER PROGRAM

Procedure * II uses the method of Fletcher and Reeves namely, computes the new direction with (7). Procedure III uses a modification where the new direction is given by:

$$p_k = \nabla f(x^k) + p_{k-1} \frac{|\nabla f(x^k)|^2 - \nabla f(x^k)^T \cdot \nabla f(x^{k-1})}{p_{k-1}^T \cdot \nabla f(x^{k-1})} \tag{10}$$

3. 1. **Program description**

The program is composed of a main program and three subroutines: STEP, FUN, and GRAD. They have been written in FORTRAN IV language and tested on IBM 7044 and on IBM 360, model 40.

The main program processes the five functions (or more, if necessary) by two procedures and for different values of epsilon, upper limit of the slope of the function, at maximun, in the one - dimensional optimization routine (see annex 1 for the listing of the program).

The subroutine STEP, written by Miss Melanie Lenard, performes one dimensional optimization, using a convergence criterion. (see Annex 1).

The subroutine FUN computes the values of the different functions. According to the value of NP, function number, it goes to the corresponding statements which processes the NPth function. Similarly the subroutine GRAD computes the values of the gradient of the functions.

3.2. **Test characteristics**

3.2.1. *Convergence criterion*

Since the exact maximum or minimum of the function must satisfy $\nabla f(x) = 0$ we will use as test the relation:

$$|\nabla f(x)| < 10^{-4}$$

However this test is only valid when the maximum or minimum has an horizontal tangent. It seems that the function of problem 5 has an optimum with a vertical tangent, therefore this convergence test does not work, even if the processus is convergent. It will be necessary, in such cases, to add another test on the behaviour of the function.

* Here the name "Procedure II" or "Procedure III" is related to professor Wolfe's assignment as well as to the method of computation of the new direction.

3.2.2. *Epsilon for subroutine STEP*

The value of Epsilon, for the convergence criterion in the one dimensional optimization routine, should be set to some fraction of the starting slope, that is:

$$\varepsilon = \varepsilon_0 | p^T \cdot \nabla f(x_{k-1}) |$$

where

p : direction of search
∇f: gradient of f

ε_0: 0.5, 0.1, 0.01, 0.001, in addition of these epsilons variable we have used a fixed epsilon equal to 0.0001. The results were as good as with the others, in particular the value of the function obtained after 2000 iterations for problem 5, was the smallest.

3.2.3. *First step in one-dimensional optimisation routine*

For the initial iteration the value of PSTEP, length of the first step to be taken, was taken = 1.0; and for the following iterations PSTEP = TSTEP, distance from x_{k-1} to x_k in the direction p.

3.2.4. *Lower bounds*

A lower bound of 0.0001 was taken for Epsilon and PSTEP

3.2.5. *Error code*

From the subroutine STEP:

IERR = 0 no error detected
= 1 too many variables
= 2 function appears to be unbounded
= 3 search for maximum fails to converge

From the two other subroutines

INDEX = 1 non defined value for f in routine FUN
= 2 non defined value for gradient in GRAD
IERR = 4 if INDEX $\neq$ 0
FALL 2000 : stop after 2000 iterations

3.2.6. *Restart*

If IERR $\neq$ 0 the program automatically restarts the procedure with a different initial x^0 and initial PSTEP:

$$x_i^0 = x_i^0 + i$$
$$\text{PSTEP} = \text{PSTEP} + 1$$

This procedure has been successful as we can see for problem 3 (procedure II and fixed epsilon) and again problem 3 (procedure III and some epsilon).

3.3. **Output description**

Intermediate results are printed for each iteration: the step number, the accuracy, the value of f at x_k and x_{k+1}, epsilon, TSTEP and the components of x.

The final results are printed on a new page with either the message: "THE VARIABLES AT OPTIMUM" if we have an optimum value or with an error message; then the same values than above for the last step, and the number of iteration, number of function evaluation and of gradient evaluation are printed.

3.4. Summary of the results

Problem 1

$f(x_1, x_2) = 4x_1^2 + 2\,x_1\,x_2 + x_2^2 + 6\,x_1 + x_2$
initial $x^\circ = (0,0)$

proc.	eps.	acc.	n.it.	n.f.ev.	n.g.ev.	v. funct.	x_1	x_2
II	0.5	0.00000	2	10	3	−2.33	−0.83	0.33
	0.1	„	„	12	„	„	„	„
	0.01	„	„	„	„	„	„	„
	0.001	„	„	„	„	„	„	„
	0.0001	„	„	„	„	„	„	„
III	0.5	0.00015	9	69	10	−2.33	−0.83	0.33
	0.1	0.00033	„	63	„	„	„	„
	0.01	0.00028	„	78	„	„	„	„
	0.001	0.00012	„	85	„	„	„	„
	0.0001	0.00029	„	82	„	„	„	„

procedure II is faster than procedure III

Problem 2

$f(x_1, x_2) = 1/4\,x_1^4 + 4\,x_1^2 + 2\,x_1\,x_2 + x_2^2 + 6\,x_1 + x_2$
initial $x^\circ = (0,0)$

proc.	eps.	acc.	n.it.	n.f.ev.	n.g.ev.	v. funct.	x_1	x_2
II	0.5	0.00070	14	64	15	−2.23	−0.76	0.26
	0.1	0.00059	11	51	12	„	„	„
	0.01	0.00094	7	43	8	„	„	„
	0.001	0.00008	6	40	7	„	„	„
	0.0001	0.00099	15	81	16	„	„	„
III	0.5	0.00077	9	55	10	„	„	„
	0.1	„	8	65	9	„	„	„
	0.01	0.00058	7	54	8	„	„	„
	0.001	0.00066	„	66	„	„	„	„
	0.0001	0.00056	„	„	„	„	„	„

The two procedures give comparable results and performances.

