Computer Science Journal of Moldova, vol.**3**, no.2(8), 1995

Global optimization in one-dimensional case using analytically defined derivatives of objective function

A.Shpak

Abstract

A number of algorithms with simple theoretical base (accessible even for non-specialists) for a wide class of global one-dimensional optimization problems is described below. Good rate of convergence is demonstrated with a lot of numerical examples.

1 Introduction

The following problem will be considered:

determine (numerically) f^* and x^* such that $x^* \in [a,b]$ and for all $x \in [a,b] \quad f^* \le f(x), \quad f(x^*) - f^* \le \varepsilon$, where f is a given bounded real-valued function of one variable, $[a,b]$ is a given interval, and ε is a given positive number (accuracy).

If f has only one local minimum point on $[a,b]$ then we can use one of known algorithms (such as Golden Section Algorithm) to solve this problem. But if f has more then one local minima on $[a,b]$, or number of local minimum points is unknown, then we have a global optimization problem, and it is not so simple. In spite of abundance of literature dedicated to global optimization problems (see survey [1]), now we have not a simple and reliable way to solve them. So, the normal reaction of a non-specialist in global optimization when he meets such a problem is usually one of listed below:

– he tries so-called "grid method": takes a great number (10^6 say) of points on $[a,b]$ (i.e. a grid on the given interval), and finds a minimum point over this grid. This way is good only if an objective function is simple enough and require not much time to compute it;

– he tries an algorithm taken from some book, but (being non-specialist) he cannot decide — is his objective function of class described there, or how to pick out a lot of parameters required to run this algorithm, or why this algorithm converges “almost always” but not for his problem, etc.

Therefore the aim of this paper is to give not a new good theory, but method with the following properties:

- simple mathematical background, clear for non-specialist, no “euristics” and “empirics”;
- guaranteed convergence;
- the class of objective functions is as wide as possible;
- “ready-to-use”: anyone familiar with some programming language can program it without any difficulties;
- good rate of convergence.

Mathematical base is considered in sections 2 and 3, formally described routine is given in section 4, and convergence is discussed in section 5. One can find it to be useful the further discussion in section 6.

2 Methodology

The approach suggested below is based on the well-known branch and bounds method. Our algorithm will work iteratively, and let us suppose that on k-th iteration we have $k+1$ points x_i such that $a = x_0 < x_1 < \ldots, < x_{k-1} < x_k = b$. Then we have to:

1. estimate each interval, i.e. find some appropriate numbers $R_i \le \min_{x\in[x_{i-1},x_i]} f(x), \quad i = 1, 2, \ldots, k;$

2. find interval with the best (minimal) $R^* = \min_{i \in \{1,\dots,k\}} R_i$;
3. if $\min_{i \in \{0,1,\dots,k\}} f(x_i) - R^* \le \varepsilon$ then we got a solution and the algorithm stops;
4. bisect the best interval (found on step 2) by a new point x_{new};
5. add the new point x_{new} to the set $\{x_i\}_0^k$, set $k := k+1$, reindex points to get the order $a = x_0 < x_1 < \dots, < x_{k-1} < x_k = b$ and go to step 1.

So, two things are interesting for us: R_i on step 1 and x_{new} on step 4.

Now it is the time to discuss the class of objective functions, wide enough to cover our practical needs, but limited enough to construct an effective algorithm. It is "almost the fact" in global optimization that the class of bounded (or even bounded and continuous) functions is too wide: unlikely one cannot hope to construct something effective for such functions. The next class (narrowed but still wide) is the class of functions with bounded first derivatives. So, let us suppose that the objective function f has a bounded derivative on $[a,b]$. Using Taylor's formula we have for all $x \in [x_{i-1}, x_i]$

$$f(x) \ge f(x_{i-1}) + t h_i m_i(f'),$$
$$f(x) \ge f(x_i) - (1-t) h_i M_i(f'),$$

where $h_i = x_i - x_{i-1}, \quad t = (x - x_{i-1})/h_i \quad (0 \le t \le 1)$ and all $x \in [x_{i-1}, x_i]$ verify the inequalities

