Commun. Optim. Theory 2023 (2023) 21 https://doi.org/10.23952/cot.2023.21

MATHRES Communications in Optimization Theory
Available online at http://cot.mathres.org

THE ELLIPSOID METHOD AND COMPUTATIONAL ASPECTS

ANDREAS FISCHER[1], OLGA KHOMYAK[2], PETRO STETSYUK[2,*]

[1]Faculty of Mathematics, Technische Universität Dresden, Dresden, Germany
[2]V. M. Glushkov Institute of Cybernetics, National Academy of Sciences of Ukraine, Kyiv, Ukraine

Dedicated to the memory of Professor Naum Z. Shor on the occasion of his 85th birthday

Abstract. This paper gives an overview of particular older and recent results for the ellipsoid method with respect to the contributions by Naum Zuselevich Shor. Therefore, we present this method as a subgradient algorithm with space dilation. For a certain choice of the dilation coefficient, this is a method of outer approximation of semi-ellipsoids by ellipsoids with monotonous decrease in their volume. The paper shows results on properties and applications of the ellipsoid method including computational aspects. Two forms of the ellipsoid method are described which differ in the way of updating the inverse space transformation matrix. The applicability of the ellipsoid method to several problem classes, like convex programs and saddle point problems for convex-concave functions, is discussed. Finally, the acceleration of the ellipsoid method by deeper ellipsoid approximations is also dealt with.

Keywords. Ellipsoid method; Nonsmooth optimization; Subgradient algorithm; Space dilation; Saddle point problem.

2020 Mathematics Subject Classification. 65K05, 90C25, 90C30, 90C90.

1. INTRODUCTION

The classical ellipsoid method was first proposed in 1976 by Yudin and Nemirovski [16]. They derived this method from the cutting plane scheme and called it modified centered cutting method. Independently, the ellipsoid method was discovered by Shor in the paper [7] from 1977. There, the method is presented as a particular case of subgradient methods with space dilation, which were proposed by Shor at the end of the sixties [5, 6]. Based on the framework of methods with space dilation, a parametrized version of the ellipsoid method is able to unify existing variants of the ellipsoid method, namely those by Shor [7], Nemirovski and Yudin [3, page 76], and Khachiyan [2].

*Corresponding Author.
E-mail address: andreas.fischer@tu-dresden.de (A. Fischer), khomiak.olha@gmail.com (O. Khomyak), stetsyukp@gmail.com (P. Stetsyuk).

Received: November 14, 2022; Accepted: February 24, 2023.

In this paper, we provide properties of two algorithmic realizations of the ellipsoid method. The first algorithm is based on updating a possibly nonsymmetric matrix B, as in the ellipsoid method suggested by Shor, and the second updates a symmetric matrix $H = BB^\top$, as proposed by Skokov [10] for subgradient methods with space dilation.

Furthermore, we present the Algorithm **emshor** (**e**llipsoid **m**ethod of **Shor**) for the unconstrained minimization of a convex function. In particular, it is able to generate a point at which the function value does not deviate more than a specified tolerance from the optimal function value. The algorithm was successfully applied to the minimization of convex ravine functions.

We describe how the ellipsoid method can be applied to the solution of constrained convex programs and to determining a saddle point of a convex-concave function. It is also shown that one can use deeper ellipsoid approximations, i.e., minimal volume ellipsoids based on two cutting hyperplanes. To this end, an anti-ravine technique, similar to that in Shor's r-algorithm [8, 9], is considered.

The material is presented as follows. In Section 2, the B- and the H-form of the ellipsoid method and their properties are described. Thereafter, Section 3 shows Algorithm **emshor** together with an Octave implementation and numerical results for a smooth and a non-smooth convex ravine function. In addition, the parametrized version of the ellipsoid method is discussed in Section 3.3. Then, in Section 4, we demonstrate how the ellipsoid method can be applied to constrained convex programs and to saddle point problems of a convex-concave function. The accelerating the ellipsoid method by deeper ellipsoid approximations is discussed in Section 5.

2. The ellipsoid method and its properties

Let a mapping $g : \mathbb{R}^n \to \mathbb{R}^n$ be given. We assume that $x^* \in \mathbb{R}^n$ exists so that $g(x)^\top (x - x^*) \geq 0$ for all $x \in \mathbb{R}^n$ and $g(x) \neq 0$ for all $x \neq x^*$. The ellipsoid method can be used to approximately determine x^*. A prominent application in this setting is the minimization of a convex function $f : \mathbb{R}^n \to \mathbb{R}$ with g being the subgradient of f. This case is dealt with in Section 3.

2.1. The B-form of the ellipsoid method.

This particular way to describe the ellipsoid method is used in the following algorithm.

Algorithm 1 – The B-form of the ellipsoid method

Step 0. Choose $x_0 \in \mathbb{R}^n$, a non-singular matrix $B_0 \in \mathbb{R}^{n\times n}$, and $r_0 > 0$ so that

$$\left\| B_0^{-1}(x_0 - x^*) \right\| \leq r_0.$$

Moreover, set $\alpha := \sqrt{\frac{n+1}{n-1}}$ and $k := 0$.

Step 1. If $g(x_k) = 0$, then set $x^* := x_k$ and STOP.

Step 2. Calculate

$$x_{k+1} := x_k - h_k B_k \xi_k, \quad \text{where} \quad \xi_k := \frac{B_k^\top g(x_k)}{\left\| B_k^\top g(x_k) \right\|}, \quad h_k := \frac{1}{2}\left(1 - \frac{1}{\alpha^2}\right) r_k.$$

Step 3. Update

$$B_{k+1} := B_k + \left(\frac{1}{\alpha} - 1\right)(B_k \xi_k)\, \xi_k^\top \quad \text{and} \quad r_{k+1} := \frac{1}{2}\left(\alpha + \frac{1}{\alpha}\right) r_k.$$

Step 4. Set $k := k+1$ and go to Step 1.

In Algorithms 2 – 4 later on, the appearance of α will be equivalently replaced by means of

$$\frac{1}{2}\left(1-\frac{1}{\alpha^2}\right)=\frac{1}{n+1},\quad \frac{1}{\alpha}-1=\sqrt{\frac{n-1}{n+1}}-1,\quad \text{and}\quad \frac{1}{2}\left(\alpha+\frac{1}{\alpha}\right)=\frac{n}{\sqrt{n^2-1}}. \tag{2.1}$$

Throughout, for any x_k, x_{k+1} generated by Algorithm 1, let the ellipsoids

$$\mathscr{E}_k := \{x \mid \|B_k^{-1}(x_k - x)\| \le r_k\}$$

be defined. Moreover, let $vol(\mathscr{E})$ denote the volume of the ellipsoid $\mathscr{E}$.