Problem 3

$f(x_1, x_2) = 100\ (x_2 - x)^2 + (1 - x_1^2)^2$

initial $x^\circ = (-1.2, 1)$

proc.	eps	acc.	n.it.	n.f.ev.	n.g.ev.	v. funct.	x_1	x_2
II	0.5	0.00074	89	646	90	0.0	1.0	1.0
	0.1	0.00095	60	531	61	„	„	„
	0.01	0.00022	103	862	104	„	„	„
	0.001*	0.00087	54	626	55	„	„	„
	0.0001*	0.00084	„	635	„	„	„	„
III	0.5*	0.00091	207	1320	208	0.0	1.0	1.0
	0.1*	0.00100	984	5172	985	„	„	„
	0.01	0.00006	184	1596	185	„	„	„
	0.001	0.00098	519	3665	520	„	„	„
	0.0001	0.00099	611	4474	612	„	„	„

Procedure II is faster than procedure III. As mentioned in (4.6), the results written with* are coming from a restart.

Problem 4

$$f(x_1, x_2, x_3, x_4) = (x_1 + 10x_2)^2 + 5\ (x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10\ (x_1 - x_4)^4$$

initial $x^\circ = (3, -1, 0, 1)$

proc.	eps	acc.	n.it.	n.f.ev.	n.g.ev.	v. funct.	x_1	x_2	x_3	x_4
II	0.5	0.00094	175	738	176	0.0	0.024	−0.002	0.012	0.012
	0.1	0.00054	108	522	109	0.0	0.033	−0.003	0.016	0.017
	0.01	0.00063	107	534	108	„	0.036	−0.004	0.017	„
	0.001	0.00054	92	497	93	„	0.030	−0.003	0.015	0.015
	0.0001	0.00085	144	725	145	„	0.025	−0.003	0.013	0.013
III	0.5	49	2000	10748	2001	13.09	1.96	−0.41	0.35	1.48
	0.1	56	„	9593	„	8.63	0.95	−0.37	0.19	0.51
	0.01	38	„	9394	„	4.64	1.04	−0.29	0.23	0.59
	0.001	28	„	11987	„	6.70	1,66	−0.28	0.38	1.28
	0.0001	30	„	10389	„	8.16	1.80	−0.30	0.39	1.40

Procedure II reaches the optimum with some difference at the second digit level for the *xi*'s. Procedure III, while giving decreasing values of f, falls after 2000 iteration, without reaching an optimum.

Problem 5

$f(x_1, x_2) = (r - 1)^3 (2 + \cos (\theta - 1/(r - 1))$
where $r^2 = x_1^2 + x_2^2$ and $\theta = \arctan (x_2/x_1)$
initial $x^\circ = (-0.0001, 0.0002)$

proc.	eps.	acc.	n. it.	n.f.ev.	n.g.ev.	v. funct.	x_1	x_2
II	0.5	3761	2000	8001	2001	−2.53841	−0.00010	0.00020
	0.1	„	„	„	„	„	„	„
	0.01	„	„	„	„	„	„	„
	0.001	„	„	„	„	„	„	„
	0.0001	„	„	„	„	„	„	„
III	0.5	33556	„	8002	„	−2.54009	−0.00002	0.00002
	0.1	„	„	„	„	„	„	„
	0.01	„	„	„	„	„	„	„
	0.001	„	„	„	„	„	„	„
	0.0001	99999	„	80039	„	−2.54026	−0.00000	0.00000

As mentioned above the procedure cannot converge with the gradient criterion, although it can be seen on the computer outprint the value of the function reaches somme „stationary levels".

proc.	eps.	acc.	n.it.	n.f.ev.	n.g.ev.	v.funct.	x_1	x_2	IERR
II	0.5	3761	2000	8001	2001	−2.53841	−0.000096	0.0002010	
	0.5*	55233	2000	8123	2001	−2.54000	−0.000012	0.0000088	
	0.1	3761	2000	8001	2001	−2.53840	−0.000096	0.0002010	
	0.1*	53532	2000	8132	2001	−2.54020	−0.000012	0.0000090	
	0.01	3761	2000	8001	2001	−2.53840	−0.000096	0.0002016	
	0.01*	72899	2000	8107	2001	−2.54020	−0.000002	0.0000113	
	0.001	71713	2000	8121	2001	−2.54020	−0.000002	0.0000115	
	0.001*	3761	2000	8001	2001	−2.53840	−0.000096	0.0002016	
III	0.5	33557	2000	8002	2001	−2.5401	−0.000017	0.000017	
	0.5*	5266	6	127	6	−2.8076	−0.009877	0.006289	3
	0.1	33556	2000	8002	2001	−2.5401	−0.000017	0.000017	
	0.1*	9.65	387	1968	387	−2.9855	−0.000683	0.001008	3
	0.01	33556	2000	8002	2001	−2.5401	−0.000017	0.000017	
	0.01*	8818	3	108	3	−2.9022	0.003127	−0.004257	3
	0.001	33556	2000	8002	2001	−2.5401	−0.000017	0.000017	
	0.001*	8.821	3	112	3	−2.9040	0.003089	−0.004247	3

One —dimensional cptimization routine

by

MELANIE LENARD

This subroutine will be invoked by the FORTRAN statement: CALL STEP (NOVAR, XZERO, FZERO, PSTEP, P, EPSLON, XSTAR, FSTAR, TSTEP, IERR), where the following values must be supplied by the user:

NØVAR the number of independent variables
XZERØ the values of the independent variables
FZERØ the value of the function at XZERØ
PSTEP the length of the first step to be taken
P the direction in which the search is to be conducted
EPSLON uper limit for slope of function at maximum

The following values will be returned:

XSTAR the values of the independent variables at the maximum
FSTAR the value of the function at XSTAR
TSTEP the distance from XZERØ to XSTAR in the direction P
IERR error code
= 0 no error detected
= 1 too many variables (not more than 20)
= 2 function appears to be unbounded
= 3 search for maximum fails to converge

All arguments are "real" except for NØVAR and IERR.