$$m_i(f') \le f'(x) \qquad \text{and} \qquad M_i(f') \ge f'(x).$$

Hence

$$\begin{aligned} f(x) &= (1-t) f(x) + t f(x) \ge \\ &\ge (1-t) f(x_{i-1}) + t f(x_i) + t(1-t) h_i \Big[m_i(f') - M_i(f') \Big] \end{aligned}$$

or, denoting $u_i^{(1)} = h_i \Big[M_i(f') - m_i(f') \Big]$,

$$f(x) \ge (1-t) f(x_{i-1}) + t f(x_i) - t(1-t) u_i^{(1)}. \tag{1}$$

Suppose now that f has bounded second derivative on $[a, b]$ (such class of functions is more limited than the one discussed above, and we may hope to estimate better our function). Using Everett's formula we have

$$f(x) \geq (1-t)f(x_{i-1}) + tf(x_i) - t(1-t)\frac{h_i^2}{2}M_i(f'').$$

where $h_i = x_i - x_{i-1}$, $\quad t = (x - x_{i-1})/h_i \quad (0 \leq t \leq 1)$ and for all $x \in [x_{i-1}, x_i]$ the inequality $M_i(f'') \geq f''(x)$ holds. Denoting now $u_i^{(2)} = \frac{h_i^2}{2}M_i(f'')$ one gets the formula

$$f(x) \geq (1-t)f(x_{i-1}) + tf(x_i) - t(1-t)u_i^{(2)}. \tag{2}$$

Comparing (1) and (2), we get the expression

$$(1-t)f(x_{i-1}) + tf(x_i) - t(1-t)u_i \tag{3}$$

with $u_i = u_i^{(1)}$ or (if possible) $u_i = u_i^{(2)}$.

Now, we can get required R_i as a value of minimum point of (3) under the constraint $0 \leq t \leq 1$. By means of simple transformations we have

$$R_i = \begin{cases} \min\{f(x_{i-1}), f(x_i)\}, \\ \qquad\qquad \text{if } u_i \leq |f(x_{i-1}) - f(x_i)|; \\ \frac{1}{2}\Big[f(x_{i-1}) + f(x_i)\Big] - \frac{1}{4}u_i - \frac{1}{4u_i}\Big[f(x_{i-1}) - f(x_i)\Big]^2, \\ \qquad\qquad \text{if } u_i > |f(x_{i-1}) - f(x_i)|. \end{cases} \tag{4}$$

and the expression for minimum point

$$t_{min} = \frac{1}{2}\Big[1 + \frac{f(x_{i-1}) - f(x_i)}{u_i}\Big] \quad \text{if } u_i > |f(x_{i-1}) - f(x_i)|. \tag{5}$$

Note that if $u_i \leq |f(x_{i-1}) - f(x_i)|$ then (3) has minima either in $t = 0$ or in $t = 1$. It is obvious that in this case i-th interval is not interesting for further investigation.

Let us now discuss the way to get x_{new}. It is quite natural to use (5) for that. Hence, taking into account that $t = (x - x_{i-1})/h_i$, we have

$$x_{new} = \frac{1}{2}\Big[x_{j-1} + x_j + h_j \frac{f(x_{j-1}) - f(x_j)}{u_j}\Big], \tag{6}$$

where j is such that $R_j = \min_{i\in 1,\dots,k} R_i$. Thus the only question is: how can one calculate u_i? Let us discuss it below.

3 Computing bounds with interval analysis

The problem we need to solve now is:

$\varphi(x)$ is a given bounded analytically defined function on $[x_-, x_+]$ (we are interested in $\varphi(x) = f'(x)$ or $\varphi(x) = f''(x)$). One has to find numbers φ_- and φ_+ such that $\varphi_- \le \varphi(x) \le \varphi_+$ for all $x \in [x_-, x_+]$.

