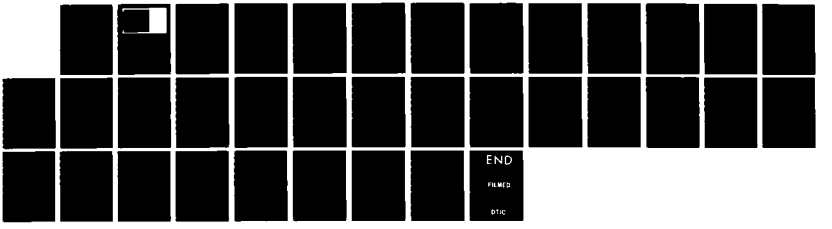
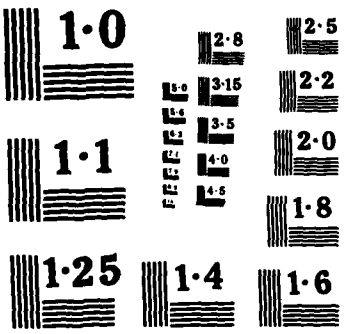
AD-A158 150 GLOBAL OPTIMIZATION USING AUTOMATIC DIFFERENTIATION AND INTERVAL ITERATION(U) WISCONSIN UNIV-MADISON MATHEMATICS RESEARCH CENTER L B RALL JUN 85 1/1

UNCLASSIFIED MRC-TSR-2832 DAAG29-80-C-0041 F/G 12/1 NL





NATIONAL BUREAU OF STANDARDS
MICROCOPY RESOLUTION TEST CHART

(2)

AD-A158 150

MRC Technical Summary Report #2832

GLOBAL OPTIMIZATION USING AUTOMATIC DIFFERENTIATION AND INTERVAL ITERATION

L. B. Rall

Mathematics Research Center
University of Wisconsin—Madison
610 Walnut Street
Madison, Wisconsin 53705

June 1985

(Received June 11, 1985)

Approved for public release
Distribution unlimited

Sponsored by

U. S. Army Research Office
P. O. Box 12211
Research Triangle Park
North Carolina 27709

UNIVERSITY OF WISCONSIN-MADISON
MATHEMATICS RESEARCH CENTER

GLOBAL OPTIMIZATION USING AUTOMATIC DIFFERENTIATION AND INTERVAL ITERATION

L. B. Rall

Technical Summary Report #2832
June 1985

ABSTRACT

Algorithms are presented which find one or all of the critical points of a smooth function in a rectangular region, or the critical points at which the function has maximum or minimum value. If no critical points of the function exist in the given region, then the algorithm verifies this fact. The computation is self-validating, in that the existence or nonexistence of critical points is established conclusively, and guaranteed upper and lower bounds are computed for all quantities of interest, including the values of the gradient vector and Hessian matrix of the function. The algorithm makes use of an existing implementation of automatic differentiation and interval computation. Numerical examples are given.

AMS (MOS) Subject Classifications: 65K10, 65G10, 68Q40

Key words: Global unconstrained optimization, Critical points, Automatic differentiation, Interval iteration, Self-validating computation

Work Unit Number 3 - Numerical Analysis and Scientific Computing

Sponsored by the United States Army Under Contract No. DAAG29-80-C-0041.

SIGNIFICANCE AND EXPLANATION

A standard problem in scientific computation is the optimization of a smooth (at least twice differentiable) function, possibly subject to smooth constraints. Here, optimization means finding the maximum or minimum value of the function in some region, or perhaps its stationary points. The algorithm presented in this paper solves this problem for regions which are rectangular in form. The method used is a version of interval iteration which finds one or all of the critical points of the function in the given region (the point or points at which its gradient vector vanishes), or just the critical points at which the function has its maximum or minimum values. The required values of the gradient vector and Hessian matrix of the function are obtained by automatic differentiation, which does not involve symbolic differentiation or numerical approximation of the derivatives of the function being investigated. Unlike algorithms which sample function values only at a discrete set of points, the ones given here use interval computation to furnish guaranteed bounds for all quantities of interest over the required subregions of the initial region. The results given by these optimization algorithms are thus self-validating. The algorithms presented here have been programmed using standard implementations of automatic differentiation and interval computation. Numerical examples are given to illustrate the effectiveness of this approach to the type of optimization problem considered.

The responsibility for the wording and views expressed in this descriptive summary lies with MRC, and not with the author of this report.

GLOBAL OPTIMIZATION USING AUTOMATIC DIFFERENTIATION AND INTERVAL ITERATION

L. B. Rall

1. Preliminaries

This paper presents an algorithm for global, unconstrained optimization of a smooth (at least twice differentiable) function $f : \mathbf{R}^n \to \mathbf{R}$, that is,

$$f(x) = f(x_1, x_2, \ldots, x_n). \tag{1.1}$$

As is well-understood, this also includes the case of optimization of a function $\phi : \mathbf{R}^m \to \mathbf{R}$ subject to $n - m$ smooth constraints

$$g_i(x_1, x_2, \ldots, x_m) = 0, \qquad i = 1, 2, \ldots, n - m, \tag{1.2}$$

by formation of the function

$$f(x) = \phi(x_1, \ldots, x_m) + \sum_{i=1}^{n-m} x_{m+i} \cdot g_i(x_1, \ldots, x_m), \tag{1.3}$$

where the new variables $x_{m+1}, \ldots, x_n$ are simply the Lagrange multipliers for the problem. No special properties of f, such as convexity, are assumed.

The method to be used is a *critical point* method, which will find one or all solutions of the system of equations

$$\nabla f(x) = 0 \tag{1.4}$$

in a rectangular region $X \subset \mathbf{R}^n$, where $\nabla f(x)$ denotes the *gradient vector*

$$\nabla f(x) = \left(\frac{\partial f(x)}{\partial x_1}, \frac{\partial f(x)}{\partial x_2}, \ldots, \frac{\partial f(x)}{\partial x_n} \right)^T, \tag{1.5}$$

Sponsored by the United States Army under Contract No. DAAG29-80-C-0041.

or just the critical points at which the value of f is a maximum or minimum in X. Such points will be called *critical extremal points* to distinguish them, if necessary, from non-critical points on the boundary ∂X of X at which f might attain a maximum or minimum value.

The algorithm will make use of automatic differentiation [11] to compute the gradient vector $\nabla f(x)$ of f at $x = (x_1, x_2, \ldots, x_n)$, and also its *Hessian matrix*

$$Hf(x) = \left(\frac{\partial^2 f(x)}{\partial x_i \partial x_j}\right). \tag{1.6}$$

This technique will be combined with the use of interval arithmetic and interval evaluation of library functions [8] in order to compute guaranteed bounds for values of functions and their derivatives over the region of interest. The result will be an automatic, self-validating optimization algorithm.

Automatic differentiation has been used, at least in a restricted form, by McCormick [6] for optimization problems. Interval methods have been applied by Hansen [2], [3] and Hansen and Sengupta [4] to global optimization problems, including constrained problems. Although the basic algorithm given below is for unconstrained problems, the ideas presented by Hansen indicate the possibility of introducing constraints into the calculations.

