stichting
mathematisch
centrum

MC

AFDELING NUMERIEKE WISKUNDE
(DEPARTMENT NUMERICAL MATHEMATICS)

NN 9/77 JANUARI

J.C.P. BUS

A PROPOSAL FOR THE CLASSIFICATION AND DOCUMENTATION OF TEST PROBLEMS IN THE FIELD OF NONLINEAR PROGRAMMING

Preprint

2e boerhaavestraat 49 amsterdam

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O).

AMS(MOS) subject classification scheme (1970): 65K05, 90C30

A proposal for the classification and documentation of test problems in the field of nonlinear programming *)

by

J.C.P. Bus

ABSTRACT

We give a proposal for the classification and documentation of test-problems in the field of nonlinear programming. The ideas given here are meant to be the first step on our way to create a set of classified and well-documented testproblems. This paper is explicitly meant as a contri - bution to the discussions about testing methodologies for mathematical programming algorithms.

KEY WORDS & PHRASES: *Nonlinear programming, testing methodologies, classification and documentation of testproblems.*

---

*) This report will be submitted for publication elsewhere

1. INTRODUCTION

In the last years there has been considerable discussion about the evaluation of software in the field of Mathematical Programming. In an effort to canalize these discussions the Mathematical Programming Society established the Working Committee on Algorithms with the charge to concern itself, among other things, with testing methodologies for mathematical programming algorithms (see Math. Prog. 9.1). Since a set of standard test problems is one of the basic tools necessary for evaluating software in this field, the Dutch Working Group on Nonlinear Programming decided to contribute to these discussions by working on a proposal for classification and documentation of test problems, especially in the field of nonlinear programming. The result of this work is given in this paper. Meanwhile, a set of about 100 test problems is gathered from literature and practice. By now, some members of the group started to classify and describe these problems according to the guidelines given in this proposal in order to give us practice. Our ultimate goal is to obtain a representative set of test problems in the field of nonlinear programming. The classification of this set should be suitable for testing software in this field and the description should be given in a standardized format and in machine readable form. Moreover, the documentation should be such that it becomes easy to extend the set of test problems. This set may be used, for instance, in a clearing house (cf. LOOTSMA [1976]), for comparison, certification and validation of nonlinear programming software. It may also be used to create standard driver programs for testing on different computers and in various languages.

We expect that this proposal fits very well into the discussions and activities of the Working Committee on Algorithms.

2. CLASSIFICATION OF TEST PROBLEMS

A problem classification to be used for the classification of test problems should satisfy two general criteria. Firstly, the tester of a program should be able to choose (classes of) test problems to which the program is applicable. Moreover, he should have enough information about special properties of the test problems to be able to recognize a special

behaviour of the program. Secondly, the user of the programs tested should be able to classify his real-life problems so that he can choose a program which appeared to be "best" for the class to which his problem belongs. With this in mind we consider the following special properties of nonlinear programming problems (see also LOOTSMA [1976]).

a. The type of the objective function. We may distinguish linear and quadratic functions, functions which are sums of squares and other functions.
b. The type of the constraints. We may distinguish unconstrained and linearly constrained problems and problems with nonlinear constraints. Furthermore, the constraints may be bounds on the variables, equality constraints or inequality constraints.
c. The functions (objective function and constraint functions) may be regular (sufficiently differentiable) or irregular on the feasible region. The algorithm underlying a program to be tested should have a sound mathematical basis. Most frequently differentiability is assumed in such mathematical theory. For example, when the functions are regular we may use first- and second-order theory (see FIACCO & McCORMICK [1968]) to prove optimality of some point. For irregular functions optimality conditions may become very complicated.
d. The size of the problem. This includes the number of variables and the number of constraints. Computation time and memory required by a program as well as numerical stability of a program depend on the size of the problem. An important criterion for the usefulness of a program is the maximum size of the problem that can generally be solved by the program.
e. First and/or second order partial derivatives are calculated analytically or numerically (see also COLVILLE [1968]). One reason for distinguishing between analytically and numerically calculated derivatives is the fact that numerical approximation does not require the same amount of computation time as evaluation of the analytical derivatives. The ratio between these quantities depends heavily on the problem. Therefore, the efficiency of a program may be highly influenced by the way the derivatives are calculated. A second reason for this distinction is that one program may be more sensitive to errors due to approximation of the derivatives than another. And finally the program tester should be able to recognize whether a program break-down is due to numerical

approximation of the derivatives or to something else.