Theorem 2.1 (Theorem 1 in [12]). *Let x_k and x_{k+1} be generated by Algorithm 1. Then, the ratio of volumes of the ellipsoids $\mathscr{E}_{k+1}$ and $\mathscr{E}_k$ does not depend on k and is equal to*

$$q_n(\alpha) := \frac{vol(\mathscr{E}_{k+1})}{vol(\mathscr{E}_k)} = \frac{1}{\alpha}\left(\frac{1}{2}\left(\alpha+\frac{1}{\alpha}\right)\right)^n < 1.$$

Moreover, $x^* \in \mathscr{E}_k$ holds for all $k \in \mathbb{N}$.

Theorem 2.2 (Theorem 3.14 in [8]). *Let the sequence $\{x_k\}$ be generated by Algorithm 1. Then,*

$$\|B_k^{-1}(x_k - x^*)\| \le r_k \tag{2.2}$$

holds for $k \in \mathbb{N}$.

At each iteration of Algorithm 1, the matrix B_k is updated in Step 3. Obviously, the update requires $O(n^2)$ operations and can be described based on the space dilation operator $R_\alpha(\xi) : \mathbb{R}^n \to \mathbb{R}^n$ with

$$R_\alpha(\xi) := I_n + (\alpha - 1)\xi\,\xi^\top,$$

where $\xi \in \mathbb{R}^n$ with $\|\xi\| = 1$ is the direction of dilation and $I_n \in \mathbb{R}^{n\times n}$ the identity matrix. The properties of this operator were studied in detail in [8, 9]. Setting $\beta := 1/\alpha$ and denoting the inverse dilation operator by $R_\alpha^{-1}(\xi)$, we have

$$R_\alpha^{-1}(\xi) = R_\beta(\xi)$$

and

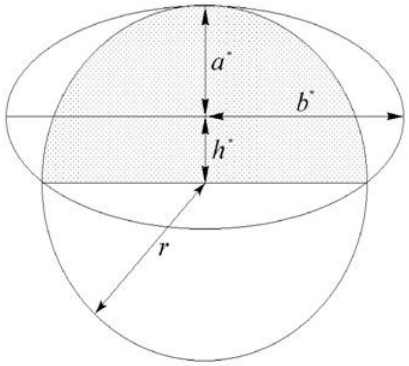
$$B_{k+1} = B_k R_\beta(\xi_k).$$

The latter shows the meaning of space dilation for the update of the B-matrices.

Algorithm 1 uses an ellipsoid of smaller volume containing a half-ball of radius r in $\mathbb{R}^n$ ($n \ge 2$). Such an ellipsoid has an oblate shape in the direction ξ. The parameters of the ellipsoid are shown in Fig. 1, where a is the length of the minor semi-axis of the ellipsoid, b is the length of the major semi-axes of the ellipsoid (the number of such semi-axes is equal to $n-1$), h is the distance from the center of the ball to the center of the ellipsoid in the direction of its minor semi-axis.

The volume of this ellipsoid is $v_e = v_0 a b^{n-1}$ and the volume of the ball is $v_b = v_0 r^n$, where v_0 denotes the volume of the unit ball in $\mathbb{R}^n$. Therefore, the volume reduction factor is equal to

$$\frac{v_e}{v_b} = \left(\frac{a}{r}\right)\left(\frac{b}{r}\right)^{n-1} = \left(\frac{a}{b}\right)\left(\frac{b}{r}\right)^{n} = \frac{1}{\alpha}\left(\frac{1}{2}\left(\alpha+\frac{1}{\alpha}\right)\right)^n = q_n(\alpha).$$



$$a^* = r\frac{n}{n+1}$$

$$b^* = r\frac{n}{\sqrt{n^2-1}}$$

$$h^* = r\frac{1}{n+1}$$

FIGURE 1. The parameters of minimal volume ellipsoid containing a half-ball in $\mathbb{R}^n$.

For $\alpha = \frac{b}{a} = \sqrt{\frac{n+1}{n-1}}$, it follows that $q_n(\alpha) < 1$. To transform the ellipsoid, containing a half-ball, into a new ball, it is sufficient to dilate the space of variables in the direction of the minor semi-axis with the coefficient $\alpha = \frac{b}{a}$. This can be done using the operator of space dilation $R_\alpha(\xi)$, where the direction ξ coincides with the direction of the minor semi-axis of the ellipsoid.

If $X = \mathbb{R}^n$ is the original space of variables, then in the transformed space of variables $Y = R_\alpha(\xi)X$, we get a new ball of radius b, which contains the solution of our problem. Repeating this procedure, but for the new ball in the transformed space, we obtain Algorithm 1. Here, in Step 2, the direction of the minor semi-axis of the ellipsoid in the transformed space $Y_k = B_k^{-1}X$ is calculated and the transition to its center is performed. The calculated direction is used for the next space dilation, which is implemented in Step 3 by determining the matrix B_{k+1}. In the next transformed space $Y_{k+1} = B_{k+1}^{-1}X$, we get a ball of radius r_{k+1}.

2.2. **The H-form of the ellipsoid method.** The B-form of the ellipsoid method (Algorithm 1) can be written in H-form by means of positive definite symmetric matrices H_k. This is presented and discussed below.

Algorithm 2 – The H-form of the ellipsoid method

Step 0. Choose $x_0 \in \mathbb{R}^n$, a positive definite symmetric matrix $H_0 \in \mathbb{R}^{n\times n}$, and $r_0 > 0$ so that

$$(x_0 - x^*)^\top H_0^{-1}(x_0 - x^*) \le r_0^2.$$

Moreover, set $k := 0$.

Step 1. If $g(x_k) = 0$, then set $x^* := x_k$ and STOP.

Step 2. Calculate

$$x_{k+1} := x_k - h_k \frac{H_k g(x_k)}{\sqrt{g(x_k)^\top H_k g(x_k)}}, \quad \text{where } h_k := \frac{1}{n+1} r_k.$$

Step 3. Update

$$H_{k+1} := H_k - \frac{2}{n+1}\frac{H_k g(x_k) g(x_k)^\top H_k}{g(x_k)^\top H_k g(x_k)} \quad \text{and} \quad r_{k+1} := \frac{n}{\sqrt{n^2-1}} r_k.$$

Step 4. Set $k := k+1$ and go to Step 1.