2. Automatic Differentiation

The basic idea behind automatic differentiation is to use the formula or subroutine for the evaluation of the function f at x to obtain also values of its derivatives at the same point. This is done by the introduction of a new representation of variables, and arithmetic operations which include the rules for differentiation. The resulting computational scheme is simple to program for computers [11], [13], and avoids both the complexity of symbolic differentiation and the inaccuracy of numerical differentiation. The new variables are *triples*

$$U = (u, u', u''), \tag{2.1}$$

where $u \in \mathbf{R}$ is a real number, $u' \in \mathbf{R}^n$ is an n-dimensional real (column) vector, and u'' is a symmetric real $n \times n$ matrix. The set of these elements will be denoted by $\mathbf{H}^n$, and each $u \in \mathbf{H}^n$ is said to be of type HESSIAN. A variable U of type HESSIAN will be interpreted in the following way: Its first component u will represent the value of a real-valued function at some point $x \in \mathbf{R}^n$, and u' and u'' the values of its gradient vector and Hessian matrix, respectively, at the same point.

It is obvious that $\mathbf{H}^n$ forms a linear space. More importantly, *all* the standard arithmetic operations can be defined in $\mathbf{H}^n$:

$$U + V = (u, u', u'') + (v, v', v'') = (u + v, u' + v', u'' + v''), \tag{2.2}$$

$$U - V = (u, u', u'') - (v, v', v'') = (u - v, u' - v', u'' - v''), \tag{2.3}$$

$$\begin{aligned} U \cdot V &= (u, u', u'') \cdot (v, v', v'') \\ &= (u \cdot v, u \cdot v' + v \cdot u', u \cdot v'' + u' v'^T + v' u'^T + v \cdot u''), \end{aligned} \tag{2.4}$$

$$\begin{aligned} U/V &= (u, u', u'')/(v, v', v'') \\ &= \left(\frac{u}{v}, \frac{v \cdot u' - u \cdot v'}{v^2}, \frac{v^2 \cdot u'' - v \cdot (v' u'^T + u' v'^T) + 2u \cdot v' v'^T - uv \cdot v''}{v^3}\right), \\ & v \neq 0. \end{aligned} \tag{2.5}$$

The above definitions implement the rules for evaluation and differentiation of sums, differences, products, and quotients of functions with known values and derivatives. In order to use an algorithm for evaluation of a real function to obtain the corresponding values in $\mathbf{H}^n$, it is necessary to be able to represent the independent variables x_i, $i = 1, 2, \ldots, n$ and constants c as elements of $\mathbf{H}^n$. This is done by the mapping

$$x_i \mapsto (x_i, e_i, 0), \tag{2.6}$$

for the ith independent variable x_i, where e_i denotes the ith unit vector, and 0 the $n \times n$ zero matrix. (0 will be used to denote zero vectors and matrices, as well as the real number zero.) Similary, constants c are represented by

$$c \mapsto (c, 0, 0). \tag{2.7}$$

It follows that calculation of the value, gradient vector, and Hessian matrix of a rational function can be done simply by making the substitutions (2.6) and (2.7), and applying the rules (2.2)–(2.5). The results are exact, not numerical approximations, and are obtained without symbolics.

In actual practice, instead of using the representation (2.7) for constants, it is simpler to define a *mixed arithmetic* between elements $c \in \mathbf{R}$ and $U = (u, u', u'') \in \mathbf{H}^n$ [13]:

$$c + U = U + c = (c + u, u', u''), \tag{2.8}$$

$$c - U = (c - u, -u', -u''), \tag{2.9}$$

$$U - c = (u - c, u', u''), \tag{2.10}$$

$$c \cdot U = U \cdot c = (c \cdot u, c \cdot u', c \cdot u''), \tag{2.11}$$

$$c/U = \left(\frac{c}{u}, -\frac{c \cdot u'}{u^2}, \frac{2c \cdot u'u'^T - cu \cdot u''}{u^3}\right), \quad u \neq 0, \tag{2.12}$$

$$U/c = \left(\frac{u}{c}, \frac{u'}{c}, \frac{u''}{c}\right), \quad c \neq 0. \tag{2.13}$$

For example, consider the two-dimensional Rosenbrock function ([1], p. 95):

$$f(x) = 100(x_2 - x_1)^2 + (1 - x_1)^2. \tag{2.14}$$

In order to evaluate this function together with its gradient vector and Hessian matrix at the point $x = (-1.2, 1.0)$, one sets

$$x_1 = \left(-1.2, \begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}0 & 0\\0 & 0\end{pmatrix}\right), \qquad x_2 = \left(1.0, \begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}0 & 0\\0 & 0\end{pmatrix}\right), \tag{2.15}$$

and evaluates (2.14) using the above rules. The result is

$$(f(x), \nabla f(x), Hf(x)) = \left(24.2, \begin{pmatrix}-215.6\\-88.0\end{pmatrix}, \begin{pmatrix}1330.0 & 480.0\\480.0 & 200.0\end{pmatrix}\right), \tag{2.16}$$

which is exactly what one would get by differentiating (2.14) symbolically and then evaluating the results for $x_1 = -1.2$, $x_2 = 1.0$ in real arithmetic.

In addition to rational functions of several variables, other standard functions can be defined readily on $\mathbf{H}^n$. For example,

$$\sin U = \sin(u, u', u'') = \left(\sin u, \cos u \cdot u', \cos u \cdot u'' - \sin u \cdot u' u'^T\right). \tag{2.17}$$

In general, if $g : \mathbf{R} \to \mathbf{R}$ is twice differentiable, then it can be extended immediately to the mapping $g : \mathbf{H}^n \to \mathbf{H}^n$ by use of the chain rule:

$$g(U) = g\left((u, u', u'')\right) = \left(g(u), g'(u) \cdot u', g'(u) \cdot u'' + g''(u) \cdot u' u'^T\right), \tag{2.18}$$

[11], [13].

It is easy to program automatic differentiation in languages such as Ada and Pascal-SC [13], which permit introduction of data types and additional definitions of the standard operator symbols to manipulate such types. (This is sometimes called "overloading" the standard operator symbols.) In these languages, the variables x_1 and x_2 in (2.14) would

declared to be of type HESSIAN, along with the result f, and the evaluation would be ried out on the basis of an expression of the same form as (2.14). In ordinary Pascal FORTRAN, (2.14) would have to be rewritten as a sequence of calls to subroutines addition, exponentiation, etc. [11]. The algorithms described in this paper have been grammed in Pascal-SC, and some results are given in the final section.

Interval Computation

ordinary optimization algorithms, the function to be optimized is sampled only at a crete set of points. This can result in the loss of valuable information about the function. e algorithms presented in this paper, on the other hand, use interval computation, which oduces guaranteed bounds for the values of functions and their derivatives over entire gions [8]. This prevents the process from being misled by incomplete information.