f. The problem is a fully analysed theoretical problem. The functions can be calculated in almost full precision of arithmetic and the solution is also known in full precision of arithmetic. Clearly, this property is not relevant to the user, in fact his problems do not have this property. However, it is very important for practical testing to have such problems at hand for a careful examination of the program to be tested, since for real-life problems rounding errors may confuse the algorithmic aspects, to be tested.

g. The problem is convex. Some programs may take advantage of this property (FIACCO & McCORMICK [1968]). However the user will frequently be unable to prove convexity.

h. The objective function has several local minima or other stationary points in the feasible region. In this case, one usually cannot expect that the program finds the global solution. Moreover, the program may break down in the neighbourhood of a stationary point which is not a local minimum. As an example one may consider Box' function (BOX [1966]). As is illustrated in BUS [1972] gradient methods sometimes break down on this problem.

i. The hessian of the Lagrangian function at the solution is ill-conditioned or even singular. Usually such a property makes a problem difficult to solve. As an example one may consider the problem of calculating the unconstrained minimum of Powell's function of four variables (POWELL [1962]). Numerical results with this function are also reported in FLETCHER [1970] and BUS [1975].

Usually, the properties a to e can be verified for real-life problems. Therefore, they are suitable as primary classification criteria. However, the properties g to i may be difficult or even impossible to verify in practice. Therefore, these properties should not be used as primary classification criteria. We will give them as "special properties" in the documentation so that they can be used for testing. These properties may give an indication for the degree of difficulty of a problem. However, one can imagine other properties that make a problem difficult to solve by some program. In our opinion it would be desirable to develop measures for the degree of difficulty of a problem so that we may create graded sets of

testproblems. We think that it is easy to incorporate such measures in the classification and documentation scheme proposed here at the time they are available.

The classification scheme

The classification number of a problem has the form

OCD-KI-s,

where the letters has the following meaning:

O reflects properties of the objectfunction:

- O = S : the objective function is a sum of squares;
- L : linear objective function;
- Q : quadratic objective function;
- G : nonlinear, non-quadratic objective function which is no sum of squares.

C reflects properties of the constraints:

- C = U : unconstrained problem;
- L : linear constraints;
- N : nonlinear constraints.

D reflects the differentiability of the problem functions:

- D = R : the problem is "regular"; at least the first and second derivatives of the problem functions exist on the feasible region;
- I : "irregular" problem; there are points in the feasible region where the first and/or second derivative of one of the problem functions do not exist.

K denotes whether a problem is a so-called theoretical problem or a practical problem:

- K = T : "theoretical" and well-analyzed problem; in order to avoid ambiguity we use as a criterion that the solutions of the

problem are given in full precision

P : "practical" problems; all problems which are not theoretical in the sense given above.

I denotes which partial derivatives are calculated analytically:

I = 2 : first and second order partial derivatives are calculated analytically;

1 : the first order partial derivatives are calculated analytically;

0 : no partial derivatives are calculated analytically.

s gives a serial number within the class of testproblems identified by OCD-KI.

Remarks

1. The classification code is split into two groups. The first group gives information about the exact problem which is relevant to the tester as well as to the user. The second group is mainly relevant to the tester. This group gives information how the problem is given in the testset and in the documentation.
2. The properties a,b,c,e and f are reflected in this classification code. The other properties are given in standard format in the heading of the problem documentation.

Example

The problem

$$\text{minimize } f(x) = 100(x_2 - x_1^2)^2 + (1-x_1)^2$$

belongs to class SUR-T2. When its serial number within this class is 1, we denote this problem by SUR-T2-1.

3. DOCUMENTATION OF TEST PROBLEMS

PROPOSAL

PROBLEM: OCD-KI-s

NAME: name of the problem, if it has any

SOURCE: author [year], problem/page number

NO. OF VARIABLES: N

NO. OF CONSTRAINTS:

bounds on variables	M1
linear equalities	M2
linear inequalities	M3
nonlinear equalities	M4
nonlinear inequalities	M5

(Note: if the problem is defined for fixed values of N and/or M1 to M5 then these values are specified here, otherwise they are considered as parameters; one or more may be expressed as formulas depending on the others. Specific values for these parameters are given in the block DATA AND RESULTS.)