The sequence $\{x_k\}$ generated by Algorithm 2 is (theoretically) identical to the corresponding sequence generated by Algorithm 1 provided that in Step 0 of the latter the same values for x_0, r_0 are chosen as for Algorithm 2 and B_0 is chosen so that $H_0 = B_0 B_0^\top$ holds. To see that the H-form indeed produces the same sequence $\{x_k\}$, one inductively shows by simple calculations that

$$x_{k+1} = x_k - h_k B_k \xi_k = x_k - h_k \frac{H_k g(x_k)}{\sqrt{g(x_k)^\top H_k g(x_k)}},$$

$$B_{k+1} B_{k+1}^\top = H_k - \frac{2}{n+1} \frac{H_k g(x_k) g(x_k)^\top H_k}{g(x_k)^\top H_k g(x_k)}$$

holds for all $k \in \mathbb{N}$ for Algorithm 1 meaning that

$$H_{k+1} = B_{k+1} B_{k+1}^\top$$

is valid for $k \in \mathbb{N}$. According to this, $\mathscr{E}_k$ has equivalent representations by means of B_k and H_k-matrices, namely

$$\mathscr{E}_k = \{x \mid \|B_k^{-1}(x_k - x)\| \le r_k\} = \{x \mid (x_k - x) H_k^{-1} (x_k - x) \le r_k^2\}.$$

Hence, Theorems 2.1 and 2.2 are valid for Algorithm 2 as well.

On the one hand, computing a sequence $\{x_k\}$ by Algorithm 2 requires just a half of the operations than needed by Algorithm 1. The RAM memory usage shows about the same relation, if the B_k matrices are nonsymmetric. On the other hand, the H-form is computationally less stable since the matrices H_k may become unsymmetric and indefinite. Let us demonstrate this by means of a small example.

For $n = 2$, $H_0 := I_2$, $g(x_{2k}) := (1,-1)^\top$ and $g(x_{2k+1}) := (2,1)^\top$ for $k \in \mathbb{N}$, the formula for updating H_k in Step 3 of Algorithm 2 (H-form of the ellipsoid method) then yields

$$H_{50} = \begin{pmatrix} 8.6163e{-}13 & 9.5855e{-}14 \\ 9.5914e{-}14 & 1.6273e{-}12 \end{pmatrix}, \quad H_{70} = \begin{pmatrix} 9.9927e{-}18 & -1.8545e{-}17 \\ 4.0653e{-}17 & -3.5467e{-}17 \end{pmatrix},$$

where H_{50} is nonsymmetric and H_{70} is neither symmetric nor positive definite.

For Algorithm 1 (B-form of the ellipsoid method), such problems were not observed. If we look at the same example, the matrices B_{50} and B_{70} computed Algorithm 1 lead to

$$B_{50} B_{50}^\top = \begin{vmatrix} 8.6162e{-}13 & 9.5889e{-}14 \\ 9.5889e{-}14 & 1.6273e{-}12 \end{vmatrix}, \quad B_{70} B_{70}^\top = \begin{vmatrix} 1.4592e{-}17 & 1.6239e{-}18 \\ 1.6239e{-}18 & 2.7559e{-}17 \end{vmatrix},$$

i.e., $H_k = B_k B_k^\top$ stay symmetric and positive definite.

3. Ellipsoid algorithms for minimizing convex functions

The ellipsoid method can be used to find the (unconstrained) minimizer x^* of a convex function $f : \mathbb{R}^n \to \mathbb{R}$. The minimum value of f is denoted by $f^* := f(x^*)$. For simplicity, we assume that x^* is the only minimizer of f. To apply Algorithm 1 to the minimization problem just described let the mapping $g : \mathbb{R}^n \to \mathbb{R}^n$ used in Section 2 be specialized to $g_f : \mathbb{R}^n \to \mathbb{R}^n$ with $g_f(x)$ being a subgradient of f at x, which implies

$$(x - x^*)^\top g_f(x) \ge f(x) - f(x^*) = f(x) - f^* \ge 0 \quad \text{for all } x \in \mathbb{R}^n. \tag{3.1}$$

Moreover, compared with Algorithm 1, a more appropriate termination criterion is used in Step 1, which guarantees that Algorithm 3 below does not stop before

$$f(x_k) \leq f^* + \varepsilon \tag{3.2}$$

is fulfilled for some predefined $\varepsilon > 0$. This criterion is derived as follows. For any $(x_k, B_k, r_k) \in \mathbb{R}^n \times \mathbb{R}^{n\times n} \times (0,\infty)$ generated by Algorithm 1, we obtain

$$\begin{array}{rcl} f(x_k) - f^* & \leq & (x_k - x^*)^\top g_f(x_k) \\ & = & (B_k^{-1}(x_k - x^*))^\top B_k^\top g_f(x_k) \\ & \leq & \|B_k^{-1}(x_k - x^*)\| \|B_k^\top g_f(x_k)\| \\ & \leq & r_k \|B_k^\top g_f(x_k)\|, \end{array} \tag{3.3}$$

where the first inequality follows from (3.1) and the last is a consequence of $x^* \in \mathscr{E}_k$ according to Theorem 2.1. Hence, $r_k\|B_k^\top g_f(x_k)\| \leq \varepsilon$ implies that (3.2) is fulfilled.

The next algorithm includes this termination criterion. Moreover, instead of choosing any non-singular matrix B_0 as in Step 0 of Algorithm 1, the identity matrix is used below. Recall further that we now replace terms with α according to (2.1). To honor the work of Naum Z. Shor, the next algorithm is called Algorithm **emshor**, see [1, 13].

Algorithm 3 – Algorithm emshor

Step 0. Choose $x_0 \in \mathbb{R}^n$ and $r_0 > 0$ so that $\|x_0 - x^*\| \leq r_0$.
Moreover, choose $\varepsilon > 0$, set $B_0 := I_n$ and $k := 0$.

Step 1. If $\left\|B_k^\top g_f(x_k)\right\| r_k \leq \varepsilon$, then set $k^* := k$, $x_\varepsilon^* := x_k$ and STOP.