The basic component of interval computation is *interval arithmetic* [8]. Let $\mathbf{IR}$ denote e set of bounded, closed intervals on the real line $\mathbf{R}$. For $I = [a,b] \in \mathbf{IR}$, $J = [c,d] \in \mathbf{IR}$, e arithmetic operations are defined by

1) $$I \star J = [a,b] \star [c,d] = \{x \star y \mid x \in I,\ y \in J\} = [r,s],$$

ere $\star \in \{+,-,\cdot,/\}$, and division by an interval containing 0 is excluded. In actual plementation on computers, directed rounding is used (downward for lower endpoints, ward for upper endpoints), so the actual result computed is $[\nabla r, \Delta s]$, which always ntains the exact result $[r,s]$ of the interval operation.

Evaluation of a real rational function $f : \mathbf{R} \to \mathbf{R}$ in interval arithmetic results in an *erval inclusion* $F : \mathbf{IR} \to \mathbf{IR}$ of f [8], which has the property

2) $$f(X) = \{f(x) \mid x \in X\} \subseteq F(X), \qquad X \in \mathbf{IR}.$$

noting the endpoints of an interval $I = [a,b]$ by $\inf I = a$, $\sup I = b$, respectively, (3.2) ans that

3) $$\inf F(X) \le f(x) \le \sup F(X). \qquad x \in X.$$

These bounds for the range of $f(x)$ over X are obtained *automatically*, without investigation of the minimum and maximum values of $f(x)$ on X, and are furthermore *guaranteed* (although they may be somewhat crude) [8]. This is the basis of the self-validating character of interval computation. Furthermore, interval extensions obtained by using interval arithmetic are *monotone* in the sense that $X \subseteq Y$ imples that $F(X) \subseteq F(Y)$. In exact arithmetic, F is an *extension* of f in the sense that $F([x,x]) = f(x)$ for $x \in \mathbf{R}$ [8]. In what follows, x will be used to denote the degenerate interval $[x,x] \in \mathbf{IR}$ as well as the real number x. Other handy notations to be used from time to time are

$$w(I) = w([a,b]) = b - a, \qquad m(I) = m([a,b]) = \frac{a+b}{2}, \tag{3.4}$$

for the width and midpoint, respectively, of an interval $I \in \mathbf{IR}$.

Just as in the case of differentiation arithmetic, interval arithmetic can be extended to include various standard functions encountered in applications. Efficient implementations of interval arithmetic and interval inclusions of standard functions are now available in a number of computational environments, for example, Pascal-SC for microcomputers and the *ACRITH* package for *IBM 370* computers.

The space $\mathbf{IR}^n$ of interval vectors $X = (X_1, X_2, \ldots, X_n)$ is defined in the same way as $\mathbf{R}^n$, and the notions of interval matrices and vector and matrix-vector interval arithmetic arise in a natural way. The *interval scalar product* of interval vectors X, Y is defined to be

$$X \cdot Y = \left\{ \sum_{i=1}^{n} x_i y_i \;\middle|\; x_i \in X_i, y_i \in Y_i \right\}, \tag{3.5}$$

and the notation $m(X)$ will be used for the *midpoint*

$$m(X) = (m(X_1), m(X_2), \ldots, m(X_n)) \tag{3.6}$$

of the interval vector X.

Now, if the evaluation of the function (2.14) is performed in interval arithmetic with $x_1 = [0.9, 1.2]$, $x_2 = [0.8, 1.1]$, then the result is

$$F(X) \quad [0.0, 41.0]. \tag{3.7}$$

X denotes the interval vector $X = ([0.9, 1.2], [0.8, 1.1])$. This means that

$$0 \le f(x) \le 41$$

$\le x_1 \le 1.2$, $0.8 \le x_2 \le 1.1$. Thus, the bounds (3.8) are obtained automatically, by evaluation of (2.14) in interval arithmetic, in much the same way that values gradient vector and Hessian matrix of (2.14) were obtained in §2 by the use of ıtiation arithmetic. Furthermore, as stated above, the bounds given by interval etic are guaranteed to be valid.

he next step is to combine the differentiation arithmetic in §2 with interval arith- An element Υ of type IHESSIAN will be a triple

$$\Upsilon = (U, U', U''),$$

$U \in \mathbf{IR}$ is an interval, $U' \in \mathbf{IR}^n$ is an interval (column) vector, and U'' is a symmet- rval $n \times n$ matrix. The resulting set of elements will be denoted by $\mathbf{IH}^n$. Arithmetic ions in $\mathbf{IH}^n$ are defined by (2.2)-(2.5), with the operations inside the parentheses d by the interval operations (3.1). Similarly, operations between constants $c \in \mathbf{IR}$ ments of $\mathbf{IH}^n$ are defined by (2.8)-(2.13) and the corresponding interval operations. onstants c are mapped into $\mathbf{IR}$ by $c \mapsto [c, c]$, as before.

r example, the evaluation of (2.14) as type IHESSIAN can be carried out over the ls $0.9 \le x_1 \le 1.2$, $0.8 \le x_2 \le 1.1$ by setting

$$x_1 = \left([0.9, 1.2], \begin{pmatrix} [1,1] \\ [0,0] \end{pmatrix}, \begin{pmatrix} [0,0] & [0,0] \\ [0,0] & [0,0] \end{pmatrix}\right),$$

$$x_2 = \left([0.8, 1.1], \begin{pmatrix} [0,0] \\ [1,1] \end{pmatrix}, \begin{pmatrix} [0,0] & [0,0] \\ [0,0] & [0,0] \end{pmatrix}\right).$$

ılt is

$$\Phi(X) = (F(X), F'(X), F''(X)) =$$

$$= \left([0.0, 41.0], \begin{pmatrix} [-139.4, 307.6] \\ [-128.0, 58.0] \end{pmatrix}, \begin{pmatrix} [534.0, 1410.0] & [-480.0, -360.0] \\ [-480.0, -360.0] & [200.0, 200.0] \end{pmatrix}\right),$$

where X denotes the interval vector $X = ([0.9, 1.2], [0.8, 1.1])$. This gives not only the bounds (3.8) for $f(x)$ over X, but also the bounds

$$\nabla f(x) \in F'(X) = \begin{pmatrix} [-139.4, 307.6] \\ [-128.0, 58.0] \end{pmatrix}, \tag{3.12}$$

for the gradient vector of f, and

$$Hf(x) \in F''(X) = \begin{pmatrix} [534.0, 1410.0] & [-480.0, -360.0] \\ [-480.0, -360.0] & [200.0, 200.0] \end{pmatrix}, \tag{3.13}$$

for the Hessian matrix of f over X. These bounds, obtained automatically by the use of IHESSIAN arithmetic, are guaranteed.