"Analytically defined function" means here that we can apply the interval analysis to it. One familiar with this technique can go to the next section. There is a plenty of literature about interval analysis. We used [2] as a basis for this section.

Let $I(X)$ denotes a set of all intervals contained in an interval X, i.e.

$$I(X) = \Big\{[x_-, x_+] \,|\, x_- \in X,\ x_+ \in X,\ x_- \le x_+\Big\}.$$

Definition. *The function $\Phi : I(X) \to I(I\!R)$ is an inclusion function for $\varphi : X \to I\!R$ if $\Phi(Y) \supseteq \Big\{\varphi(x) \,|\, x \in Y\Big\}$ for all $Y \in I(X)$.*

Let us assume that some basic set (BS) of functions is available including four arithmetic operations and "standard" functions such as $\sin x,\ \cos x, \dots, \log x,\ \exp x,\ \sqrt{x},\ x^n$, etc. (one can fill up this list with any standard functions available on a computer).

The interval arithmetic operations in $I(I\!R)$ are defined by

$$A \star B = \{a \star b \mid a \in A,\ b \in B\} \quad \text{for} \quad A, B \in I(I\!R),$$

where "$\star$" denotes one of the operations "+", "–", "·" and "/" (note that A/B is not defined if $0 \in B$). This definition is equivalent to the following rules

$$\begin{aligned}
[a,b]+[c,d] &= [a+c, b+d];\\
[a,b]-[c,d] &= [a-d, b-c];\\
[a,b]\,[c,d] &= [\min\{ac,ad,bc,bd\}, \max\{ac,ad,bc,bd\}];\\
[a,b]\,/\,[c,d] &= [a,b]\,[1/d, 1/c] \quad \text{(if defined)}.
\end{aligned}$$

Interval $[a,a]$ is equivalent to "explicit" a, thus the operations $a+[b,c]$, $a-[b,c]$ etc. are defined.

It is obvious that interval arithmetic gives inclusion functions for "usual" arithmetic operations. There is no problem to calculate inclusion functions for other representatives of BS. We give below a library of corresponding routines in PASCAL-like notation for interested readers (routines used for numerical experiments are given).

- $[a,b]^2$ — inclusion function for x^2:

```
IF     $a > 0$   THEN  $c := a^2$;   $d := b^2$;
ELSIF  $b < 0$   THEN  $c := b^2$;   $d := a^2$;
ELSE                   $c := 0$;     $d := max(a^2, b^2)$;
END;             RETURN $[c,d]$;
```

- $[a,b]^3$ — inclusion function for x^3:
  RETURN $[a^3, b^3]$;

- Iexp$([a,b])$ — inclusion function for $\exp(x)$:
  RETURN $[\exp(a), \exp(b)]$;

- CONST $TwoPi = 2\pi$ — constant often used in trigonometry;

- $ChkPi(a,b,c)$ — service procedure for trigonometry, returns BOOLEAN=TRUE when there is no such integer k that $a \leq 2\pi(k+c) \leq b$ (supposing that $a \leq b$, $b-a < TwoPi$); TRUNC means integer part of nonnegative REAL:

```
$a1 := a/TwoPi - c$;     $b1 := b/TwoPi - c$;
IF $(b1 < 0)$ OR $(a1 > 0)$ THEN
```

```
        RETURN TRUNC(Mod(ABS(a1),2))=
                           TRUNC(Mod(ABS(b1),2));
ELSE    RETURN FALSE;
END;
```

- Isin($[a, b]$) — inclusion function for sin(x):

```
$c := -1$;     $d := 1$;
IF $(b - a) < TwoPi$ THEN
    $Smin := \min(\sin(a), \sin(b))$;  $Smax := \max(\sin(a), \sin(b))$;
    IF $ChkPi(a, b, -0.25)$ THEN $c := Smin$ END;
    IF $ChkPi(a, b, 0.25)$   THEN $d := Smax$ END;
END;                 RETURN $[c, d]$;
```