XZERØ, P, and XSTAR are arrays which must have dimension of at least NØVAR.

The user of subroutine "STEP" must supply another subroutine which will compute the value of the function (to be *maximized*) for given values of the independent variables. It must begin with the statement

SUBRØUTINE FUN (NØVAR, X, F)

where the following values will be supplied:

NØVAR the number of independent variables
X values of the independent variables

and a value is to be returned for:

F the value of the function at X.

X is an array which must have a dimension of at least NOVAR. The dimension of X in this routine must be less than or equal to (with the WATFØR compiler, equal to) the dimension of the independent variable array in the main routine.

Method. The one-dimensional optimization performes as follows:

First, we seek to "box-in" the maximum with three equally spaced points. Using the specified starting values of X_1, (XZERØ) and F_1, (FZERØ), and step length g, (PSTEP), we set $X_2 = X_1 + G$, where $G = (g_1, ..., g_n)$, $g_i = g$, we find $F_2 = F(X_2)$. If $F_2 < F_1$, we give to G the value $-$ G, interchange the values of F_1 and F_2, and reverse the direction of the search.

Then, we double the step length, $g = 2\,g$, put $X_3 = X_2 + G$ and evaluate $F_3 = F(X_3)$. If $F_2 \leqslant F_3$, we update $F_1 = F_2$ and $F_2 = F_3$. Then, we again double step length, $g = 2g$, put $X_3 = X_2 + G$, and compute $F_3 = F(X_3)$, continuing in this way until $F_3 < F_2$. (If more than 25 such stept are required, the error code is set to 2 and control is returned to the calling program). At this stage, we put $X_4 = X_3 - \frac{1}{2} G$, and evaluate $F_4 = F(X_4)$. If $F_2 \leqslant F_4$, we set $F_1 = F_2$ and $F_2 = F_4$; otherwise, we put $F_3 = F_4$.

We now have three equally spaced points X_1, X_2, X_3 with function values F_1, F_2, F_3, "boxing-in" the maximum.

We fit a quadratic to these three points and find the location of the turning point of the quadratic. The distance, d, from the middle of the three points, X_2, to the turning point is limited to a maximum value (STEPMX). Then, if the turning point is a maximum, we put $d = -d$ and we will note $D = (d_1, ... d_n)$, $d_i = d$.

We now have four points, X_1, X_2, X_3, and $X_4 = X_2 + D$. We arrange the X_j so taht $X_1 \leqslant X_2 \leqslant X_3 \leqslant X_4$, and perform the same rearrangement on the F_j. We find the maximum of the F_j, say F_i. If i is 1 or 2, we discard the point X_4; if i is 3 or 4, we discard X_1. We treat the remaining three points as before, fitting a quadratic and finding the turning point.

Iterations proceed until values of successive maxima differ by less than 0.1% (ESPLØN) or the number of iterations reaches 50*). In the latter case, the error code is set to 3 before control is returned to the calling program.

The search for a maximum will be considered converged when the slope at the maximum is less than EPSLØN, that is,

$$\left|\frac{F_4 - F_2}{X_4 - X_2}\right| < \varepsilon_2$$

where

F_4 current maximum of F
F_2 value of F at intermediate point of 3 points used for interpolation
X_4 step length to point at which $F(x_4) = F_4$
X_2 step lenght to point at which $F(x_2) = F_2$
ε_2 value of argument EPSLØN

To a void dividing by a very small number, the search is also convergent if $|X_4 - X_2| \leqslant 10^{-6}$

The value of EPSLØN used in Version 2 should be set by the user in the calling program to some fraction of the starting slope that is,

$$\varepsilon_2 = \varepsilon |P^T \cdot \nabla F(X^0)|$$

where

P direction of search
$\nabla F(X^0)$ gradient of F at starting point

Starting Step Size

The starting step size supplied through the argument PSTEP is divided by the norm of the direction vector P before being used. This guarantees that the starting step will not be distorted by the size of P