In addition to bounds for the values of f and its derivatives on X, the IHESSIAN computation (3.11) provides information about the continuity of f and ∇f on X. For an interval $I = [a, b] \in \mathbf{IR}$, let

$$|I| = |[a, b]| = \max\{|a|, |b|\}. \tag{3.14}$$

If $X = (X_1, X_2, \ldots, X_n)$ is an interval vector, then $\|X\|$ will denote the quantity

$$\|X\| = \max_i |X_i|, \tag{3.15}$$

and for an $n \times n$ interval matrix $M = (M_{ij})$, let

$$\|M\| = \max_i \sum_{j=1}^{n} |M_{ij}|, \tag{3.16}$$

analogously to the ∞-norm in $\mathbf{R}^n$ [8]. If the IHESSIAN value of a function $f : \mathbf{R}^n \to \mathbf{R}$ over $X \in \mathbf{IR}^n$ is denoted by $\Phi(X) = (F(X), F'(X), F''(X))$, then the existence of $F'(X)$ implies that f is Lipschitz continuous on X, and $L = \|F'(X)\|$ is a Lipschitz constant for f on X. Similarly, the existence of $F''(X)$ implies that ∇f is Lipschitz continuous on X, and $\|F''(X)\|$ is a Lipschitz constant for ∇f on X. Thus, for the function (2.14), it follows from (3.11) that

$$|f(x) - f(y)| \leq 307.6 \cdot \|x - y\|_\infty, \tag{3.17}$$

0.5) $$F_2'(X^*) = \begin{pmatrix} [-4.802 \times 10^{-9}, 1.242 \times 10^{-8}] \\ [-6.200 \times 10^{-9}, 2.400 \times 10^{-9}] \end{pmatrix},$$

d

0.6)

$$''_2(X^*) = \begin{pmatrix} [801.999999987, 802.000000030] & [-400.000000004, -399.999999998] \\ [-400.000000004, -399.999999998] & 200 \end{pmatrix}.$$

he midpoint of X^* was calculated to be $\bar{x} = (1,1)$, with $F_2(\bar{x}) = 0$, $F_2'(\bar{x}) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, and e Hessian matrix $F''_2(\bar{x}) = \begin{pmatrix} 802 & -400 \\ -400 & 200 \end{pmatrix}$, which are validated to be the exact values x^*, $f_2(x^*)$, $\nabla f_2(x^*)$, and $Hf_2(x^*)$ by the above.

The results in three dimensions were completely similar, with the midpoint $\bar{x} = (1,1,1)$ X^* giving the exact values $f_3(\bar{x}) = 0$, $\nabla f_3(\bar{x}) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, and

0.7) $$Hf_3(\bar{x}) = \begin{pmatrix} 802 & -400 & 0 \\ -400 & 1002 & -400 \\ 0 & -400 & 200 \end{pmatrix}.$$

The three-humped camel function g(x) given by (10.2) has five critical points:

0.8) $$x^* = (0,0),$$

hich is its global minimum point,

0.9) $$\pm y^* = \pm\left(\sqrt{2.1 - \sqrt{0.865}}, \frac{1}{2}\sqrt{2.1 - \sqrt{0.865}}\right),$$

hich are saddle points, and

0.10) $$\pm z^* = \pm\left(\sqrt{2.1 + \sqrt{0.865}}, \frac{1}{2}\sqrt{2.1 + \sqrt{0.865}}\right),$$

hich are relative minimum points. One has

0.11) $$0 = g(x^*) < g(-z^*) = g(z^*) < g(-y^*) = g(y^*).$$

The search for all critical points was conducted in the inital region

0.12) $$X_0 = ([-2.0, 1.8], [-0.9, 1.0]),$$

'he modification of the program to search for a global critical minimum gave the ing results:

	Case 1	Case 2	Case 3	Case 4
Transformations	92	140	346	219
Rejected	75	110	169	172
Critical Points	1	1	1	1
Transformations to Locate	83	103	330	201

: results show a considerable improvement over the search for all critical points. Once obal minimum value has been found, remaining regions are generally rejected quickly. ılgorithm was very effective for the largest problem considered, Case 4 above.

The required increase in minimum function values in the search for a global critical mum forced the algorithm toward the boundary of X_0, where regions were quickly ed. The corresponding results were:

	Case 1	Case 2	Case 3	Case 4
Transformations	9	16	27	37
Rejected	9	16	28	38
Critical Points	0	0	0	0

: last two cases, the final regions were rejected without having to perform a Krawczyk formation.

n two dimensions, the interval iteration to the critical point x^* converged to

$$X^* = ([0.999999999999, 1.00000000001], [0.999999999999, 1.00000000001])$$

$$F_2(X^*) = [0.00, 9.62 \times 10^{-20}],$$

The Rosenbrock function (10.1) has the global minimum $f_n(x^*) = 0$ at the critical point $x^* = e = (1, 1, \ldots, 1)$. It is easy to find x^* by Newton's method, but methods which try to reduce $f_n(x)$ at each step find this function rather difficult, particularly in higher dimensions. Four cases were considered:

1. $n = 2$, $X_0 = \big([0.9, 1.2], [0.8, 1.1]\big)$;
2. $n = 2$, $X_0 = \big([-3.7, 1.4], [-1.6, 3.5]\big)$;
3. $n = 3$, $X_0 = \big([0.9, 1.2], [0.8, 1.1], [0.9, 1.2]\big)$;
4. $n = 3$, $X_0 = \big([-3.7, 1.4], [-1.6, 3.5], [-3.7, 1.4]\big)$.

The value $\epsilon = 0$ was taken in each case, and no exceptions occured in any of the examples given here. Searching for all critical points gave the following results:

	Case 1	Case 2	Case 3	Case 4
Transformations	242	957	553	1976
Rejected	167	685	432	1567
Critical Points	1	1	1	1
Transformations to Locate	128	187	328	1672

The algorithm was very busy in the neighborhood of the critical points $x^* = (1, 1)$ and $x^* = (1, 1, 1)$. The region in which (5.5) holds turned out to be rather small, and nearby regions not containing x^* had to be made very small before they could be rejected with certainty. The increase in area of X_0 by a factor of 289 between cases 1 and 2 increased the number of Krawczyk transformations required by a factor of less than four, while the increase in volume of X_0 between cases 3 and 4 by a factor of 4913 resulted in an even smaller increase in the number of transformations, less than 3.6. Going from two to three dimensions increased the number of transformations required to search the entire initial region by a factor of about two in each case.

0. Numerical Examples

he functions selected for numerical computation were the n-dimensional Rosenbrock func-
on [1],

$$f_n(x) = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2\right], \tag{10.1}$$

nd the "three-humped camel" function [3],

$$g(x) = 2x_1^2 - 1.05x_1^4 + \frac{1}{6}x_1^6 - x_1x_2 + x_2^2. \tag{10.2}$$

The program used was written in Pascal-SC for a microcomputer with a Z80 processor nd the CP/M operating system. This was done to take advantage of support for interval rithmetic, an already written library of operators and functions for type IHESSIAN, and ıe utility procedure LGLI for solving linear systems with interval coefficient matrices and ght sides. On the other hand, the small amount of storage available in this machine (64 ilobytes) limited the values of n for the Rosenbrock function (10.1) to $n = 2, 3$. The ctual machine used was also rather slow, with a 1MHz system clock, giving typical times ›r floating-point interval addition and subtraction of 13.5 milliseconds, multiplication, 7.5 milliseconds, and division, 77.5 milliseconds. Nevertheless, the results given below 'ere obtained in a reasonable amount of time.