Step 2. Calculate

$$x_{k+1} := x_k - h_k B_k \xi_k, \quad \text{where} \quad \xi_k := \frac{B_k^\top g_f(x_k)}{\left\|B_k^\top g_f(x_k)\right\|}, \quad h_k := \frac{1}{n+1} r_k.$$

Step 3. Update

$$B_{k+1} := B_k + \left(\sqrt{\frac{n-1}{n+1}} - 1\right)(B_k\xi_k)\,\xi_k^\top \quad \text{and} \quad r_{k+1} := \frac{n}{\sqrt{n^2-1}} r_k.$$

Step 4. Set $k := k+1$ and go to Step 1.

The next theorem follows directly from Theorem 2.1, if one substitutes α according to its definition and by taking into account the stopping criterion in Step 1 of Algorithm 3 with the explanation above.

Theorem 3.1. *Let x_k and x_{k+1} be generated by Algorithm 3. Then, the ratio of volumes of the ellipsoids $\mathscr{E}_k$ and $\mathscr{E}_{k+1}$ does not depend on k and is equal to*

$$q_n := \frac{vol(\mathscr{E}_{k+1})}{vol(\mathscr{E}_k)} = \frac{n}{n+1}\left(\frac{n}{\sqrt{n^2-1}}\right)^{n-1} < \exp\left\{-\frac{1}{2n}\right\} < 1.$$

Moreover, $x^* \in \mathscr{E}_k$ holds for all $k = 0, 1, \ldots, k^*$ and, if Algorithm 3 stops, $f(x_\varepsilon^*) \leq f^* + \varepsilon$ is valid.

3.1. **Octave implementation of Algorithm emshor.** Algorithm 3 (Algorithm **emshor**) was implemented in Octave [4, 13]. It uses a function of the form `function[f,g]=calcfg(x)`, which calculates the value $f(x)$ and a subgradient $g_f(x)$ at x. This function has to be provided by the user. The code of the implementation and some short comments are given below.

Octave code for Algorithm emshor

```
# Input parameters:
# calcfg - name of the function for calculation of f and g
# x0 - starting point, x0(1:n)
# r0 - the radius of the ball localizing the minimum point
# epsf, maxitn - stop parameters (accuracy, max. iter.)
# intp - printing interval (after each intp iterations)
# Output parameters:
# x - approximation of the minimum point, x(1:n)
# f - value of function f at the point x
# itn - number of iterations performed
# ist - exit code (1=epsf, 4=maxitn)
function[x,f,itn,ist]=emshor(calcfg,x0,r0,epsf,maxitn,intp);    #row01
n=length(x0); x=x0; B=eye(n); r=r0;                             #row02
dn=double(n); beta=sqrt((dn-1.d0)/(dn+1.d0));                   #row03
for (itn=0:maxitn)                                              #row04
   [f,g1]=calcfg(x); g=B'*g1; dg=norm(g);                       #row05
   if((mod(itn,intp)==0)&&(intp<=maxitn))                       #row06
     printf("itn %4d  f %14.6e\n",itn,f);                       #row07
   endif                                                        #row08
   if(r*dg<epsf) ist=1;  return; endif                          #row09
   xi=(1.d0/dg)*g; dx=B*xi;                                     #row10
   hs=r/(dn+1.d0); x-=hs*dx;                                    #row11
   B+=(beta-1.d0)*B*xi*xi';                                     #row12
   r=r/sqrt(1.d0-1.d0/dn)/sqrt(1.d0+1.d0/dn);                   #row13
endfor                                                          #row14
ist=4;                                                          #row15
endfunction                                                     #row16
```

The iterative process is executed in a for-loop (rows 04–14), where Step 1 of Algorithm 3 is implemented in rows 05–09, Step 2 in rows 10–11, and Step 3 in rows 12–13. After every intp iterations in the for-loop intermediate results are printed (rows 06–08). The Octave code for Algorithm 3 stops if either the termination criterion in Step 1 of the algorithm is satisfied so that x_ε^* with $f(x_\varepsilon^*) \le f^* + \varepsilon$ is found (`ist=1` in row 09) or if the maximal number of iterations `maxitn` is reached (rows 04 and 15).

Let us mention that an Octave implementation of Algorithm 2 (The H-form of the ellipsoid method) can be easily derived from the above code by replacing the rows 02, 05, and 12 with:

```
 n=length(x0); x=x0; H=eye(n); r=r0;                            #row02
    [f,g1]=calcfg(x); g=H*g1; dg=sqrt(g1'*g);                   #row05
    H+=(beta*beta-1.d0)*xi*xi';                                 #row12
```

3.2. **Computational experiments for ravine function.** We will demonstrate the work of Algorithm 3 (Algorithm **emshor**) by means of its Octave code provided above. To this end, the code is applied to the minimization of two ravine quadratic and piecewise linear convex functions $f_1, f_2 : \mathbb{R}^n \to \mathbb{R}$ with

$$f_1(x) = \sum_{i=1}^{n} t^{i-1} (x_i - 1)^2, \quad f_2(x) = \sum_{i=1}^{n} t^{i-1} |x_i - 1| \tag{3.4}$$

for fixed $t \in \{1.2, 2.0\}$. Obviously, the unique minimizers of both functions f_1 and f_2 are equal to $x^* = (1, 1, \ldots, 1)^\top \in \mathbb{R}^n$ with the same optimal values $f_1^* = f_1(x^*) = f_2^* = f_2(x^*) = 0$. The ratio of the largest and smallest summands in f_1 and f_2 depends on the ratio of the coefficients t^{n-1} and $t^0 = 1$. For example, for $t = 1.2$ and $n = 100$, the ratio of coefficients is about 6.9e+07, whereas for $t = 2$ and n between 20 and 100, the ratio ranges approximately from 5.2e+05 to 6.3e+29.

Table 1 provides results obtained by the Octave code for Algorithm **emshor** for the smooth function f_1 with $t = 2$, where $x_0 = 0 \in \mathbb{R}^{10}$, $r_0 = 5$, and different ε-values were used. For the nonsmooth function f_2 with $t = 2$, the calculations were done for $x_0 = 0 \in \mathbb{R}^{10}$, $r_0 = 5$, and again with different values for ε, see Table 2. Results for f_1 and f_2 with $t = 1.2$ are shown in Table 3 for n up to 100, where $x_0 = 0$, $r_0 = 10$, $\varepsilon = 1.0\text{e-}16$ for f_1 and $\varepsilon = 1.0\text{e-}08$ for f_2 were used. These three tables show the number of iterations k^*, the function value $f(x_\varepsilon^*)$ at the last iteration, and the distance $\|x_\varepsilon^* - x^*\|$ from the solution.