Nevertheless when control is returned to the calling program

$$|X^* - X^0| = t|P|$$

where

X^* is XSTAR
X^0 is XZERØ
t is TSTEP
P is P

*) The level of such parameters can be easely modified.

```
C          UNCONSTRAINED  NONLINEAR  OPTIMISATION  PROGRAMME
C     USING  FLETCHER  AND  REEVES  METHOD (PROCEDURE II)  AND
C     A MODIFIED  FLETCHER  AND  REEVES  FORMULA,(PROCEDURE III).
C
C          THE  PROGRAMME  HAS  BEEN  WRITTEN  IN  FORTRAN IV,
C     DECEMBER  1969, BY
C
C               JUAN  AMUNATEGUI , AND  VINCENTIU  DUMITRU
C
C     AS  A  FINAL  ASSIGNMENT  GIVEN  BY
C
C                    PROFESSOR  PHILIP WOLFE
C     AT  COLUMBIA  UNIVERSITY  IN THE  CITY  OF  NEW  YORK.
C
C          NOVAR  = NUMBER OF INDEPENDENT VARIABLES
C          NEPS   = NUMBER OF EPSILONS FOR ONE-DIMENSION OPTIMIZATION
C          ACC    = ACCURACY FOR GRADIENT
C          NP     = PROBLEM NUMBER
C          MAX    = TYPE OF OPTIMIZATION INDICATOR
C                     IF +1  - MAXIMIZATION
C                     IF -1  - MINIMIZATION
C          NPROC  = TYPE OF THE PROCEDURE
C                     IF 2 ,PROCEDURE TWO WILL BE APPLIED
C                     IF 3 ,PROCEDURE THREE WILL BE APPLIED
C                     IF 0 ,BOTH PROCEDURE WILL BE APPLIED
C          EMIN   = LOWER BOUND FOR EPSILON
C          BEPS   = UPPER BOUND FOR EPSILON
C          BTNG   = UPPER BOUND FOR ACCURACY
C          ESTEP  = LOWER BOUND FOR STEP IN THE OPTIMALITY TEST
C          PSTEP  = LENGTH OF FIRST STEP TO BE TAKEN
C          X      = ARRAY OF INDEPENDENT VARIABLES
C          Y      = ARRAY OF INDEPENDENT VARIABLES AT THE MAXIMUM
C          PV     = DIRECTION P
C          RX     = GRADIENT AT X
C          E      = ARRAY OF EPSILON FOR ONE-DIMENSION OPTIZATION
C          FX     = VALUE OF THE FUNCTION FOR DIFFERENT X
C          FY     = VALUE OF THE FUNCTION AT MAXIMUM Y
C          RY     = THE GRADIENT AT Y
C          PZER   = INITIAL DIRECTION ARRAY
C
C          SUBROUTINE STEP = ONE-DIMENSIONAL OPTIMIZATION SUBROUTINE
C          SUBROUTINE FUN  = COMPUTE THE VALUE OF THE FUNCTION
C          SUBROUTINE GRAD = COMPUTE THE VALUE OF THE GRADIENT
      DIMENSION X(50),Y(50),PZER(50),PV(50),RX(50),RY(50)
      DIMENSION XZER(50),RKRKM(50),PVMRKM(50),SXZER(50),E(10),TITLE(20)
      DATA F/4HF    /
      DATA T/4HT    /
      COMMON NOFUN,NOGRAD,NP,MAX,NPROC,INDEX
      INTEGER FALL
C
    1 CONTINUE
C
      NP=0
C
C     READ CONTROL CARDS
    5 CONTINUE
      READ(1,6) (TITLE(I),I=1,20)
      IF(TITLE(20).EQ.F) GO TO 500
      IF(TITLE(20).NE.T) GO TO 5
      READ(1,10) NP,NPROC,MAX,NEPS,EMIN,BEPS,ACC,BTNG,PSTEP,ESTEP,NOVAR
      IF(EMIN.GT.0.0) GO TO 11
      EMIN=0.0001
   11 CONTINUE
      IF(BEPS.GT.0.3) GO TO 12
      BEPS=100.
   12 CONTINUE
      IF(BTNG.GT.0.0) GO TO 13
      BTNG=100000.0
```

```
   13 CONTINUE
      IF(ESTEP.GT.0.0) GO TO 14
      ESTEP=0.001
   14 CONTINUE
      IF(NP.LT.1) GO TO 400
      READ(1,20) (XZER(I),I=1,NOVAR)
      READ(1,20) (E(I),I=1,NEPS)
      SPSTEP=PSTEP
      DO 21 I=1,NOVAR
      SXZER(I)=XZER(I)
   21 CONTINUE
C
      LTEST=NPROC
      LPROC=0
      LPROC=LPROC+1
      IF(LTEST.EQ.0) NPROC=2
C
      GO TO 26
C
   25 CONTINUE
C
      LPROC=LPROC+1
      NPROC=NPROC+1
   26 CONTINUE
C
C     LOOP FOR DIFFERENT VALUE OF EPSILON
C
      DO 300 IND=1,NEPS
      WRITE(3,27) NP,NPROC
      WRITE(3,29) (TITLE(I),I=1,19)
C
C     INITIALIZATION
      K=0
      GX=0.0
      GY=0.0
      ETX=0.0
      ETY=0.0
      LERR=0
      NTURN=0
      PSTEP=SPSTEP
      NOFUN=0
      NOGRAD=0
      FX=0.0
      FY=0.0
      DO 30 I=1,NOVAR
      X(I)=XZER(I)
      Y(I)=0.0
      RX(I)=0.0
      RY(I)=0.0
   30 CONTINUE
   31 CONTINUE
C
C     COMPUTE F OF INITIAL X
C
      CALL FUN(NOVAR,X,FX)
      IF(INDEX.GE.1) GO TO 261
C
      CALL GRAD(NOVAR,X,RX)
      IF(INDEX.GE.1) GO TO 261
C     INITIAL  C
      DO 35 I=1,NOVAR
      PZER(I)=RX(I)
      PY(I)=RX(I)
   35 CONTINUE