The most time-consuming part of the computation is the performance of the Krawczyk ·ansformation $K(X)$ (actually, $\hat{K}(X)$), using the Pascal-SC utility program LGLI to solve ıe system (5.1) with interval coefficient matrix and right side. A count is made of the umber of times this transformation is performed, the number of critical points found, ıe number of regions rejected, and the number of regions (if any) in which exceptions re encountered. The sum of the number of regions rejected (which cannot contain crit-al points), the number of regions in which critical points are found, and the number of ‹ceptional regions gives the total number of subregions examined. The Krawczyk trans-›rmation may be applied to a given region several times before it is accepted as containing critical point, or rejected.

points which are not critical, an alteration has to be made in the above procedure. The value of $m(t)$ is updated only when regions X^* containing critical points x^* are computed by interval iteration. If $\sup F(x^*) < m(t-1)$, then we set $m(t) = \sup F(x^*)$, otherwise $m(t) = m(t-1)$. The rejection criterion $\inf F(Z) > m(t)$ remains unaltered. The algorithm will generally be slower than the one given above in this case, but usually still faster than an exhaustive search for all critical points.

In the same way, a function $M(t)$ giving the lower bound for the global maximum M of f is constructed by setting $M(0) = -\text{MAXREAL}$, and updating by $M(t)$ to be the maximum of $M(t-1)$ and $\inf F(x)\}$, assuming that the global maximum is critical. Intervals are rejected if $\sup F(X) < M(t)$. Otherwise, $M(t)$ is updated only at critical points, as above. The modification of the choice algorithm for bisected intervals is done by reversing inequality signs and interchanging infs with sups in the above.

9. Use of the Algorithm for Validation

In addition to its use for global searching, the algorithm given in §6 can be used to validate solutions to optimization problems given by other algorithms. For example, suppose that $\hat{x}$ is an approximate critical point of f found by Newton's method or some other numerical technique. Then, the initial region X can be taken to be, say

$$X = \hat{x} + \frac{\delta}{2} \cdot e \cdot [-1, 1], \tag{9.1}$$

where $\delta > 0$ and $e = (1, 1, \ldots, 1) \in \mathbf{R}^n$ denotes the vector with all components equal to one. If (5.5) holds for this value of X, then all components of $\hat{x}$ are validated to be accurate to a tolerance of δ. Furthermore, the interval iteration (6.1) will give approximations of possibly increased accuracy for the critical point, as well as bounds the for function, gradient vector, and Hessian matrix values. It should be noted that the interval calculations furnish upper and lower bounds for maximum and minimum values of the function, while some other methods give only one-sided bounds.

Successful termination of the computer program will be accompanied with a list of the number of regions processed (given by the number of times the Krawczyk transformation was performed), the number rejected, and the number of critical points found, and the number output to the exception file. Given all its critical points, the global critical maximum and minimum of the function can be found simply by sorting the function values.

8. Global Critical Extrema

The algorithm of §6 can be speeded up if only the global critical maximum or mininum of f on X is desired. The modification of the algorithm to find the global critical minimum will be described; finding the global critical maximum follows exactly the same pattern.

Suppose first that the global critical minimum of f on X is actually its global minimum on X, that is, $f(x) \geq m = f(x^*)$ for $x \in X$, where x^* is the global critical minimum point. The algorithm will compute a decreasing sequence of upper bounds $m(t)$ for m, and reject subregions Z such that $\inf F(Z) > m(t)$. The modifications of the corresponding steps for the single-processor algorithm are:

1°. Set $m(0) =$ MAXREAL, the largest floating-point number (for example, in Pascal-SC, MAXREAL$=9.99999999999 \times 10^{99}$).

2°. If $\sup F(x) < m(t-1)$, then set $m(t) = \sup F(x)$, otherwise, $m(t) = m(t-1)$.

6°. Reject Z^l if $0 \notin F'(Z^l)$ or $\inf F(Z^l) > m(t)$; reject Z^r if $0 \notin F'(Z^r)$ or $\inf F(Z^r) > m(t)$. If neither Z^l nor Z^r can be rejected, then Z^l is considered to be more promising if $\inf F(Z^l) < \inf F(Z^r)$, and Z^r is stacked, or conversely. In case $\inf F(Z^l) = \inf F(Z^r)$, then Z^r is stacked if $\sup F(Z^r) \geq \sup F(Z^l)$, otherwise, Z^l is stacked.

Considerable savings in computer time have been observed due to the introduction of the additional rejection conditions in 6°. In the case of multiprocessors, an efficient way to share the current value of $m(t)$ is necessary, and the rejection of regions in which function values are too large would be carried out in step 1°.

In case the function f can attain smaller values than m on the boundary ∂X of X at

7. Exceptions

Several exceptions can arise in the execution of the algorithm in §6 which could terminate the computation prematurely, or cause it to run indefinitely. These and the way they are handled will be discussed now, because they may also occur in the search for global critical extrema.

1°. If $F''(x)$ contains a singular or badly conditioned matrix, then the attempt to perform the Krawczyk transformation by solving (5.1) will fail. One solution is to replace $F''(x)$ by some nonsingular matrix, for example, $m(F''(X))$ could work [9]. The implementation used for the examples given below simply outputs X to a file for later examination, with an appropriate message, and then selects the next region to be processed from the stack.

2°. The intersection-bisection process can lead to regions which do not differ from the previous ones, because of outward rounding, or which are so small that total time to explore the entire region is prohibitive. This can happen, in particular, if a critical point lies exactly on a bisection coordinate. For this reason, the user is provided with a parameter ϵ such that if the volume V of the region to be processed satisfies

$$V \leq \epsilon \cdot V_0, \tag{7.1}$$

then the region will be output to a file for later examination, with an appropriate message.

The choice $\epsilon = 0$ is permitted; this allows the processing of smaller and smaller regions until their volume (6.9) underflows to 0 or some coordinate becomes degenerate.

3°. If the storage space allotted to the stack is full, then additional regions will be output to a file.

4°. Numerical exceptions, such as division by zero and overflow, are allowed to terminate the present program. However, they could be used as signals to output the offending region to a file with an appropriate message.

In the case of a single processor, the choice of which interval to stack in step 6°(*iv*) will be modified in the case global critical maxima or minima are sought. If only one critical point is sought, the algorithm is terminated at the end of step 3°. Otherwise, the algorithm can continue until all critical points of f in X are found, and no regions remain on the stack to process. Complete processing of X without finding critical points proves that it contained none.