TABLE 1. Results for minimizing the smooth function f_1 by Algorithm **emshor**

ε	k^*	$f(x_\varepsilon^*)$	$\|x_\varepsilon^* - x^*\|$	ε	k^*	$f(x_\varepsilon^*)$	$\|x_\varepsilon^* - x^*\|$
1.0e-02	685	3.1e-05	2.0e-03	1.0e-12	2938	2.9e-15	2.7e-08
1.0e-04	1137	4.2e-07	2.5e-04	1.0e-14	3452	4.2e-17	2.3e-09
1.0e-06	1580	2.3e-09	1.5e-05	1.0e-16	3926	8.9e-19	4.2e-10
1.0e-08	2055	1.8e-11	1.4e-06	1.0e-18	4463	1.4e-20	3.5e-11
1.0e-10	2502	5.3e-13	3.8e-07	1.0e-20	4889	6.4e-23	4.8e-12

TABLE 2. Results for minimizing the non-smooth function f_2 by Algorithm **emshor**

ε	k^*	$f(x_\varepsilon^*)$	$\|x_\varepsilon^* - x^*\|$	ε	k^*	$f(x_\varepsilon^*)$	$\|x_\varepsilon^* - x^*\|$
1.0e-02	2057	2.5e-04	1.4e-05	1.0e-10	5750	2.1e-12	8.6e-14
1.0e-04	2957	2.9e-07	3.4e-08	1.0e-12	6485	4.9e-14	7.4e-15
1.0e-06	3829	7.2e-08	1.1e-08	1.0e-14	6765	4.4e-16	4.4e-16
1.0e-08	4795	7.5e-10	1.3e-10	1.0e-16	6780	0.0e+00	0.0e+00

From Tables 1 and 2, it can be seen that Algorithm **emshor** finds very precise approximations to the minimizer of the ravine convex function. It can be observed from Table 3 that the number of iterations grows slightly faster than n^2 for the given initial data.

In practice, Algorithm 2 (The H-form of the ellipsoid method) is not able to approximate the minimum point of the ravine function f_2 with a reasonable accuracy. This is confirmed by numerical experiments with the modification of the Octave code for Algorithm **emshor**, in which the operations with the matrix B (in rows 5, 10 and 12) are replaced by the corresponding

TABLE 3. Results for minimizing f_1, f_2 by Algorithm **emshor**

	smooth function f_1			non-smooth function f_2		
n	k^*	$f(x_\varepsilon^*)$	$\|x_\varepsilon^* - x^*\|$	k^*	$f(x_\varepsilon^*)$	$\|x_\varepsilon^* - x^*\|$
10	3808	3.4e-19	3.9e-10	4484	8.2e-10	1.3e-10
20	15883	1.0e-18	4.3e-10	19044	4.7e-10	5.2e-11
50	104771	5.0e-19	3.1e-10	135113	6.9e-11	5.9e-13
100	454650	1.8e-19	6.9e-11	563705	5.3e-11	2.2e-12

operations with the symmetric matrix $H = BB^\top$. Results for applying this modified code to the minimization of function f_2 with $t = 2$, $\varepsilon = 1.0\text{e-}3$, $x_0 = 0$, and $r_0 \in \{5, 500\}$ are given in Table 4. In contrast to the other tables, `itn` denotes the total number of iterations performed, whereas `itr` (with `itr`$\leq$`itn`) is the index, where the smallest value of f_2 was observed during the iteration process.

TABLE 4. Results for applying the H-form of Algorithm **emshor** to f_2

	$r_0 = 5$				$r_0 = 500$			
n	`itn`	$f(x_{\texttt{itn}})$	`itr`	$f(x_{\texttt{itr}})$	`itn`	$f(\texttt{itn})$	`itr`	$f(x_{\texttt{itr}})$
5	461	1.3e-03	446	1.0e-05	453	2.1e-01	443	2.0e-03
10	1664	3.1e-02	1467	8.3e-05	1767	2.4e+00	1690	2.0e-03
15	6541	6.5e-05	6528	2.9e-07	8615	5.6e-05	7804	3.1e-07
20	5627	5.7e+00	5356	2.4e-03	5434	1.1e+04	4980	2.1e-01

From Table 4, it can be seen that the achieved accuracies are insufficient since $f_2(x_{\texttt{itn}}) \leq$ 1.0e-03 or $f_2(x_{\texttt{itnr}}) \leq$ 1.0e-03 could only be obtained for $n = 15$. In all cases, the function values increased before stopping. This bad behavior appears because the norm of the matrices H converges to zero much faster than the norm of the matrices B and due to the accumulation of rounding errors when updating the symmetric matrix H (see also the example at the end of Section 2).

3.3. **A unified presentation.** Here, we would like to introduce a parametrized version of Algorithm 3, see [14]. The additional parameter λ allows to obtain different versions of the ellipsoid methods by just adapting this λ.

Algorithm 4

Step 0. Choose $\lambda > 0$, $x_0 \in \mathbb{R}^n$, $r_0 > 0$, $\varepsilon > 0$ such that $\|x_0 - x^*\| \leq r_0$.
Set $B_0 := I_n \in \mathbb{R}^{n\times n}$ and $k := 0$.

Step 1. If $\left\|B_k^\top g_f(x_k)\right\| r_k \leq \varepsilon$, then set $k^* := k$, $x_\varepsilon^* := x_k$, and STOP.

Step 2. Calculate

$$x_{k+1} := x_k - h_k B_k \xi_k, \quad \text{where} \quad \xi_k := \frac{B_k^\top g_f(x_k)}{\left\|B_k^\top g_f(x_k)\right\|}, \quad h_k = \frac{1}{n+1} r_k.$$

Step 3. Update

$$B_{k+1} := \lambda \left(B_k + \left(\sqrt{\frac{n-1}{n+1}} - 1 \right) (B_k \xi_k)\, \xi_k^\top \right) \quad \text{and} \quad r_{k+1} := \frac{1}{\lambda} \frac{n}{\sqrt{n^2-1}} r_k.$$

Step 4. Set $k := k+1$ and go to Step 1.

For Algorithm 4, Theorem 2.2 (for $k = 0, \ldots, k^*$) and Theorem 3.1 hold for any parameter $\lambda \in (0, \infty)$. The proofs can be carried out similar to [1, Theorems 1 and 2].