      WRITE(3,40) E(IND)
      WRITE(3,45)
C
   50 K=K+1
```

```
      IF(IND.EQ.NEPS) GO TO 65
C
C     COMPUTE EPSILON FOR VERSION 2
C
      EPSLON=0.0
      DO 60 I=1,NOVAR
   60 EPSLON=EPSLON+PVER(I)*PV(I)
      EPSLON=ABS(EPSLON)*E(IND)
      IF(EPSLON.LT.EMIN) EPSLON=EMIN
      IF(EPSLON.GT.BEPS) EPSLON=BEPS
      GO TO 66
   65 EPSLON=ABS(E(IND))
   66 CONTINUE
C
C     ONE DIMENSION OPTIMIZATION
C
      CALL STEP(NOVAR,X,FX,PSTEP,PV,EPSLON,Y,FY,ISTEP,IERR)
      IF(INDEX.GE.1) GO TO 261
      IF(IERR.NE.0) GO TO 250
C     COMPUTE GRADIENT
C
      CALL GRAD(NOVAR,Y,RY)
C
      IF(INDEX.GE.1) GO TO 261
C
      ETX=0.0
      ETY=0.0
C
C     COMPUTE ACCURACY
C
      DO 70 I=1,NOVAR
      RKRKM(I)=RY(I)*RX(I)
      PKMRKM(I)=PV(I)*RX(I)
      ETX=ETX+RX(I)**2
   70 ETY=ETY+RY(I)**2
      ETX=SQRT(ETX)
      ETY=SQRT(ETY)
C
C     OUTPUT INTERMEDIATE RESULT
C
      GX=FX
      GY=FY
      IF(MAX.EQ.1) GO TO 71
      GX=-FX
      GY=-FY
   71 CONTINUE
      IF(K.GT.135) GO TO 85
      IF(K.NE.46) GO TO 75
      IF(K.NE.91) GO TO 75
      WRITE(3,27) NP,NPROC
      WRITE(3,29) (TITLE(I),I=1,19)
   75 CONTINUE
      WRITE(3,80)  K,ETY,GX,GY,EPSLON,ISTEP,(Y(I),I=1,NOVAR).
   85 CONTINUE
C
C     STOP IF OPTIMUM
      IF(ETY.LT.ACC) GO TO 200
C
      IF((ETY.GT.BIG).AND.(ISTEP.LT.ESTEP)) GO TO 150
C     NEW DIRECTION
      IF(NPROC.EQ.3) GO TO 91
C
      DO 90 I=1,NOVAR
   90 PV(I)=RY(I)+PV(I)*(ETY/ETX)**2
      GO TO 95
   91 CONTINUE
      DO 93 I=1,NOVAR
      IF(PKMRKM(I).EQ.0.0) GO TO 261
```

```
   93 PV(I)=RY(I)+PV(I)*(ETY-RKRKM(I))/PKMRKM(I)
   95 CONTINUE
C
C     CHANGE FOR NEXT STEP
C
      DO 100 I=1,NOVAR
      PZER(I)=RY(I)
      X(I)=Y(I)
      RX(I)=RY(I)
  100 CONTINUE
      FX=FY
      PSTEP=TSTEP
      IF(PSTEP.LT.EMIN) PSTEP=EMIN
      IF(K.GT.1999) GO TO 270
      GO TO 50
  150 CONTINUE
      NTURN=1
  200 CONTINUE
      IF(MAX.EQ.1) GO TO 201
      GY=-FY
  201 CONTINUE
      WRITE(3,28) NP,NPROC
      WRITE(3,40) E(IND)
      WRITE(3,205)
      WRITE(3,29) (TITLE(I),I=1,19)
      WRITE(3,45)
      WRITE(3,80) K,ETY,GX,GY,EPSLON,TSTEP,(Y(I),I=1,NOVAR)
      WRITE(3,210)K,NOFUN,NOGRAD
      IF(NTURN.EQ.1) GO TO 263
      GO TO 300
C
  250 CONTINUE
      WRITE(3,28) NP,NPROC
      WRITE(3,29) (TITLE(I),I=1,19)
      WRITE(3,40) E(IND)
      WRITE(3,260) IERR
      WRITE(3,45)
      WRITE(3,80) K,ETY,GX,GY,EPSLON,TSTEP,(Y(I),I=1,NOVAR)
      WRITE(3,210) K,NOFUN,NOGRAD
      GO TO 263
  261 IERR=4
      WRITE(3,28) NP,NPROC
      WRITE(3,29) (TITLE(I),I=1,19)
      WRITE(3,40) E(IND)
      WRITE(3,80) K,ETY,GX,GY,EPSLON,TSTEP,(Y(I),I=1,NOVAR)
  262 WRITE(3,260) IERR
  263 CONTINUE
      IERR=IERR+1
      IF(IERR.GT.1) GO TO 300
      K=0
      GX=0.0
      GY=0.0
      ETX=0.0
      ETY=0.0
      PSTEP=SPSTEP+2.
      NOFUN=0
      NOGRAD=0
      FX=0.0
      FY=0.0
      DO 264 I=1,NOVAR
      PERXO=I
      X(I)=XZER(I)+PERXO
      IF(NTURN.EQ.1) X(I)=-SXZER(I)
      Y(I)=0.0
      RX(I)=0.0
      RY(I)=0.0
  264 CONTINUE
      NTURN=0
```

```
      WRITE(3,27) NP,NPROC
      WRITE(3,29) (TITLE(I),I=1,19)
      WRITE(3,265)
      GO TO 31
  270 CONTINUE
      FALL=2000
      WRITE(3,28) NP,NPROC
      WRITE(3,40) E(IND)
      WRITE(3,271)FALL
      WRITE(3,45)
      WRITE(3,80) K,ETY,GX,GY,EPSLON,TSTEP,(Y(I),I=1,NOVAR)
      WRITE(3,210) K,NOFUN,NOGRAD
      GO TO 263
  300 CONTINUE
      IF(LTEST.NE.0) GO TO 1
      IF(LPROC.GE.2) GO TO 1
      IF(NPROC.EQ.3) GO TO 1
C
      GO TO 25
C
  400 WRITE(3,401)
      GO TO 1
  500 CONTINUE
      WRITE(3,29) (TITLE(I),I=1,19)
    6 FORMAT(20A4)
   10 FORMAT(4I2,2X,4E10.0,F10.2,E10.0,3X,I2)
   20 FORMAT(8F10.4)
   27 FORMAT(1HI,25X,11HPROBLEM NO ,I2,10X,13HPROCEDURE NO ,I2)
   28 FORMAT(1HO,25X,11HPROBLEM NO ,I2,10X,13HPROCEDURE NO ,I2)
   29 FORMAT(1HO,19A4)
   40 FORMAT(1HO,40HOPTIMIZATION OF FUNCTION F FOR EPSILON =,F10.4)
   45 FORMAT(1HO,3X,1HK,9X,3HACC, 7X,5HF(XK), 5X,7HF(XK+1), 8X,4HEPS.,
     17X,5HTSTEP,10X,2HX1,10X,7HX2 ETC.)