Because the processes of intersection and bisection can result in subregions of a wide range of sizes, a simple count of the number processed at any given time does not give a good indication of the progress being made by the algorithm. For this reason, it has been found convenient to compute the initial volume

$$V_0 = \prod_{i=1}^{n} w(X_i) \tag{6.9}$$

of the region $X = (X_1, X_2, \ldots, X_n)$ to be searched. The unexplored part of the initial region has volume $V_u(t)$ at time t, where $V(0) = V_0$. $V_u(t)$ can be computed simply as the sum of the volumes of the intervals being processed and those on the stack awaiting processing at time t. $V_u(t)$ is a monotone decreasing function of t, and the algorithm terminates when the stack is empty and $V_u(t) = 0$, if an exhaustive search is desired.

and one takes

$$Z^l = (Z_1, \ldots, Z_{j-1}, [\inf Z_j, m(Z_j)], Z_{j+1}, \ldots, Z_n),$$

(6.8)

$$Z^r = (Z_1, \ldots, Z_{j-1}, [m(Z_j), \sup(Z_j)], Z_{j+1}, \ldots, Z_n).$$

6°. (Single processor) Compute $\Phi(Z^l)$, $\Phi(Z^r)$. The test (4.1) is applied to $F'(Z^l)$ and $F'(Z^r)$. There are four cases:

(i) If $0 \notin F'(Z^l)$ and $0 \notin F'(Z^r)$, then X is rejected;

(ii) If $0 \in F'(Z^l)$ but $0 \notin F'(Z^r)$, then return to step 2° with $X = Z^l$;

(iii) If $0 \notin F'(Z^l)$ but $0 \in F'(Z^r)$, then return to step 2° with $X = Z^r$;

(iv) If $0 \in F'(Z^l)$ and $0 \in F'(Z^r)$, then one of the interval vectors is to be placed on a push-down (last in, first out) stack in storage, while the other replaces X for continued processing at step 2°. In this algorithm, the choice is made by the following heuristic: If $w(F(Z^l)) \leq w(F(Z^r))$, then one takes $X = Z^l$ and stacks Z^r for processing later; otherwise, one takes $X = Z^r$ and stacks Z^l.

The region selected for X is considered to be "more promising" than the one stacked because the variation of a function in the neighborhood of a critical point is asymptotically less than it is elsewhere. The goal is to find critical points as quickly as possible, particularly if only one is desired.

6°. (Multiprocessors) Z^l is sent to another processor following the bisection (6.8). Return to step 1° with the current processor taking $X = Z^r$, while the other takes $X = Z^l$. If no processors happen to be free, then Z^l circulates or is put on a common stack to await the first available processor. If a number of processors are free, it could be expeditious to decompose X into more than two subregions, and then send one to each processor. The choices here will depend to a great extent on the multiprocessor configuration actually used.

2°. Compute $\Phi(x) = (F(x), F'(x), F''(x))$ in IHESSIAN arithmetic. Compute an inclusion $\tilde{K}(X)$ of the Krawczyk transformation of X by (5.1).

3°. If $\tilde{K}(X) \subseteq X$, then the interval iteration

$$X^0 = X, \qquad X^{n+1} = \tilde{K}(X^n) \cap X^n \tag{6.1}$$

is performed until it converges to

$$X^* = X^N \subseteq X^{N+1}, \tag{6.2}$$

in a finite number of steps [12]. The values

$$X^*, \qquad \Phi(X^*) = \big(F(X^*), F'(X^*), F''(X^*)\big), \tag{6.3}$$

are output. The existence of a critical point x^* of f in X is guaranteed, and furthermore the bounds

$$x^* \in X^*, \qquad \big(f(x^*), \nabla f(x^*), Hf(x^*)\big) \in \big(F(X^*), F'(X^*), F''(X^*)\big), \tag{6.4}$$

for x^* and the values of the function f, its gradient vector ∇f, and its Hessian matrix Hf at x^*. These bounds are usually as good as can be obtained by floating-point computation, and $F''(X^*)$ can be used to determine the nature of the critical point x^*, if necessary.

4°. If

$$\tilde{K}(X) \cap X = \emptyset, \tag{6.5}$$

then X is rejected.

5°. In the indeterminate case, the region

$$Z = (Z_1, Z_2, \ldots, Z_n) = \tilde{K}(X) \cap X \tag{6.6}$$

is *bisected* in the direction of its widest component. An index j is determined such that

$$w(Z_j) \geq w(Z_i), \quad i = 1, 2 \ldots, n, \tag{6.7}$$

Furthermore,

$$\tilde{K}(X) \subseteq X \tag{5.5}$$

or

$$\tilde{K}(X) \cap X = \emptyset \tag{5.6}$$

thus imply (4.5) or (4.6), respectively. In this way, existence or nonexistence of a critical point of f in X can be established conclusively by a computation done in floating-point interval arithmetic by a computer, providing it is possible to obtain an inclusion for the solution of the system (5.1). In actual practice, the widths of the components of $F''(x)$ will be small, and good inclusions of Y can be obtained by a process of floating-point approximation followed by interval iterative refinement [15], which will fail only if $F''(x)$ contains a singular or very badly conditioned matrix. Interval linear system solvers of this type are available in Pascal-SC and the ACRITH package.

6. The Basic Algorithm

The algorithm described in this section will find one or all the critical points of a function f in a given initial region X, or show that X contains no critical points, provided no exceptions arise. Exceptions will be discussed in a later section. The computer program implementing this algorithm handles exceptions in such a way that the computation always terminates in a finite number of steps. Validated upper and lower bounds are given for all critical points and values found. The algorithm will be presented first for the case of a single processor, in the way it has actually been implemented. Adaptation to a multiprocessor environment will be discussed at the end of the section.

The basic steps of the algorithm are:

1°. Compute $\Phi(X) = (F(X), F'(X), F''(X))$ in IHESSIAN arithmetic. If $0 \notin F'(X)$, then X is rejected.

make use of the fact that an optimization problem underlies the system of equations being solved, which provides information additional to that inherent in an arbitrary system of nonlinear equations. Furthermore, the algorithms given here differ by bisecting intersected intervals in the inconclusive case, which results in a certain amount of increase in speed.

The use of subregions has the advantage that the tests (4.1) and (4.5)–(4.6) become more sensitive as the size of the region decreases, that is, as $\|w(X)\|_\infty \to 0$. In fact, if x^* is a ***regular*** critical point of f, that is, if $(Hf(x^*))^{-1}$ exists, then (4.5) will hold for sufficiently small X such that $x^* \in X$ if ∇f is Lipschitz continuous [10]. The disadvantages are the extra bookkeeping and storage required for pending subregions. However, these are not overwhelming on modern computers.

5. Implementation of the Krawczyk Transformation

The system of equations (4.3), as stated, has the real coefficient matrix $Hf(x)$, interpreted as a degenerate interval matrix, and an interval right side. In actual computation, instead of solving (4.3), one obtains an inclusion $\tilde{\Xi}$ of the solution Y of the system

$$F''(x)Y = -F'(x) + \{F''(x) - F''(X)\}(X - x), \tag{5.1}$$

which has an interval coefficent matrix and an interval right side. The solution of such a system

$$AY = B \tag{5.2}$$

is defined to be

$$Y = \{y \mid ay = b,\ a \in A,\ b \in B\}, \tag{5.3}$$

where a is a real matrix, and $b, y \in \mathbf{R}^n$, provided all the indicated real systems are solvable. In this case, it follows that $\Xi \subseteq Y \subseteq \tilde{\Xi}$, where Ξ is the solution of (4.3), and thus

$$K(X) = x + \Xi \subseteq x + \tilde{\Xi} = \tilde{K}(X). \tag{5.4}$$

where I denotes here the $n \times n$ identity matrix, and $x = m(X)$ the midpoint vector of the interval region X. The real-valued vectors and matrices in (4.2) are of course interpreted as degenerate interval-valued objects.