For certain values of λ, existing variants of the ellipsoid method are obtained, namely those by Shor [7], Nemirovsky and Yudin [3, page 76]), and Khachiyan [2, Lemma 4]. A fourth almost unknown variant is given in [7] and denoted by [7]* in Tables 5 and 6. Details of these variants can be found in Table 5. All variants (with different λ) are theoretically equivalent, though slight differences could be observed due to the accumulation of numerical errors. For example, using an Octave code for Algorithm 4 for the minimization of the non-smooth function f_2 defined in (3.4) with $t = 2$, $x = 0 \in \mathbb{R}^{10}$, $r_0 = 5$, and $\varepsilon = 1.0\text{e-}08$, the results in Table 6 are obtained.

TABLE 5. Characteristics of four variants of the ellipsoid method

λ	Update of the B-matrix	Multiplications	Updated radius	Reference
1	$B_1 = B + \left(\sqrt{\frac{n-1}{n+1}} - 1 \right) (B\xi)\xi^\top$	$3n^2 + n$	$r_1 = \frac{n}{\sqrt{n^2-1}} r$	[7]
$\frac{n}{\sqrt{n^2-1}}$	$B_2 = \frac{n}{\sqrt{n^2-1}} B_1$	$4n^2 + n$	$r_2 = r$	[2]
$\sqrt[n]{\frac{n+1}{n-1}}$	$B_3 = \left(\frac{n+1}{n-1}\right)^{\frac{1}{2n}} B_1$	$4n^2 + n$	$r_3 = \left(\frac{n-1}{n+1}\right)^{\frac{1}{2n}} r_1$	[3]
$\left(\frac{n}{\sqrt{n^2-1}}\right)^{\frac{3}{2}}$	$B_4 = \left(\frac{n}{\sqrt{n^2-1}}\right)^{\frac{3}{2}} B_1$	$4n^2 + n$	$r_4 = \left(\frac{\sqrt{n^2-1}}{n}\right)^{\frac{1}{2}} r$	[7]*

TABLE 6. Results for Algorithm 4 applied to f_2 for different values of λ

ε	λ according to	k^*	$f(x_\varepsilon^*)$	$\|x_\varepsilon^* - x^*\|$	$\|B_{k^*}\|$	r_{k^*}
1.0e-07	Shor [7]	4351	9.2e-09	9.2e-10	2.6e-18	1.6e+10
1.0e-07	Khachiyan [2]	4351	9.2e-09	9.2e-10	8.2e-09	5.0e+00
1.0e-07	Nemirovski and Yudin [3]	4351	9.2e-09	9.2e-10	2.4e+01	1.7e-09
1.0e-07	Shor [7]*	4351	9.2e-09	9.2e-10	4.6e-04	8.9e-05
1.0e-08	Shor [7]	4821	9.0e-10	1.1e-10	2.3e-20	1.7e+11
1.0e-08	Khachiyan [2]	4807	9.2e-10	1.8e-10	7.9e-10	5.0e+00
1.0e-08	Nemirovski and Yudin [3]	4811	9.3e-10	9.0e-11	2.5e+01	1.7e-10
1.0e-08	Shor [7]*	4819	9.0e-10	1.6e-10	1.4e-04	2.8e-05
1.0e-14	Shor [7]	6716	0.0e+00	0.0e+00	2.0e-29	2.3e+15
1.0e-14	Khachiyan [2]	6724	2.2e-16	2.2e-16	2.4e-14	5.0e+00
1.0e-14	Nemirovski and Yudin [3]	6741	1.1e-16	1.1e-16	4.0e+00	1.1e-14
1.0e-14	Shor [7]*	6738	6.7e-16	6.7e-16	3.1e-07	2.2e-07

4. Further applications of the ellipsoid method

In this section, we briefly review how the ellipsoid method can be applied to convex programs and saddle point problems for convex-concave functions.

4.1. **Constrained convex programming.** Let us consider the constrained program

$$\underset{x}{\text{minimize}} \quad f_0(x) \quad \text{subject to} \quad f_i(x) \le 0 \quad i = 1,2,\dots,m \tag{4.1}$$

where, for $i = 0,1,\dots,m$, the functions $f_i : \mathbb{R}^n \to \mathbb{R}$ are convex and $g_i : \mathbb{R}^n \to \mathbb{R}^n$ denotes a mapping so that $g_i(x)$ is a subgradient of f_i at x. Let us assume that problem (4.1) has the unique solution x^* and that the Slater condition is satisfied. Moreover, let the mapping $g : \mathbb{R}^n \to \mathbb{R}^n$ be defined by

$$g(x) = g_{i(x)}(x),$$

where the mapping $x \mapsto i(x)$ satisfies the conditions

$$\begin{aligned} i(x) &= 0, && \text{if } f_1(x) \le 0,\dots,f_m(x) \le 0, \\ i(x) &\in \{i \mid f_i(x) > 0\}, && \text{if } \max\{f_1(x),\dots,f_m(x)\} > 0. \end{aligned} \tag{4.2}$$

Then, it can be seen that $(x-x^*)^\top g(x) \ge 0$ holds for all $x \in \mathbb{R}^n$. For this purpose, let us first consider the case that x satisfies $\max\{f_1(x),\dots,f_m(x)\} \le 0$. According to (4.2), we have $g(x) = g_0(x)$. Moreover, by $f_0(x) \ge f_0(x^*)$ and the convexity of f, we have

$$(x-x^*)^\top g(x) = (x-x^*)^\top g_0(x) \ge f_0(x) - f_0(x^*) \ge 0.$$

If, otherwise, $\max\{f_1(x),\dots,f_m(x)\} > 0$, then (4.2) implies $f_j(x) > 0$ for $j := i(x)$. By the convexity of f_j and by $f_j(x^*) \le 0$, we get

$$(x-x^*)^\top g(x) = (x-x^*)^\top g_j(x) \ge f_j(x) - f_j(x^*) \ge 0.$$

Thus, $(x-x^*)^\top g(x) \ge 0$ holds for all $x \in \mathbb{R}^n$.

Hence, to approximate x^* we can apply one of the previous algorithms. In particular, we may use the same stopping criterion as in Algorithm 3, see (3.3) as well. At the end of this subsection, we would like to mention the possibility of equivalently reformulating (4.1) as convex unconstrained minimization problem by penalizing the constraints if a sufficiently large penalty parameter is known. This enables the application of Algorithm **emshor** or of an other appropriate algorithm to the resulting convex unconstrained program. In [15], this approach is studied for the reformulation of linear programs and the use of an r-algorithm, for the latter see [8, 9, 11].