   80 FORMAT(1H ,I4,9F12.5)
   81 FORMAT(1H1,I4,9F12.5)
  205 FORMAT(1HO,24HTHE VARIABLES AT OPTIMUM/)
  210 FORMAT(1HO,3X,2HK=,I5,10X,6HNOFUN=,I5,10X,7HNOGRAD=,I5)
  260 FORMAT(1HO,5HIERR=,I2)
  265 FORMAT(1HO,10HNEW START.)
  271 FORMAT(1HO,4HFALL,2X, I4/)
  401 FORMAT(1HO,25HNUMBER OF PROBLEM MISSING)
      STOP
      END
```

```
      SUBROUTINE GRAD(NOVAR,X,V)
      DIMENSION X(NOVAR),V(NOVAR)
      COMMON NOFDN,NOGRAD,NP,MAX,NPROC,INDEX
      INDEX=0
      NOGRAD=NOGRAD+1
      IF((NP.LT.1).OR.(NP.GT.5)) GO TO 90
      GO TO (10,20,30,40,50),NP
   10 CONTINUE
      V(1)=8.*X(1)+2.*X(2)+6.
      V(2)=2.*X(1)+2.*X(2)+1.
      GO TO 99
   20 CONTINUE
      V(1)=8.*X(1)+2.*X(2)+6.+X(1)**3
      V(2)=2.*X(1)+2.*X(2)+1.
      GO TO 99
   30 CONTINUE
      V(1)=-2.*(200.*(X(2)-X(1)**2)*X(1)-X(1)+1.)
      V(2)=200.*(X(2)-X(1)**2)
      GO TO 99
   40 CONTINUE
      V(1)=2.*(X(1)+10.*X(2))+40.*(X(1)-X(4))**3
      V(2)=20.*(X(1)+10.*X(2))+4.*(X(2)-2.*X(3))**3
      V(3)=-8.*(X(2)-2.*X(3))**3+10.*(X(3)-X(4))
      V(4)=-10.*(X(3)-X(4))-40.*(X(1)-X(4))**3
      GO TO 99
   50 CONTINUE
      IF((ABS(X(1)).LT.1.E-3).AND.( ABS(X(2)).LT.1.E-3)).GO TO 51
      TETA=ATAN2(X(2),X(1))
      GO TO 52
   51 TETA=0.0
   52 CONTINUE
      R=SQRT(X(1)**2+X(2)**2)
      IF(R.EQ.1.0) GO TO 92
      F1=-SIN(TETA-(1./(R-1.)))*(R-1.)**3
      F2=3.*(R-1.)**2*(2.+COS(TETA-(1./(R-1.))))-SIN(TETA-(1./(R-1.)))*(
     1 R-1.)
      IF(F2.EQ.0.0) GO TO 92
      V(1)=-(X(2)/R**2)*F1+X(1)*(1./R)*F2
      V(2)=(X(1)/R**2)*F1+X(2)*(1./R)*F2
      GO TO 99
   90 CONTINUE
      WRITE(3,91)
   91 FORMAT(1H0,25HNUMBER OF PROBLEM MISSING)
      GO TO 99
   92 INDEX=2
      WRITE(3,93) INDEX
   93 FORMAT(1H0,3X,6HINDEX=,I2)
   99 CONTINUE
      IF(MAX.EQ.1) GO TO 100
      DO 95 I=1,NOVAR
   95 V(I)=-V(I)
  100 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE STEP(NOVAR,XZERO,FZERO,PSTEP,P,EPSLON,XSTAR,FSTAR,TSTEP
     1,IERR)
C
C         THIS IS VERSION 2
C
C
C     THE PURPOSE OF THIS ROUTINE IS TO DETERMINE A STEP ALONG A LINE IN
C     A GIVEN DIRECTION FROM A GIVEN POINT WHICH MAXIMIZES THE VALUE OF A
C     FUNCTION.
C
C     ARGUMENTS SUPPLIED BY CALLING PROGRAM
C          NOVAR  NUMBER OF INDEPENDENT VARIABLES
C          XZERO  STARTING VALUE OF INDEPENDENT VARIABLES
C          FZERO  VALUE OF FUNCTION AT XZERO
C          PSTEP  GUESS FOR STARTING STEP LENGTH
C          P      DIRECTION TO MOVE FROM XZERO
C          EPSLON CONVERGENCE TOLERANCE
C                 VERSION 2 UPPER LIMIT FOR SLOPE OF F
C     ARGUMENTS RETURNED TO CALLING PROGRAM
C          XSTAR  NEW VALUES OF INDEPENDENT VARIABLES
C          FSTAR  VALUE OF FUNCTION AT XSTAR
C          TSTEP  LENGTH OF STEP TAKEN TO ACHIVE MAXIMUM
C          IERR   ERROR CODE
C                 = 0 NO ERROR DETECTED
C                 = 1 TOO MANY VARIABLES
C                 = 2 FUNCTION IS UNBOUNDED
C                 = 3 SEARCH FOR MINIMUM DID NOT CONVERGE
C
      DIMENSION XZERO(NOVAR),P(NOVAR),XSTAR(NOVAR),X(20,4),F(4),D(4)
C     D MEASURES DISTANCE TO EACH POINT FROM XZERO
C     XSTAR WILL BE USED AS A WORK AREA FOR CALLS TO 'FUN'
C
      IERR= 0
      IF(NOVAR.GT.20) GO TO 990
      GSTEP=PSTEP
      STEPMX=1.0E+05
      PNORM=0.0
      DO 10 I = 1,NOVAR
   10 PNORM = PNORM + P(I)*P(I)
      PNORM = SQRT(PNORM)
      STEPMX= STEPMX/PNORM
      GSTEP = GSTEP/PNORM
C     'BOX-IV' MAXIMUM WITH 3 EQUALLY SPACED POINTS
      F(1) = FZERO
      D(1) = 0.0
      DO 20 I = 1,NOVAR
      X(I,1) = XZERO(I)
      X(I,2) = XZERO(I) + GSTEP*P(I)
      XSTAR(I) = X(I,2)
   20 CONTINUE
      D(2) = GSTEP
      CALL FUN(NOVAR,XSTAR,F(2))
      IF(F(2).GT.F(1))GO TO 24
      GSTEP=-GSTEP
C     INTERCHANGE 1 AND 2
      STORE = F(2)
      F(2) = F(1)
      F(1) = STORE
      STORE=D(2)
      D(2)=D(1)
      D(1)=STORE
      DO 23 I = 1,NOVAR
      STORE = X(I,2)
      X(I,2)= X(I,1)
   23 X(I,1)= STORE
   24 CONTINUE
      NITER=0
   25 CONTINUE