In actual practice, $K(X)$ is computed by solving the linear system

$$(Hf(x))\Xi = -\nabla f(x) + \{Hf(x) - F''(X)\}(X - x) \tag{4.3}$$

for Ξ, from which

$$K(X) = x + \Xi. \tag{4.4}$$

Once $K(X)$ has been computed, one of the following alternatives holds. If

$$K(X) \subseteq X, \tag{4.5}$$

then there exists a critical point $x^* \in K(X)$ of f; if

$$K(X) \cap X = \emptyset, \tag{4.6}$$

is empty, then X does *not* contain a critical point of f (another rejection criterion); otherwise, the test is inconclusive [7].

With regard to (4.6), the intersection of interval vectors X, Y is said to be empty if for some i, X_i and Y_i are disjoint intervals. It also follows from (4.2) that if $x^* \in X$ is a critical point of f, then $x^* \in K(X)$. Thus, in the inconclusive case, the region

$$Z = X \cap K(X) \subseteq X \tag{4.7}$$

will also contain any critical points of f which lie in X. This suggests decomposing Z (which may be equal to X) into several subregions, and applying the above tests to each. The resulting algorithms, described in more detail below, are essentially modifications of the one given by Moore and Jones [9] for locating solutions of systems of nonlinear equations in several variables. These algorithms differ from the Moore-Jones method in that they

and

$$\|\nabla f(x) - \nabla f(y)\|_\infty \leq 1890.0 \cdot \|x - y\|_\infty \tag{3.18}$$

for $x, y \in X = ([0.9, 1.2], [0.8, 1.1])$ [14]. The Lipschitz continuity of ∇f will enter into the discussion later.

In actual practice, the arithmetic operators and standard library functions for type IHESSIAN can be programmed once and for all, and stored in a small subroutine library, as has been done in Pascal-SC [13].

4. Tests for Existence or Nonexistence of Critical Points

The key issue in the algorithm described in this paper is to determine if a region in $\mathbf{R}^n$ defined by an interval vector $X \in \mathbf{IR}^n$ contains a critical point of the function $f : \mathbf{R}^n \to \mathbf{R}$ or not. First of all, if

$$0 \notin F'(X), \tag{4.1}$$

then it is impossible that $\nabla f(x) = 0$ for $x \in X$, and X can be *rejected*, since it does not contain a critical point of f [7]. On the other hand, $0 \in F'(X)$ does *not* necessarily mean that X contains a critical point of f, because $F'(X)$ overestimates $\nabla f(X)$ in general. The intersection of all interval vectors containing $f(X)$ is called the *interval hull* of $f(X)$. In several dimensions, the interval hull of $f(X)$ can contain points outside of $f(X)$ in general [8].

In addition to the rejection criterion (4.1), a test which is capable of establishing the existence of a critical point x^* in X is necessary. For this purpose, the test given by Moore [m3] will be used. This test is based on the application of the *Krawczyk transformation* K [5] to X:

$$K(X) = x - \left(Hf(x)\right)^{-1} \nabla f(x) + \left\{I - \left(Hf(x)\right)^{-1} F''(X)\right\}(X - x), \tag{4.2}$$

with $\epsilon = 0$. The total number of Krawczyk transformations required was 82, 48 intervals were rejected, all 5 critical points were found, and there were no exceptions. This function is less of a computational challenge than the Rosenbrock function $f_2(x)$; however, the critical points $\pm y^*$ and $\pm z^*$ tend to shield x^* from straightforward iterative procedures, such as Newton's method. Letting $T(\cdot)$ denote the number of Krawczyk transformations required to locate a given critical point, the results were:

$$
\begin{aligned}
T(x^*) &= 6, \\
T(y^*) &= 21, \\
T(z^*) &= 45, \\
T(-y^*) &= 69, \\
T(-z^*) &= 79.
\end{aligned}
\tag{10.13}
$$

The search for the global minimum of $g(x)$ in X_0 required 60 Krawczyk transformations, 46 intervals were rejected, 2 critical points were located in order of decreasing function value, and there were no exceptions. The critical points found were first $-z^*$ and then the global minimum point x^*, with

$$
\begin{aligned}
T(-z^*) &= 10, \\
T(x^*) &= 57.
\end{aligned}
\tag{10.14}
$$

Obviously, the search took an entirely different path than the exhaustive search (10.13) for all critical points of $g(x)$ in X_0.

The search for the global critical maximum of $g(x)$ was somewhat slower, due to the fact that $g(x)$ attains it maximum at a noncritical point on the boundary ∂X_0 of X_0. Consequently, the function $M(t)$ which gives a lower bound for the critical maximum was updated only at critical points. This computation required 76 Krawczyk transformations, 50 intervals were rejected, and three critical points (x^*, y^*, and $-y^*$) were found in order

of nondecreasing function values. The number of transformations required were:

$$T(x^*) = 6,$$

$$T(y^*) = 21, \tag{10.15}$$

$$T(-y^*) = 64.$$

The critical points $\pm z^*$ were also located, but rejected, because the values of $g(x)$ at these relative minimum points is smaller than at the saddle points $\pm y^*$. Investigation of the nature of critical points is done by finding bounds for the eigenvalues of all symmetric matrices A such that $A \in G''(X^*)$, using the Pascal-SC procedure EIGEN. Denoting the eigenvalues of a 2×2 symmetric matrix A by $\lambda_1(A)$ and $\lambda_2(A)$, then intervals $\Lambda_1(X^*)$ and $\Lambda_2(X^*)$ are computed by EIGEN such that

$$\{\lambda_i(A) \mid A \in G''(X^*)\} \subseteq \Lambda_i(X^*), \qquad i = 1,2. \tag{10.16}$$

The character of the critical point $x^* \in X^*$ can be decided on the basis of these bounds.

The results of the interval iteration to critical points were:

$$\begin{aligned} X^* &= \left([-2.0 \times 10^{-99}, 2.0 \times 10^{-99}], [-2.0 \times 10^{-99}, 2.0 \times 10^{-99}]\right), \\ G(X^*) &= [-2.05 \times 10{-}99, 5.00 \times 10^{-99}], \\ G'(X^*) &= \begin{pmatrix} [-1.52 \times 10^{-98}, 1.52 \times 10^{-98}] \\ [-6.00 \times 10^{-99}, 6.00 \times 10^{-99}] \end{pmatrix}, \\ G''(X^*) &= \begin{pmatrix} [3.99999999999, 4.00000000000] & -1 \\ -1 & 2 \end{pmatrix}. \end{aligned} \tag{10.17}$$

The eigenvalues of $Hg(x^*)$ are contained in the intervals

$$\begin{aligned} \Lambda_1(X^*) &= [4.142135623, 4.142135625], \\ \Lambda_2(X^*) &= [1.58578643759, 1.58578643766], \end{aligned} \tag{10.18}$$

which proves conclusively that the critical point $x^* \in X^*$ is a minimum point, because both eigenvalues of $Hg(x^*) \in G''(X^*)$ must be positive [1]. The midpoint of X^* is $\bar{x} = x^* = (0,0)$, with $G(\bar{x}) = g(x^*) = 0$, $G'(\bar{x}) = \nabla g(x^*) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, and $G''(\bar{x}) = Hg(x^*) = \begin{pmatrix} 4 & -1 \\ -1 & 2 \end{pmatrix}$.