- Icos($[a, b]$) — inclusion function for cos(x):

```
$c := -1$;     $d := 1$;
IF $(b - a) < TwoPi$ THEN
    $Cmin := \min(\cos(a), \cos(b))$;  $Cmax := \max(\cos(a), \cos(b))$;
    IF $ChkPi(a, b, 0.5)$ THEN $c := Cmin$ END;
    IF $ChkPi(a, b, 0.0)$ THEN $d := Cmax$ END;
END;                 RETURN $[c, d]$;
```

Now, let RC be the set of functions which can be constructed recursively by composition from BS in finitely many steps. Then any function φ of RC can be represented as a finite expression consisting of the basic functions of BS. For instance,

$$\varphi(x) = \cos(x) + \frac{10}{3}\cos(\frac{10}{3}x) + \frac{1}{x} - 0.84$$

could be such an expression. Each function φ of RC has then an inclusion function Φ. The only thing one has to do to get Φ is to replace each occurrence of the variable x by the "interval"-variable $[x_-, x_+]$ and each occurrence of a function $g \in BS$ by the inclusion function G of g in an expression of φ. The resulting function is then an inclusion function of φ [2]. The inclusion function for $\varphi(x)$ mentioned above is

$$\Phi([x_-, x_+]) = \mathrm{Icos}([x_-, x_+]) + \frac{10}{3}\mathrm{Icos}(\tfrac{10}{3}[x_-, x_+]) + \frac{1}{[x_-, x_+]} - 0.84.$$

There is not a big problem also in a case when searching for inclusion function Φ for φ if φ is not explicitly given, for example, if φ is defined via a numerical algorithm. In such cases it is only necessary to find inclusions for explicitly given parts of φ and then to apply the algorithm to them.

It is one of "annoying" features of interval analysis that inclusion functions received via process described above depend on the chosen function expression for φ. Particularly, there is no distributive law in interval arithmetic ($Y_1(Y_2+Y_3) \subseteq Y_1Y_2+Y_1Y_3$), so some transformations of φ can lead to improving of quality of corresponding function Φ.

4 Routine

Here is described a formal routine for our method. All real numbers are refered as REAL, but it is recommended to use LONGREAL data type if possible. If one is about to program this routine as procedure on some high level programming language he have to provide following parameters and subroutines:

$BndL$, $BndR$	— real numbers, bounds of initial interval;
$Func(x)$	— real procedure-function, returns value of objective function in point x;
$Method$	— integer, $Method=1$ if first derivative is used, $Method=2$ otherwise;
$DrvBnd(a,b,m,M)$	— procedure for calculation inclusions $[m,M]$ of first or second (depending on "Method" parameter) derivatives of objective function on given interval $[a,b]$; read a previous section to program it;
$MaxK$	— integer constant, maximum permissible number of iterations ($MaxK=1000$ seems to be enough);
$eps1$	— required accuracy, must be a positive real;

$eps2$	— a real number; routine stops when size of the best interval becomes less then $eps2$ to avoid numerical errors; if one is not afraid of numerical errors he can set $eps2 < 0$.

Results of routine's work will be saved in the following output data:

$Xmin$, $Fmin$	— real numbers: the found minimum point and corresponding value of the objective function;
K	— integer number of made iterations;
$ExCode$	— integer exit code: $ExCode = 0$ means OK (minimum point found with a required accuracy), $ExCode = 1$ means that procedure stops on "$eps2$" condition, $ExCode = 2$ — maximum number of iterations exceeded.

The following arrays are used in procedure:

X [0..MaxK]	— searching points $x_0, x_1, \ldots, x_K$;
F [0..MaxK]	— objective function values $f(x_0), f(x_1), \ldots, f(x_K)$;
R [1..MaxK]	— estimates of intervals $R_1, R_2, \ldots, R_K$;
Xp[1..MaxK]	— for storing "possible" values for Xnew; there is some economy of calculation time if to calculate possible value of Xnew on i-th interval together with R_i.