4.2. **Saddle point problems of convex-concave functions.** Let $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ be a convex-concave function, i.e., $f(\cdot,y) : \mathbb{R}^n \to \mathbb{R}$ is convex for each $y \in \mathbb{R}^m$ and $f(x,\cdot) : \mathbb{R}^m \to \mathbb{R}$ is concave for each $x \in \mathbb{R}^n$. A pair $z^* := (x^*,y^*) \in \mathbb{R}^n \times \mathbb{R}^m$ is called saddle point of f if

$$f(x^*,y) \le f(x^*,y^*) \le f(x,y^*) \quad \text{for all } (x,y) \in \mathbb{R}^n \times \mathbb{R}^m.$$

Moreover, with $z := (x,y)$, let the set $G_x(z) \subset \mathbb{R}^n$ contain all subgradients of $f(\cdot,y)$ with respect to x, whereas $G_y(z) \subset \mathbb{R}^m$ contains all supergradients of $f(x,\cdot)$ with respect to y. Based on this, we can define the mapping $g : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n \times \mathbb{R}^m$ by

$$g(z) := \begin{pmatrix} g_x(z) \\ -g_y(z) \end{pmatrix} \quad \text{with} \quad (g_x(z),g_y(z)) \in G_x(z) \times G_y(z).$$

By the above definition of a saddle point and by the convexity of the functions $f(\cdot,y)$ and $-f(x,\cdot)$ for each fixed y and x, respectively, it follows that

$$\begin{aligned} 0 &\leq f(x,y^*)-f(x^*,y) \\ &= f(x,y^*)-f(x,y)+f(x,y)-f(x^*,y) \\ &\leq -g_y(z)^\top(y-y^*)+g_x(z)^\top(x-x^*) \\ &= g(z)^\top(z-z^*). \end{aligned}$$

Thus, $g(z)^\top(z-z^*)\geq 0$ holds for all $z\in\mathbb{R}^n\times\mathbb{R}^m$ so that the ellipsoid method can be used to approximate the saddle point z^*.

There are several other cases, where the ellipsoid method can be applied. For example, for solving non-smooth problems of small dimensions that occur in decomposition schemes (by constraints, by variables), for special convex problems with a small number of variables with a parametrically given family of constraints. The main questions are always to determine a rule for constructing cutting hyperplanes, which localize the point to find, and to develop an appropriate stopping criterion.

5. Accelerating the ellipsoid method

As it was explained in Section 2, the ellipsoid method can be described by means of the space dilation operator

$$R_\alpha(\xi)=I+(\alpha-1)\xi\xi^\top.$$

It transforms the ellipsoid, containing the half-ball in $\mathbb{R}^n$, into a new ball after one space dilation. Instead of this, we consider the case of receiving a new ball after two space dilations.

The key feature of this technique is the transformation of the ellipsoid $Ell(x_0,a,b,r)$ into a ball. The minimum volume ellipsoid $Ell(x_0,a,b,r)$ is centered at x_0 and contains the convex set $W\subset\mathbb{R}^n$ resulting from the intersection of the ball $S(x_0,r):=\{x\in\mathbb{R}^n \mid \|x-x_0\|\leq r\}$ with the two half-spaces

$$\begin{aligned} P(x_0,\xi) &:= \{x\in\mathbb{R}^n \mid (x-x_0)^\top\xi\leq 0\}, \\ P(x_0,\eta) &:= \{x\in\mathbb{R}^n \mid (x-x_0)^\top\eta\leq 0\}, \end{aligned}$$

where $-1<\xi^\top\eta<0$, $\|\xi\|=1$, $\|\eta\|=1$ holds. The ellipsoid $Ell(x_0,a,b,r)$ has the following

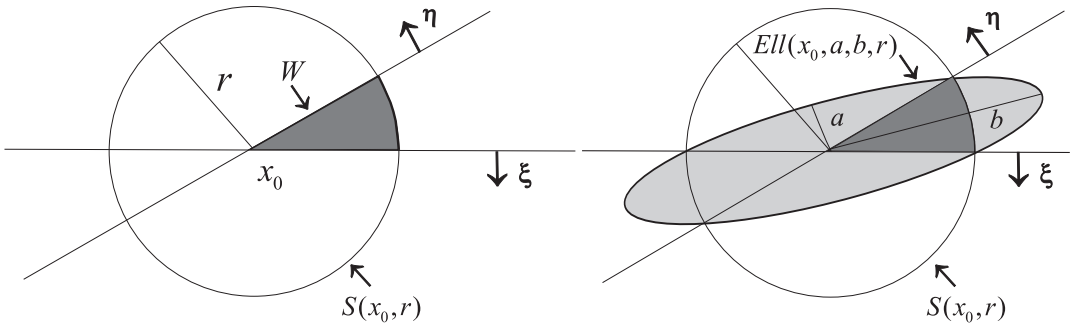


FIGURE 2. Projections of the set W and the 2d-ellipsoid $Ell(x_0,a,b,r)$.

parameters: the length of the semi-axis in the direction $(\xi-\eta)$ is equal to $a=r\sqrt{1+(\xi,\eta)}<r$; the length of the semi-axis in the direction $(\xi+\eta)$ is $b=r\sqrt{1-(\xi,\eta)}>r$; in the other $(n-2)$

directions orthogonal to ξ and η, the lengths of the semi-axes are equal to r. The ratio of the volume of $Ell(x_0,a,b,r)$ and the ball volume is equal to $q := (a/r)(b/r) = \sqrt{1-(\xi^\top\eta)^2} < 1$. The ratio decreases when the angle between ξ and η approaches 180°.