      GSTEP=2.0*GSTEP
```

```
      NITER=NITER +1
      IF(NITER.GT.25) GO TO 991
      DO 30 I=1,NOVAR
      X(I,3)=X(I,2)+GSTEP*P(I)
      XSTAR(I)=X(I,3)
   30 CONTINUE
      CALL FUN(NOVAR,XSTAR,F(3))
      D(3)=D(2)+GSTEP
      IF(F(3).LT.F(2)) GO TO 40
      F(1) =F(2)
      F(2) =F(3)
      D(1) =D(2)
      D(2) =D(3)
      DO 35 I= 1,NOVAR
      X(I,1)=X(I,2)
   35 X(I,2)=X(I,3)
      GO TO 25
   40 CONTINUE
      GSTEP=GSTEP/2.0
      DO 45 I=1,NOVAR
      X(I,4)=X(I,3)-GSTEP*P(I)
      XSTAR(I)=X(I,4)
   45 CONTINUE
      D(4)=D(3)-GSTEP
      CALL FUN(NOVAR,XSTAR,F(4))
      IF(F(4).GT.F(2))GO TO 55
      F(3)=F(4)
      D(3)=D(4)
      DO 50 I=1,NOVAR
   50 X(I,3)=X(I,4)
      GO TO 74
   55 CONTINUE
      F(1)=F(2)
      D(1)=D(2)
      F(2)=F(4)
      D(2)=D(4)
      DO 60 I=1,NOVAR
      X(I,1)=X(I,2)
   60 X(I,2)=X(I,4)
   74 CONTINUE
      NITER=0
      MAXLOC=2
   75 CONTINUE
C     FIT A QUADRATIC AND FIND TURNING POINT
      NITER=NITER+1
      IF(NITER.GT.250) GO TO 992
      TERM2=(F(2)-F(1))/(D(2)-D(1))
      TERM3=(F(3)-F(1))/(D(3)-D(1))
      A=(TERM3-TERM2)/(D(3)-D(2))
      B=TERM2-A*(D(1)+D(2))
      D(4)=-B/(2.0*A)
      IF(ABS(D(4)).GT.ABS(STEPMX)) D(4)=SIGN(STEPMX,D(4))
      IF(A.LT.0.0) GO TO 90
C
C     TURNING POINT IS A MINIMUM
C
      D(4)=2.0*D(2)-D(4)
   90 CONTINUE
      DO 85 I=1,NOVAR
      X(I,4)=X(I,2)+D(4)*P(I)
      XSTAR(I)=X(I,4)
   85 CONTINUE
      CALL FUN(NOVAR,XSTAR,F(4))
C
C     VERSION 2
C
      IF(ABS(D(4)-D(2)).LT.1.0E-06) GO TO 900
      IF(ABS((F(4)-F(2))/(D(4)-D(2))).LE.EPSLON) GO TO 900
```

```
C     REARRANGE FOUR POINTS IN ORDER OF DISTANCE FROM XZERO .
C     IF GSTEP IS NEGATIVE,VALUE OF D ARE IN DECREASING ORDER
      DO 100 J=1,3
      IF(SIGN(D(4),GSTEP).LT.SIGN(D(J),GSTEP)) GO TO 105
  100 CONTINUE
      GO TO 115
  105 CONTINUE
      JJ=4-J
      STORE=D(4)
      DO 106 K=1,JJ
      KK=4-K
  106 D(KK+1)=D(KK)
      D(J)=STORE
      STORE=F(4)
      DO 107 K=1,JJ
      KK=4-K
  107 F(KK+1)=F(KK)
      F(J)=STORE
      DO 110 I=1,NOVAR
      STORE =X(I,4)
      DO 108 K=1,JJ
      KK=4-K
  108 X(I,KK+1)=X(I,KK)
      X(I,J)=STORE
  110 CONTINUE
  115 CONTINUE
C
CC    FIND MAXIMUM F
C
      MAXLOC=1
      DO 125 J=2,4
      IF(F(J).LT.F(MAXLOC)) GO TO 125
      MAXLOC=J
  125 CONTINUE
      IF((MAXLOC.EQ.3).OR.(MAXLOC.EQ.4))GO TO 150
C
CC    DISCARD POINT 4
C
      GO TO 75
  150 CONTINUE
C
CC    DISCARD POINT 1
C
      DO 155 J=2,4
      D(J-1)=D(J)
      F(J-1)=F(J)
      DO 155 I=1,NOVAR
      X(I,J-1)=X(I,J)
  155 CONTINUE
      MAXLOC=MAXLOC-1
      GO TO 75
  900 CONTINUE
C     PROCEDURE HAS CONVERGED
C
      FSTAR=F(4)
      TSTEP=D(4)
      GO TO 999
  990 IERR=1
      GO TO 999
  991 CONTINUE
      IERR=2
      GO TO 999
  992 CONTINUE
      IERR=3
  999 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE FUN(NOVAR,X,F)
      DIMENSION X(NOVAR)
      COMMON/ NOFUN,NOGRAD,NP,MAX,NPROC,INDEX
      INDEX=0
      NOFUN=NOFUN+1
      IF((NP.LT.1).OR.(NP.GT.5)) GO TO 90
      GO TO (10,20,30,40,50),NP
   10 CONTINUE
      F=4.*X(1)**2+2.*X(1)*X(2)+X(2)**2+6.*X(1)+X(2)
      GO TO 99
   20 CONTINUE
      F=4.*X(1)**2+2.*X(1)*X(2)+X(2)**2+6.*X(1)+X(2)+(X(1)**4)/4.
      GO TO 99
   30 CONTINUE
      F=100.*(X(2)-X(1)**2)**2+(1.-X(1))**2
      GO TO 99
   40 CONTINUE
      F=(X(1)+10.*X(2))**2+5.*(X(3)-X(4))**2+(X(2)-2.*X(3))**4+10.*(X(1)
     1-X(4))**4
      GO TO 99
   50 CONTINUE
      IF((ABS(X(1)).LT.1.E-3).AND.( ABS(X(2)).LT.1.E-3)) GO TO 51
      TETA=ATAN2(X(2),X(1))
      GO TO 52
   51 TETA=0.0
   52 CONTINUE
      R=SQRT(X(1)**2+X(2)**2)
      IF(R.EQ.1.0) GO TO 92
      F=(2.+COS(TETA-(1./(R-1.))))*(R-1.)**3
      GO TO 99
   90 CONTINUE
      WRITE(6,91)
   91 FORMAT(1H0,25HNUMBER OF PROBLEM MISSING)
      GO TO 99
   92 INDEX=1
      WRITE(3,93) INDEX
   93 FORMAT(1H0,3X,6HINDEX=,I2)
   99 CONTINUE
      IF(MAX.EQ.1) GO TO 100
      F=-F
  100 CONTINUE
      RETURN
      END
```