Service procedure **CalcR(i)** is used to calculate **R[i]** and **Xp[i]** where input parameter **i** is the number of interval:

```
PROCEDURE CalcR(i)

  h := X[i]-X[i-1];                (* h = length of i-th interval *)

  DrvBnd(X[i-1],X[i], Dm,DM);      (* call user proc. to calculate *)
                                   (* inclusions for derivatives *)
  IF Method = 1                     (∗ supposing that Method=1 or 2 ∗)
      THEN Uh := DM - Dm;           (∗ Uh is tmp variable = $u_i/h$ ∗)
  ELSE Uh := 0.5 ∗ h ∗ DM;
  END;

  Fd := F[i-1] - F[i];

  Xp[i] := 0.5∗(X[i-1] + X[i] + Fd/Uh);    (∗ “possible” Xnew, ∗)
                                           (∗ see (6) ∗)
  U := Uh∗h;                               (∗     $u_i$     ∗)

  IF U > ABS(Fd) THEN
    R[i] := 0.5∗(F[i-1]+F[i]) - 0.25∗U - 0.25∗Fd∗Fd/U;
  ELSE R[i] := min(F[i-1], F[i]);          (∗ see (4) ∗)
  END;

END CalcR;
```

Now, taking into account the notation and assumptions adduced above, let us write our routine.

```
(∗ Step 1. Initialization ∗)

X[0] := BndL; X[1] := BndR;          (∗ set initial interval ∗)
F[0] := Func(BndL);
F[1] := Func(BndR);

CalcR(1);                            (∗ get Xp[1] ∗)

X[2] := X[1]; F[2] := F[1];          (∗ subdivide it here to ∗)
X[1] := Xp[1]; F[1] := Func(X[1]);   (∗ avoid problems with ∗)
K := 2;                              (∗ “empty loops” when ∗)
                                     (∗ K = 1 ∗)
Fmin:=F[0]; Xmin:=X[0];              (∗ calculate initial ∗)
FOR i:= 1 TO K DO                    (∗ values for Xmin, ∗)
    IF F[i] < Fmin THEN              (∗ Fmin ∗)
        Fmin:=F[i]; Xmin:=X[i];
    END;
END;

CalcR(1); CalcR(2);                  (∗ estimate intervals ∗)
```

```
LOOP                       (* iterate *)
 (* Step 2. Find the “best” interval *)

  Rmin := R[1]; Imin:=1;
  FOR i:= 1 TO K DO
        IF R[i] < Rmin THEN
             Rmin := R[i]; Imin := i;
        END;
  END;

(* Step 3. Check stopping conditions *)

  IF (Fmin - Rmin) < eps1        THEN ExCode:=0; EXIT END;
  IF (X[Imin]-X[Imin-1]) < eps2 THEN ExCode:=1; EXIT END;
  IF K >= MaxK                   THEN ExCode:=2; EXIT END;


(* Step 4. Find and insert new point *)

  Xnew := Xp[Imin]; Fnew := Func(Xnew);

  K := K+1;

  FOR i := K TO Imin+1 BY -1 DO
      X[i] := X[i-1];
      F[i] := F[i-1];
      R[i] := R[i-1];
      Xp[i] := Xp[i-1];
  END;


  X[Imin] := Xnew;
  F[Imin] := Fnew;

  CalcR(Imin);              (* estimate new intervals *)
  CalcR(Imin+1);

  IF Fnew < Fmin THEN  (* get new Xmin, Fmin *)
      Fmin:= Fnew; Xmin:= Xnew;
  END;

END;                   (* END of LOOP = GOTO Step 2. *)
```

5 Numerical examples and convergence rate

We will not give a proof of convergence of described algorithm for some reasons. Firstly, it is almost evident. Secondly, an interested reader can find in [2] a proof of convergence for more general algorithm. Thirdly, one can construct a convergent algorithm for solving ANY optimization problem (see the joke [3]). A more important question is the rate of convergence. There is too few algorithms (and corresponding classes of problems) in global optimization which allow to prove theoretical properties of convergence rate. So, the common practice is testing of algorithms on a wide class of numerical examples to get some empirical dependencies. Of course, testing result is not a proof, but it is more than nothing. The set of test functions used below is taken from [4, 5]:

1) $f_1(x) = \sin(x) + \sin(\frac{10}{3}x) + \ln x - 0.84x + 3, \quad 2.7 \le x \le 7.5;$

2) $f_2(x) = \sin(x) + \sin(\frac{2}{3}x), \quad 3.1 \le x \le 20.4;$

3) $f_3(x) = -\sum_{i=1}^{5} i \sin\left[(i+1)x + i\right], \quad -10 \le x \le 10;$

4) $f_4(x) = (x + \sin x)\exp(-x^2), \quad -10 \le x \le 10;$

5, 6) $f_{5,6}(x) = -\sum_{i=1}^{10} \frac{1}{k_i^2(x-a_i)^2 + c_i}, \quad 0 \le x \le 10;$

where k_i, a_i, c_i are constants, defined as

$$a=(3.040, 1.098, 0.674, 3.537, 6.173, 8.679, 4.503, 3.328, 6.937, 0.700),$$
$$k=(2.983, 2.378, 2.439, 1.168, 2.406, 1.236, 2.868, 1.378, 2.348, 2.268),$$
$$c=(0.192, 0.140, 0.127, 0.132, 0.125, 0.189, 0.187, 0.171, 0.188, 0.176)$$

for the function f_5, and

$$a=(4.696, 4.885, 0.800, 4.986, 3.901, 2.395, 0.945, 8.371, 6.181, 5.713),$$
$$k=(2.871, 2.328, 1.111, 1.263, 2.399, 2.629, 2.853, 2.344, 2.592, 2.929),$$
$$c=(0.149, 0.166, 0.175, 0.183, 0.128, 0.117, 0.115, 0.148, 0.188, 0.198)$$

for the function f_6.

First and second problems are simple, with only a few local minimum points. Problems 3) and 4) are rather complicated. Function f_3 has 20 local minimum points (3 of them are global). Function f_4 varies slowly almost on whole interval, having sharp pikes of global maxima and minima. Functions f_5 and f_6 (known as Shekel's functions) have a number of sharp local minimum points with relatively flat maximums. Values and coordinates of global minimum points are listed in table 1.

Table 1: Solutions for the test problems.

Function	Solution	Value
f_1	5.19977837	-1.60130755
f_2	17.03919896	-1.90596112
f_3	-6.77457615	-12.03124944
	-0.49139083	-12.03124944
	5.79179447	-12.03124944
f_4	-0.67957866	-0.82423940
f_5	0.68586093	-14.59265203
f_6	4.85556557	-13.92234488

All the functions $f_1, \ldots, f_6$ have bounded second derivatives and hence we can use both variants of our algorithm (i.e. we can provide inclusions both for first and second derivatives and call our routine first time with the parameter Method=1, and then with Method=2). Test results (number of iterations required to achieve the needed accuracy) are given in table 2 and table 3. In all the tests there was set $eps1 = eps$ and $eps2 < 0$ (see routine parameter in section 4), and all the problems were solved with given accuracy.

Of course, one can say that six problems are only six problems. The common practice is to use a class of test functions with pseudorandom coefficients to get more representative results. There was used the class of problems

$$f_7(x) = a_0 + \sum_{j=1}^{N} \Big(a_j \sin \frac{\pi j}{2} x + b_j \cos \frac{\pi j}{2} x \Big), \quad x \in [0, 1],$$

where a_i and b_i are uniformly distributed on interval $[-1, 1]$ pseudo-random numbers, and N is uniformly distributed on interval $[4, 14]$

Table 2: Method = 1 (with first derivatives).

ε	f_1	f_2	f_3	f_4	f_5	f_6
10^{-3}	11	13	82	17	45	64
10^{-4}	12	14	89	18	46	69
10^{-5}	16	16	91	21	50	73
10^{-6}	17	18	96	22	52	80
10^{-7}	19	19	104	24	53	84
10^{-8}	21	20	111	26	55	90
10^{-9}	22	22	115	27	56	93
10^{-10}	24	24	119	29	58	99
10^{-11}	26	25	126	30	59	102
10^{-12}	28	27	133	33	62	109

Table 3: Method = 2 (with second derivatives).