Lemma 5.1 ([11]). *Let $B_k \in \mathbb{R}^{n\times n}$, $r > 0$, $x_k, x^* \in \mathbb{R}^n$, and $g_1, g_2 \in \mathbb{R}^n$ be given such that $\|B_k^{-1}(x_k - x^*)\| \le r$, $(x_k - x^*)^\top g_1 \ge 0$, $(x_k - x^*)^\top g_2 \ge 0$, and $g_1^\top B_k B_k^\top g_2 < 0$ holds. Then, the updated matrix*

$$B_{k+1} := B_k R_{\beta_1}\left(\frac{\xi-\eta}{\|\xi-\eta\|}\right) R_{\beta_2}\left(\frac{\xi+\eta}{\|\xi+\eta\|}\right)$$

with

$$\beta_1 := \sqrt{1+\xi^\top\eta}, \quad \beta_2 := \sqrt{1-\xi^\top\eta}, \quad \xi := \frac{B_k^\top g_1}{\|B_k^\top g_1\|}, \quad \eta := \frac{B_k^\top g_2}{\|B_k^\top g_2\|}$$

has the following properties:

$$\begin{array}{ll} (i) & \|B_{k+1}^{-1}(x_k - x^*)\| \le r, \\ (ii) & \det(B_{k+1}) = \det B_k \sqrt{1-(\xi^\top\eta)^2}, \text{ and} \\ (iii) & g_1^\top B_{k+1} B_{k+1}^\top g_2 = 0. \end{array}$$

Lemma 5.1 can be interpreted as follows. Property (i) means that $y^* := B_{k+1}^{-1}x^*$ belongs to the ball $S(y_k,r)$ in the transformed space $Y := B_{k+1}^{-1}X$, where $y_k := B_{k+1}^{-1}x_k$. Property (ii) shows that the volume of ellipsoid $Ell(y_k,a,b,r)$ decreases in comparison to the volume of the ball $S(y_k,r)$ and this decrease will be the bigger the larger the obtuse angle between ξ and η is. Property (iii) provides the basis for the anti-ravine technique, similar to that used in Shor's r-algorithm [8, 9]. Subgradients with an obtuse angle in the original space of variables become orthogonal in the transformed space. This yields a less strong ravine-type shape. In this case, coefficients of space dilation in the direction of the difference of the normalized subgradients and in the direction of the sum of normalized subgradients are determined by the angle between subgradients. A more obtuse angle leads to a larger value of the coefficient of space dilation in the direction of the difference between the two normalized subgradients.

The ellipsoid $Ell(x_0,a,b,r)$ (see Fig. 2) can be used to develop accelerated variants of ellipsoid methods for solving convex programs and saddle point problems of convex-concave functions. For such accelerated methods, we can expect a convergence rate close to that of r-algorithms. This is confirmed by numerical experiments for subgradient methods with transformation of space for finding the minimizer of a convex function with a priori knowledge of the minimal function value. In particular, these methods turned out to be quite useful for dealing with ravine functions.

6. Conclusion

In this paper, we reviewed the ellipsoid method and the properties of two theoretically equivalent versions. The first one relies on updating a not necessarily symmetric matrix B, as in the ellipsoid method of Shor [7]. For the second version, a symmetric matrix $H = BB^\top$ is updated, as proposed in Skokov [10]. Based on the ellipsoid method in B-form, an Octave implementation for the problem of unconstrained minimization of a convex function allows the user to find a very accurate approximation of the minimizer of a convex ravine function in a reasonable amount of time. Further computational aspects of the application of the ellipsoid method are

dealt with as well. We also took into account a unified way of presenting the ellipsoid method, so that the description and implementation of the (theoretically equivalent) ellipsoid methods by Shor [7], by Khachiyan [2], and by Nemirovski and Yudin [3] is possible. In addition, further possible applications of ellipsoid methods were discussed as well as a technique for accelerating ellipsoid methods.

Acknowledgements

The authors are pleased to acknowledge the support by the Volkswagen Foundation under grant number 97 775 and by the National Research Foundation of Ukraine under grant number 2021.01/0136. Moreover, we would like to thank an anonymous reviewer for helpful comments.

REFERENCES

[1] A.F. Izmailov, P.I. Stetsyuk, A. Fischer, Emshor algoritm and its octave implementation, Computer Mathematics, 1 (2019) 132-142.
[2] L.G. Khachiyan, Polynomial algorithms in linear programming, USSR Comput. Math. Math. Phys. 20 (1) (1980) 53-72.
[3] A.S. Nemirovsky, D.B. Yudin, Problem Complexity and Method Efficiency in Optimization, John Wiley, New York, 1983, translated from book published by Nauka, Moscow, 1979.
[4] Octave, http://www.octave.org.
[5] N.Z. Shor, Methods of minimization of nondifferentiable functions and their applications (in Russian), Ph.D. thesis, Institute of Cybernetics of the Academy of Sciences of the Ukrainian SSR, Kyiv, 1970.
[6] N.Z. Shor, Convergence rate of the gradient descent method with dilatation of the space, Cybernetics 6 (1970) 102–108.
[7] N.Z. Shor, Cut-off method with space extension in convex programming problems, Cybernetics 13 (1977) 94-96.
[8] N.Z. Shor, Minimization Methods for Non-Differentiable Functions, Springer, Berlin, 1985.
[9] N.Z. Shor, Nondifferentiable Optimization and Polynomial Problems, Kluwer, Amsterdam, 1998.
[10] V.A. Skokov, Note on minimization methods employing space stretching, Cybernetics 10 (1974) 689-692.
[11] P.I. Stetsyuk, *r*-Algorithms and ellipsoids. Cybern. Syst. Anal. 32 (1996) 93-110.
[12] P.I. Stetsyuk, O.V. Fesiuk, O.N. Khomyak, The generalized ellipsoid method. Cybern. Syst. Anal. 54 (2018) 576-584.
[13] P.I. Stetsyuk, A. Fischer, O. Khomyak, The generalized ellipsoid method and its implementation, In: M. Jaćimović, M. Khachay, V. Malkova, M. Posypkin (Eds.) Optimization and Applications. OPTIMA 2019. Communications in Computer and Information Science, vol 1145, pp. 355-370, Springer, Cham, 2020.
[14] P.I. Stetsyuk, A. Fischer, O.M. Khomiak, Ellipsoid methods with space scaling. In: F. Aliev, T. Başar (Eds.) Proceedings of the 8th International Conference on Control and Optimization with Industrial Applications, Volume I, pp. 402-404, Baku, 2020.
[15] P. Stetsyuk, A. Fischer, O. Pichugina, A penalty approach to linear programs with many two-sided constraints, In: P. Pardalos, M. Khachay, A. Kazakov (Eds.) Mathematical Optimization Theory and Operations Research. MOTOR 2021. pp. 206-217, Lecture Notes in Computer Science, vol 12755, Springer, Cham, 2021.
[16] D.B. Yudin, A.S. Nemirovskii, Informational complexity and efficient methods for the solution of convex extremal problems, Matekon 13 (1976) 25-45.