SPECIAL PROPERTIES:

convex problem	yes/no/unknown
several stationary points	yes/no/unknown
condition hessian of Lagrangian	= .../≤ .../unknown

(Note: quantities given here may depend on parameters of the problem)

OBJECTIVE FUNCTION:

$f(x) = \ldots$,

or, if $f(x)$ is a sum of squares:

$f(x) = \sum_{i=1}^{p} [f_i(x)]^2$

$f_1(x) = \ldots$

$\vdots$

$f_p(x) = \ldots$

CONSTRAINTS:

$l_{i_j} \le x_{i_j} \le u_{i_j}$ or $l_{i_j} \le x_{i_j}$ or $x_{i_j} \le u_{i_j}$ (the total number of inequality signs equals M1)

$h_j(x) = 0, \quad j = 1,\ldots,M2$

$g_j(x) \ge 0, \quad j = 1,\ldots,M3$

$h_j(x) = 0, \quad j = M2+1,\ldots,M2+M4$

$g_j(x) \ge 0, \quad j = M3+1,\ldots,M3+M5$

(Note: the objective and constraint functions may depend on quantities (parameters) whose value(s) are specified in the block DATA AND RESULTS.)

DATA AND RESULTS:

(Note: we give here the starting point(s), the results and all additional data necessary to define the problem uniquely. This part of the documentation may consist of several blocks if the problem depends on parameters (e.g.: N, M1 to M5 or parameters in the definition of the problem function). Then each block defines one problem. Such a problem may have several starting points and several local solution points and is identified by OCD-KI-s/i, where i is the number of the block.)

BLOCK i (only if more than one block is given)

starting point(s)

a. $x^{(0)} = [\ \ldots\ ]^T$ (non)-feasible

$f(x^{(0)}) = \ldots$

b. :

additional data

(for example, if N is a parameter of the problem which is given the value 10)

N = 10

results:

a. $x^* = [\ \ldots\ ]^T$

$f(x^*) = \ldots$

b. :

PRECISIONS:

We define here tolerance values ε_1 and ε_2 which are to be regarded as input to a program to be tested.
We say that this program is successful in solving this problem if the computed solution, $\bar{x}$, satisfies:

$$\|\bar{x}-x^*\| \leq \varepsilon_1\|x^*\| + \varepsilon_2,$$

for some solution point x^*. We distinguish three levels of precision which, in general, depend on the precision of arithmetic used and on the rounding errors in the evaluation of the functions. We suggest:

high precision: ε_1 and ε_2 as small as possible with respect to rounding errors in the function.
low precision: $\varepsilon_1 = \varepsilon_2 = 10^{-2}$.
medial precision: precisions between low and high precision.

ADDITIONAL DETAILS:

A description of typical properties of the problem should be given here.
We mention:
- the precision of the data;
- the precision of the calculated values of the problem functions;
- further information about other stationary points;
- further information about the condition of the hessian of the Lagrangian function or the objective function;
- the shape of the feasible region;
- appearance of linear variables that may be separated from variables that appear nonlinearly;
- nonlinearity of the problem functions.

These and other properties may be illustrated by figures and tables.

DERIVATIVES:

If the I parameter in the classification code equals 0 then no derivatives are given;
If I = 1 then we give here

$\frac{\partial f}{\partial x_i}$, $i = 1,\ldots,N$;

$\frac{\partial f_i}{\partial x_j}$, $i = 1,\ldots,P$, $j = 1,\ldots,N$, if f is a sum of squares;

$\frac{\partial h_i}{\partial x_j}$, $i = 1,\ldots,M2+M4$, $j = 1,\ldots,N$;

$\frac{\partial g_i}{\partial x_j}$, $i = 1,\ldots,M3+M5$, $j = 1,\ldots,N$.

If I = 2 then we also give

$\frac{\partial^2 f}{\partial x_i \partial x_j}$, $i,j = 1,\ldots,N$;

$\frac{\partial^2 f_i}{\partial x_j \partial x_k}$, $i = 1,\ldots,P$, $j,k = 1,\ldots,N$, if f is a sum of squares;

$\frac{\partial^2 h_i}{\partial x_j \partial x_k}$, $i = 1,\ldots,M2+M4$, $j,k = 1,\ldots,N$;

$\frac{\partial^2 g_i}{\partial x_j \partial x_k}$, $i = 1,\ldots,M3+M5$, $j,k = 1,\ldots,N$.