ε	f_1	f_2	f_3	f_4	f_5	f_6
10^{-3}	9	10	71	16	27	33
10^{-4}	9	10	72	16	27	34
10^{-5}	9	11	72	16	27	34
10^{-6}	9	11	74	18	28	35
10^{-7}	9	11	74	19	29	35
10^{-8}	9	12	75	19	29	36
10^{-9}	11	12	77	19	29	36
10^{-10}	12	12	78	19	29	36
10^{-11}	12	13	79	19	29	36
10^{-12}	12	13	79	20	29	37

pseudorandom integer number. Another class of test problems is a set of Shekel's functions

$$f_8(x) = -\sum_{i=1}^{10} \frac{1}{k_i^2(x-a_i)^2 + c_i}, \qquad x \in [0, 10],$$

where coefficients are uniformly distributed pseudorandom numbers, $a_i \in [0, 10]$, $k_i \in [1, 3]$, and $c_i \in [0.1, 0.3]$. There were used 1000 functions of each class. The average results may be found in Table 4.

Table 4: Results for test functions f_7 and f_8.

	Method=1		Method=2	
ε	f_7	f_8	f_7	f_8
10^{-3}	17.88	44.58	10.29	26.19
10^{-4}	20.48	47.25	10.90	26.76
10^{-5}	22.94	49.91	11.39	27.29
10^{-6}	25.41	52.63	11.81	27.74
10^{-7}	27.86	55.25	12.16	28.11
10^{-8}	30.33	57.94	12.47	28.43
10^{-9}	32.83	60.64	12.78	28.73
10^{-10}	35.32	63.28	13.03	29.00
10^{-11}	37.82	65.99	13.24	29.25
10^{-12}	40.30	68.68	13.46	29.49

Obviously, one can suppose that for examined classes of test functions proposed algorithm converges linearly on the average when first derivatives are used, and superlinearly on the average with second derivatives. There are some hints about such behaviour of this algorithm in [2].

6 Further discussion

Usually in global optimization an unconstrained problem is considered. The reason is that we can reduce an optimization problem with constraints to an unconstrained one by penalty functions or something else. But some useful properties of an initial problem (for example differentiability) may be lost. One can extend our methodology to solve problems with constraints without any “reductions”. The only thing one have to do is to estimate each constraint by corresponding expression (3) and then to solve a simple problem with a quadratic objective function and quadratic constraints. Further, the found solution may be used as an estimate of a corresponding subinterval, and our algorithm may work without any other changes. We will not discuss this subject widely here because it will be a topic of an another paper.

There are a lot of algorithms similar to those discussed above (see for instance [6]. But we found that the algorithms presented in this paper have two advantages: the simplicity and the rate of convergence.

References

[1] Zhiglhiavsky A.A., Zhilinskas A.G. Methods of global optimization. Moscow, Nauka, 1991. (Russian)

[2] Ratschek H. Inclusion functions and global optimization. Mathem.Programming, v.33(1985), no.3, 300–317.

[3] Anonymous. A new algorithm for optimization. Mathem. Programming, v.3(1972), no.1, 124–128.

[4] Zhilinskas A.G. Global optimization. Axiomatics of statistical models, algorithms and their application. Vilnius, Mokslas Publischers, 1986. (Russian)

[5] Strongin R.G. Numerical methods in multiextremal problems. Moscow, Nauka, 1978. (Russian)

[6] Shpak A.L. An algorithm for minimization of multiextremal twice differentiable functions on a interval. Math.Res., no 110. Chişinău, Ştiinţa, 1989.

A.Shpak
Institute of Mathematics,
Academy of Sciences of Moldova,
5 Academiei str.,
Kishinev, 277028, Moldova
e-mail: 33*verlan@mathem.moldova.su*

Received 27 October, 1995