Next,

$$Y^* = \left([1.07054229181, 1.07054229185], [0.535271145904, 0.535271145921]\right),$$

$$\begin{aligned} G(Y^*) &= [0.877361557501, 0.877361558041], \\ G'(Y^*) &= \begin{pmatrix} [-5.81 \times 10{-}10, 5.26 \times 10^{-10}] \\ [-5.00 \times 10^{-11}, 4.00 \times 10^{-11}] \end{pmatrix}, \\ G''(Y^*) &= \begin{pmatrix} [-3.87308929305, -3.87308929080] & -1 \\ -1 & 2 \end{pmatrix}. \end{aligned} \tag{10.19}$$

The eigenvalues of $Hg(y^*)$ are contained in the intervals

$$\begin{aligned} \Lambda_1(Y^*) &= [-4.03868818, -4.03868818], \\ \Lambda_2(Y^*) &= [2.165598876, 2.165598884], \end{aligned} \tag{10.20}$$

so that the critical point $y^* \in Y^*$ is indubitably a saddle point.

The next values obtained were

$$Z^* = \left([1.74755234581, 1.74755234586]\right),$$

$$\begin{aligned} G(Z^*) &= [0.298638440884, 0.298638443572], \\ G'(Z^*) &= \begin{pmatrix} [-2.18 \times 10^{-10}, 3.82 \times 10^{-10}] \\ [-1.00 \times 10^{-11}, 0.00] \end{pmatrix}, \\ G''(Z^*) &= \begin{pmatrix} [1.21530892921, 1.21530892930] & -1 \\ -1 & -2 \end{pmatrix}. \end{aligned} \tag{10.21}$$

The eigenvalues of $Hg(z^*)$ are contained in the intervals

$$\begin{aligned} \Lambda_1 &= [5.33440787, 5.33440793], \\ \Lambda_2 &= [2.681868136, 2.681868143], \end{aligned} \tag{10.22}$$

and thus the critical point z^* is a relative minimum point.

The computed intervals

(10.23)

$$-Y^* = \left([-1.070544229185, -1.07054229181], [-0.535271145923, -0.535271145906]\right)$$

and

(10.24)

$$-Z^* = \left([-1.74755234585, -1.74755234580], [-0.873776172925, -0.873776172902]\right)$$

contain the critical points $-y^*$ and $-z^*$, respectively. The function, gradient, and Hessian values on these intervals do not differ significantly from the corresponding ones for Y^* and Z^*. In particular, the eigenvalues of $Hg(-y^*)$ lie in the intervals (10.20), while the eigenvalues of $Hg(-z^*)$ belong to the intervals (10.22). Thus, $-y^*$ is guaranteed to be a saddle point, and $-z^*$ a relative minimum point of g.

References

1. P. E. Gill, W. Murray and M. H. Wright. Practical Optimization. Academic Press, New York, 1981.

2. E. R. Hansen. Global optimization using interval analysis—the multi-dimensional case. Numer. Math. **34** (1980), 247–270.

3. E. R. Hansen. Global optimization with data perturbations. Comput. & Ops. Res. **11**, no. 2 (1984), 97–104.

4. E. R. Hansen and S. Sengupta. Global constrained optimization using interval analysis. *Interval Mathematics 1980*, ed. by K. Nickel, pp. 25–47. Academic Press, New York, 1980.

5. R. Krawczyk. Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlerschranken. Computing **4** (1969), 187–201.

6. G. P. McCormick. Nonlinear Programming. Wiley, New York, 1983.

7. R. E. Moore. A test for existence of solutions to nonlinear systems. SIAM J. Numer. Anal. **14** (1977), 611–615.

8. R. E. Moore. Methods and Applications of Interval Analysis. SIAM Studies in Applied Mathematics 2, Soc. Ind. Appl. Math., Philadelphia, 1979.

9. R. E. Moore and S. T. Jones. Safe starting regions for iterative methods. SIAM J. Numer. Anal. **14** (1977), 1051–1065.

10. L. B. Rall. A comparison of the existence theorems of Kantorovich and Moore. SIAM J. Numer. Anal. **17**, no. 1 (1980), 148–161.

11. L. B. Rall. Automatic Differentiation: Techniques and Applications. Lecture Notes in Computer Science No. 120. Springer-Verlag. Berlin-Heidelberg-New York, 1981.

12. L. B. Rall. A theory of interval iteration. Proc. Amer. Math. Soc. **86**, no. 4 (1982), 625–631.

13. L. B. Rall. Differentiation and generation of Taylor coefficients in Pascal-SC. *A New Approach to Scientific Computation*, ed. by U. W. Kulisch and W. L. Miranker, pp. 291–309. Academic Press, New York, 1983.

14. L. B. Rall. Mean value and Taylor forms in interval analysis. SIAM J. Math. Anal **14**, no. 2 (1983), 223–238.

15. S. M. Rump. Solving algebraic problems with high accuracy. *A New Approach to Scientific Computation*, ed. by U. W. Kulisch and W. L. Miranker, pp. 51–120. Academic Press, New York, 1983.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 2832	2. GOVT ACCESSION NO. AD-A158 150	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) GLOBAL OPTIMIZATION USING AUTOMATIC DIFFERENTIATION AND INTERVAL ITERATION		5. TYPE OF REPORT & PERIOD COVERED Summary Report - no specific reporting period 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) L. B. Rall		8. CONTRACT OR GRANT NUMBER(s) DAAG29-80-C-0041
9. PERFORMING ORGANIZATION NAME AND ADDRESS Mathematics Research Center, University of Wisconsin 610 Walnut Street Madison, Wisconsin 53706		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Work Unit Number 3 - Numerical Analysis and Scientific Computing
11. CONTROLLING OFFICE NAME AND ADDRESS U. S. Army Research Office P.O. Box 12211 Research Triangle Park, North Carolina 27709		12. REPORT DATE June 1985 13. NUMBER OF PAGES 29
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Global unconstrained optimization, Critical points, Automatic differentiation, Interval iteration, Self-validating computation

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Algorithms are presented which find one or all of the critical points of a smooth function in a rectangular region, or the critical points at which the function has maximum or minimum value. If no critical points of the function exist in the given region, then the algorithm verifies this fact. The computation is self-validating, in that the existence or nonexistence of critical

(cont.)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ABSTRACT (cont.)

points is established conclusively, and guaranteed upper and lower bounds are computed for all quantities of interest, including the values of the gradient vector and Hessian matrix of the function. The algorithm makes use of an existing implementation of automatic differentiation and interval computation. Numerical examples are given.

END

FILMED

10-85

DTIC