PROGRAMS:

These programs will be given in FORTRAN, ALGOL 60 and ALGOL 68. The parameterlists are:

fun	(n,x,fx)
funsq	(n,x,i,fx)
dfun	(n,x,dfx)
dfunsq	(n,x,i,dfx)
ddfun	(n,x,ddfx)

(proposal contin.)

```
ddfunsq   (n,x,i,ddfx)
 constr   (n,x,j,gx)
dconstr   (n,x,j,dgx)
ddconstr  (n,x,j,ddgx)
```

funsq, dfunsq and ddfunsq are given if the function is a sum of squares, otherwise fun, dfun and ddfun are given. ddfun, ddfunsq and ddconstr are only given if I = 2, dfun, dfunsq and dconstr are only given if I = 1 or I = 2. The parameters have the following meaning:

n : input, the number of variables;

i : input, the index of the term in the sum of squared terms which has to be evaluated or whose derivative has to be evaluated;

j : input, the index of the constraint function to be evaluated; these functions are given in the same order as in the heading of the documentation;

x : input, the vector of variables;

fx : output, the value of the objective function (fun) or the i-th term of the sum of squares (funsq);

dfx : output, the gradient vector of the objective function (dfun) or of the i-th term of a sum of squares (dfunsq);

ddfx : output, the matrix of second order partial derivatives of the objective function (ddfun) or of the i-th term of a sum of squares (ddfunsq);

gx : output, the value of the i-th constraint function;

dgx : output, the gradient vector of the i-th constraint function;

ddgx : output, the matrix of second order partial derivatives of the i-th constraint function.

TESTREPORTS:

Numerical experience with this problem has been reported in:

....

....

(One may refer here to papers given in literature and also to unpublished experiences which will be given in appendices to this documentation.)

REFERENCES:

...

(end of proposal).

Remarks

1. All problems are described as minimization problems.
2. Problems with an objective function which is a sum of squares (0=S) may also be considered as a normal problem. It is easy to program the derivatives of such a problem using the derivatives of terms of the sum of squares.
3. Stopping criteria have to be part of the program to be tested. Therefore, the precision of the solution vector asked for should be input to a program and we say that a program has failed to solve a problem if the computed solution does not satisfy the conditions given in the documentation. When comparing the efficiency one should judge a program by the work that has to be done to satisfy its own stopping criteria, provided the program did not fail in the above sense.
4. Program source text are given such that the objective function and its derivatives and the various constraint functions are evaluated separately. This may be an inefficient way to solve these particular test problems with a given program.
   For example, some programs for unconstrained minimization only ask for evaluation of the function and its gradient at the same point and for some problems it may save a lot of computation time if both the function and its gradient are given in one routine. However, if one uses computation time as a measure for the efficiency, one should measure the time required to solve the problem minus the time required for the evaluation of the problem functions and apart from this the number of problem function evaluations. In this manner one obtains a measure for the efficiency which does not depend on the time necessary for the evaluation of the problem functions. This is very desirable since otherwise we would also

introduce the evaluation time as a property of the problem. In our opinion, giving the source-texts as we propose will be adequate for testing programs in the way given above or some other way which does not use the total computation time as a measure for the efficiency of a program.

5. The programming of the problem functions will be such that run-time errors due to limitations of the arithmetrical system of a computer are avoided. For example, overflow/underflow, exponential or logarithm errors will not occur. Therefore, we need to introduce a number of machine constants, which are assumed to be known globally. By now we confine ourselves to refer to work done by the IFIP Working Group 2.5 on Numerical Software, especially to FORD & SMITH [1976a,1976b], CODY [1976] and DEKKER [1976].
6. Our ultimate goal is to present a set of testproblems in a manual which can be upgraded from time to time. Apart from this manual we should have available short documentation in machine readable form. This may consist of the outlined parts at the heads of the problem documents together with the DATA AND RESULTS, the PRECISIONS and the PROGRAMS parts. In our opinion this will be all that is required by a driver program to test programs, so that the data for such a driver program can be obtained automatically.