REFERENCES

(extracted from a bibliography given by Professor P. Wolfe)

1. CAUCHY, AUGUSTIN,: *Méthodes générales pour la résolution des systèmes d'équations simultanées.* C.R. de l'Académie des Sciences 25, Juillet-Décembre 1847.
2. CURRY, J. B.: *The method of steepest descent for nonlinear minimization problems.* Quarterly of Applied Mathematics. Vol. 2, no. 3, October 1944
3. DAVIDON, W. C.: *Variable Metric Method for Minimization,* Argonne National Laboratory, Lemont, Illinois, Report no ANL-5990, November 1959
4. HESTENES, M. R. and E. STIEFEL: *Methods of conjugate gradients for solving linear systems.* J. Rearch National Bureau of Standards, 49, 1952
5. FLETCHER, R. and M. J. D. POWELL: *A rapidly convergent descent method for minimization.* The Computer Journal, 6, 1963
6. WOLFE, P.: *Another variable metric method.* IBM Research Note, November 67 1967
7. FORSYTHES, A. I. and G. E. FORSYTHE: *IBM Experiments with Accelerated Gradient Methods for Linear Equations.* National Bureau of Standards, Washington D.C., NBS Report 1643, March 1952
8. STEIN, M. L.: *Gradient Methods for Linear Equations.* National Bureau of Standards, Washington D. C., NBS Report 52—7, July 1951
9. STIEFEL, EDUARD: *Relaxtionsmethoden bester strategie zur lösung gleichungssysteme.* Commentarii Mathematici Helveteci, Vol 29, 1955

L'EXPERIMENTATION D'UN PROGRAMME NON-LINEAIRE SANS RESTRICTIONS

Les problèmes de programmation non-linéaire sans restrictions ont attiré l'attention de nombreux mathématiciens, soit en tant que problème en soi, soit pour la résolution des problèmes non-linéaires à restrictions, à l'aide des méthodes séquentielles. Le but du présent article este celui d'exposer, dans la première partie, quelques méthodes connues pour la résolution des problèmes non-linéaires sans restrictions et, en même temps, de décrire, dans la seconde partie, un programme pour l'ordinateur en utilisant la formule Fletcher et Reeves et une modification de cette dernière, ainsi que quelques résultats obtenus avec différentes fonctions. Ce programme a été présenté par les auteurs afin de remplir l'une des conditions requises au cours de programmation non-linéaire tenu à l'Université Columbia de New York, par M. le Prof. P. Wolfe.

ЭКСПЕРИМЕНТИРОВАНИЕ НА КАЛЬКУЛЯТОРЕ НЕЛИНЕЙНОЙ ПРОГРАММЫ БЕЗ ОГРАНИЧЕНИЙ

Вопросы нелинейного программирования без ограничений привлекли недавно внимание многих математиков либо как вопрос сам по себе, либо в связи с решением нелинейных задач с ограничениями при помощи секвенционных методов. Целью настоящей статьи является привести в первой части краткое изложение нескольких существующих методов решения нелинейных задач без ограничений, а во второй части указать программу для калькулятора, в которой применяются формула Флетчера и Ривса и модификация этой формулы, а также несколько полученных результатов с различными функциями. Эта программа была приведена авторами для выполнения одного из требуемых условий на курсе нелинейного программирования, который читал профессор П. Вульф в Колумбийском университете в Нью-Йорке.