REFERENCES

BOX, M.J. [1966], *A comparison of several current optimization methods and the use of transformations in constrained problems*, Comp. J. 9, 67-77.

BUS, J.C.P. [1972], *Minimization of functions of several variables* (dutch), Mathematical Centre, NR 29/72, Amsterdam.

BUS, J.C.P. [1975], *On the convergence of a class of variable metric algorithms*, Mathematical Centre, NW 16/75, Amsterdam.

CODY, W.J. [1976], *Machine parameters for numerical analysis*, Working paper of IFIP Working Group 2.5 on Numerical Software.

COLVILLE, A.R. [1968], *A comparative study of nonlinear programming codes*, IBM, New York Sc. Center, TR 320-2949.

DEKKER, T.J. [1976], *Machine requirements for reliable portable software*, University of Amsterdam, Dpt. of Math. Rep. 76-15.

FIACCO, A.V. & G.P. McCORMICK [1968], *NOnlinear programming: sequential unconstrained minimization techniques*, Wiley Interscience.

FLETCHER, R. [1970], *A new approach to variable metric algorithms*, Comp. J. 13, 317-322.

FORD, B. & B.T. SMITH [1976a], *Intrinsic functions to assist FORTRAN portability for numeric computation*, A proposal to ANS FORTRAN Standards Committee X3J3 from IFIP WG 2.5.

FORD, B. & B.T. SMITH [1976b], *Parameters for transportable numerical software*, Working paper of IFIP Working Group 2.5 on Numerical Software.

LOOTSMA, F.A. [1976], *Nonlinear optimization in industry and the development of optimization programmes*, Paper presented at the IX Internat. Symp. on Mathematical Programming, Budapest.

POWELL, M.J.D. [1962], *An iterative method for finding stationay values of a function of several variables*, Comp. J. 5, 147-151.

APPENDIX *(example of documentation)*

PROBLEM: SUR-T2-1

NAME: Rosenbrock's parabolic valley

SOURCE: Rosenbrock [1960]

NO. OF VARIABLES: 2

NO. OF CONSTRAINTS:

bounds on variables	0
linear equalities	0
linear inequalities	0
nonlinear equalities	0
nonlinear inequalities	0

SPECIAL PROPERTIES:

convex problem	no
several stationary points	no
condition hessain of Lagrangian	2504

OBJECTIVE FUNCTION:

$$f(x) = \sum_{i=1}^{2} [f_i(x)]^2 = 100(x_2 - x_1^2)^2 + (1-x_1)^2$$

$$f_1(x) = 10(x_2 - x_1^2)$$

$$f_2(x) = (1-x_1)$$

DATA AND RESULTS:

starting point(s)

$$x^{(0)} = [-1.2, 1]^T$$

$$f(x^{(0)}) = 24.20$$

results

$x^* = [1,1]^T$

$f(x^*) = 0$

PRECISIONS

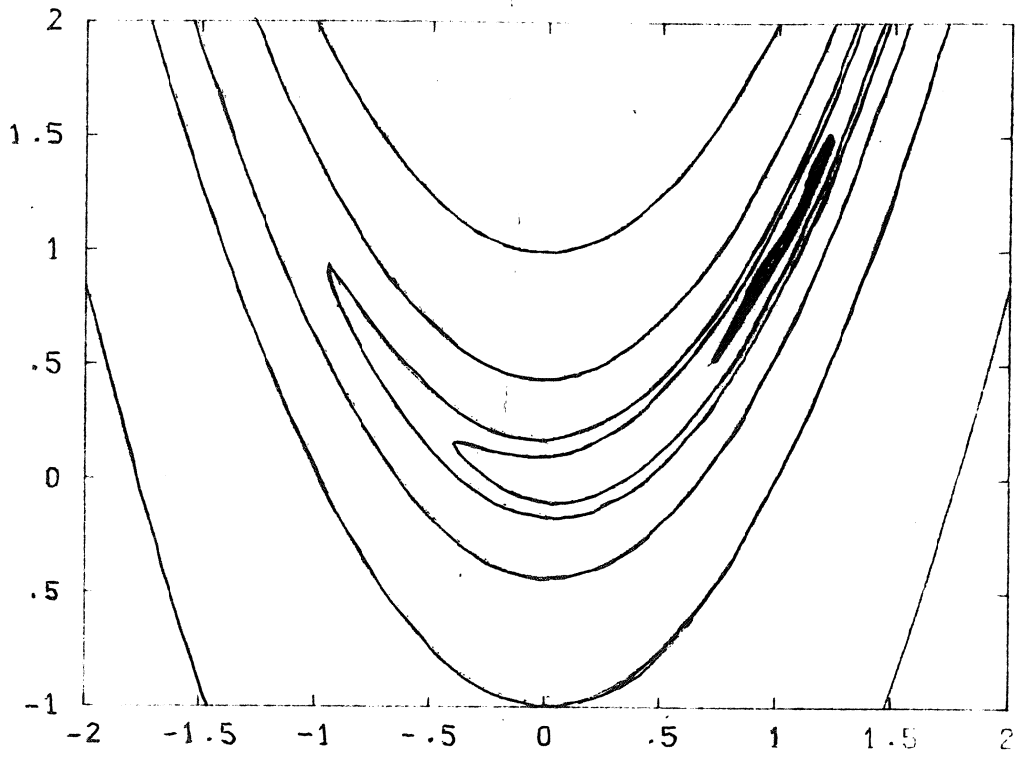
high precision : $\varepsilon_1 = \varepsilon_2 = \sqrt{\varepsilon}$, with ε the precision of arithmetic

medial precision : $\varepsilon_1 = \varepsilon_2 = \frac{1}{2}(^{10}\log(\sqrt{\varepsilon})-2)$

low precision : $\varepsilon_1 = \varepsilon_2 = 10^{-2}$

ADDITIONAL DETAILS

This function has a steep-sided parabolic valley, which is shown in the following figure



contours of Rosenbrock's function

DERIVATIVES:

$$\frac{\partial f_1}{\partial x_1} = -20x_1 \qquad \frac{\partial f_1}{\partial x_2} = 10$$

$$\frac{\partial f_2}{\partial x_1} = -1 \qquad \frac{\partial f_2}{\partial x_2} = 0$$

$$\frac{\partial^2 f_1}{\partial x_1 \partial x_1} = -20 \qquad \frac{\partial^2 f_1}{\partial x_2 \partial x_1} = 0$$

$$\frac{\partial^2 f_1}{\partial x_1 \partial x_2} = 0 \qquad \frac{\partial^2 f_1}{\partial x_2 \partial x_2} = 0$$

$$\frac{\partial^2 f_2}{\partial x_1 \partial x_1} = 0 \qquad \frac{\partial^2 f_2}{\partial x_2 \partial x_1} = 0$$

$$\frac{\partial^2 f_2}{\partial x_1 \partial x_2} = 0 \qquad \frac{\partial^2 f_2}{\partial x_2 \partial x_2} = 0$$

PROGRAMS

ALGOL 60

```
procedure funsq(n,x,i,fx); value n,i;
integer n,i; real fx; array x;
fx := if i = 1 then (x[2]-x[1]**2)*10 else 1 - x[1];

procedure dfunsq(n,x,i,dfx); value n,i;
integer n,i; array x,dfx;
if i = 1 then
begin dfx[1] := -x[1]*20; dfx[2] := 10 end else
begin dfx[1] := -1; dfx[2] := 0 end dfunsq;

procedure ddfunsq(n,x,i,ddfx); value n,i;
integer n,i; array x,ddfx;
```

```
    if i = 1 then
    begin ddfx[1,1] := -20;
       ddfx[1,2] := ddfx[2,1] := ddfx[2,2] := 0
    end else
    begin ddfx[1,1] := ddfx[1,2] := ddfx[2,1] := ddfx[2,2] := 0
    end ddfunsq;
```

FORTRAN

.
.
.

ALGOL 68

.
.
.

TESTREPORTS:

Numerical experience with this problem has been reported in various papers. We mention

FLETCHER [1970]

FLETCHER & POWELL [1963].

REFERENCES:

FLETCHER, R. [1970], A new approach to variable metric algorithms, Comp. J. 13, 317-322.

FLETCHER, R. & M.J.D. POWELL [1963], A rapidly convergent descent method for minimization, Comp. J. 6, 163-168.

ROSENBROCK, H.H. [1960], An automatic method for finding the greatest or least value of a function, Comp. J. 3, 175-184.