УДК 519.85
DOI: 10.18523/2617-3808.2019.2.16-21

Стецюк П. І., Фішер А., Ляшко В. І.

МЕТОД ЕЛІПСОЇДІВ ДЛЯ МІНІМІЗАЦІЇ ОПУКЛОЇ ФУНКЦІЇ

Розглянуто метод еліпсоїдів та його застосування для знаходження наближення до точки мінімуму опуклої функції: критерій зупинки гарантує знаходження такої точки, в якій значення функції відрізняється від мінімального не більше, ніж на задану достатньо малу величину. Метод є частковим випадком субградієнтних методів із розтягом простору в напрямку субградієнта з коефіцієнтом, який залежить тільки від вимірності простору змінних. Він може бути використаний для мінімізації гладких та негладких опуклих функцій від декількох десятків змінних.

Ключові слова: метод еліпсоїдів, перетворення простору, коефіцієнт розтягу простору, опукла функція, субградієнт, яружна негладка функція.

Вступ

Класичний метод еліпсоїдів вперше запропонували в 1976 р. Д. Б. Юдін і А. С. Немировський [5]. Вони виходили зі схеми послідовних відсікань і назвали метод еліпсоїдів модифікованим методом центрованих відсікань (ММЦВ). Незалежно метод еліпсоїдів перевідкрив у 1977 р. Н. З. Шор [4]. Він представлений як частковий випадок субградієнтних методів із розтягом простору в напрямку субградієнта, які були запропоновані Н. З. Шором в 1969–1970 р. Узагальнений метод еліпсоїдів [1] є алгоритмом із розтягом n-вимірного простору, де коефіцієнт розтягу α задовольняє нерівності $\alpha + \frac{1}{\alpha} < 2\sqrt[n]{\alpha}$. За $\alpha = \sqrt{\frac{n+1}{n-1}}$ узагальнений метод еліпсоїдів збігається з класичним методом еліпсоїдів Юдіна – Немировського – Шора.

У статті розглянемо дві алгоритмічні реалізації узагальненого методу еліпсоїдів: перша вимагає корекції несиметричної матриці B, як у методі Шора, а друга – корекції симетричної матриці $H = BB^T$, як у методі Юдіна – Немировського. Для знаходження наближення до точки мінімуму опуклої функції виберемо більш стійкий варіант класичного методу еліпсоїдів у B-формі. Він використовує перерахунок несиметричної матриці (обернена матриця перетворення простору) та критерій зупинки, який для опуклої функції гарантує знаходження такої точки, в якій значення функції відрізняється від мінімального не більше, ніж на задану достатньо малу величину.

Матеріал статті викладено в такому порядку. У розділі 1 описано узагальнений метод еліпсоїдів у B- та H-формах та наведено його властивості. У розділі 2 наведено опис алгоритму **emshor**, який використовує алгоритмічну реалізацію класичного методу еліпсоїдів у B-формі, запропонованій Н. З. Шором. Тут наведено результати тестових експериментів для яружної негладкої опуклої функції (функція із сильно витягнутими поверхнями рівня).

1. Узагальнений метод еліпсоїдів та його властивості

Метод призначений для розв'язання такої задачі. Нехай на $R^n (n \ge 1)$ задано векторне поле $g(x)$, $g(x) \in R^n$. Необхідно знайти точку x^*, таку, що $(g(x), x - x^*) \ge 0$ для всіх $x \in R^n$. Вважається, що x^* існує та $g(x) \ne 0$ для $x \ne x^*$. Тут P^n – евклідовий простір розмірності n зі скалярним добутком $x^T y$.

Узагальнений метод еліпсоїдів має такий вигляд.

Ініціалізація. Вибираємо коефіцієнт розтягу α, який задовольняє нерівності $\alpha + \frac{1}{\alpha} < 2\sqrt[n]{\alpha}$, точку $x_0 \in R^n$ і радіус r_0 такими, щоб $\left\| B_0^{-1}(x_0 - x^*) \right\| \le r_0$, де B_0 – $n \times n$-матриця. Перейдемо до наступної ітерації зі значеннями x_0, r_0, B_0.

Ітераційний процес. Нехай на k-й ітерації знайдені $x_k \in R^n$, r_k і $n \times n$-матриця B_k. Для переходу до $(k+1)$-ї ітерації виконуємо такі дії.

Крок 1. Обчислимо $g_k = g(x_k)$. Якщо $g_k = 0$, то ОСТАНОВ ($x^* = x_k$).

Крок 2. Обчислимо наступну точку

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

Крок 3. Перерахуємо матрицю B_{k+1} і радіус r_{k+1}

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

Крок 4. Переходимо до $(k+1)$-ї ітерації з x_{k+1}, r_{k+1} і B_{k+1}.

Теорема 1 [3]. Послідовність точок $\{x_k\}_{k=0}^{\infty}$, що генерується узагальненим методом, еліпсоїдів задовольняє нерівностям

$$\left\| A_k(x_k - x^*) \right\| \le r_k, \quad k = 0,1,2,\ldots, \tag{1}$$

де $A_k = B_k^{-1}$. Відношення об'ємів еліпсоїдів $E_k = \{x : \left\| A_k(x_k - x) \right\| \le r_k\}$ та $E_{k+1} = \{x : \left\| A_{k+1}(x_{k+1} - x) \right\| \le r_{k+1}\}$, що локалізують точку x^*, є величина стала і дорівнює

$$q_n(\alpha) = \frac{vol(E_{k+1})}{vol(E_k)} = \frac{1}{\alpha}\left(\frac{1}{2}\left(\alpha + \frac{1}{\alpha}\right)\right)^n < 1,\ k = 0,1,2,\ldots \tag{2}$$

На кожній ітерації узагальненого методу еліпсоїдів коригується матриця B, яка пов'язана з заміною змінних $x = By$. Тому його називають B-формою узагальненого методу еліпсоїдів.

Узагальнений метод еліпсоїдів можна записати у H-формі (як ММЦВ Юдіна–Немировського) за допомогою додатно визначеної симетричної матриці $H_k = B_k B_k^T$, $k = 0,1,2,\ldots$. Для коефіцієнта розтягу α, який задовільняє нерівності $\alpha + \frac{1}{\alpha} < 2\sqrt[n]{\alpha}$, H-форма узагальненого методу еліпсоїдів має такий вигляд.

Ініціалізація. Вибираємо точку $x_0 \in R^n$ і радіус r_0 такими, щоб $(x_0 - x^*)^T H_0^{-1}(x_0 - x^*) \le r_0^2$, де H_0 – додатно визначена симетрична $n \times n$-матриця. Перейдемо до наступної ітерації зі значеннями x_0, r_0, H_0.

Ітераційний процес. Нехай на k-й ітерації знайдені $x_k \in R^n$, r_k і симетрична $n \times n$ -матриця H_k. Для переходу до $(k+1)$-ї ітерації виконуємо такі дії.

Крок 1. Обчислюємо $g_k = g(x_k)$. Якщо $g_k = 0$, то ОСТАНОВ($x^* = x_k$).

Крок 2. Обчислюємо наступну точку

$$x_{k+1} := x_k - h_k \frac{H_k g_k}{\sqrt{g_k^T H_k g_k}}, \text{ де } h_k = \frac{1}{2}\left(1 - \frac{1}{\alpha^2}\right) r_k.$$

Крок 3. Перерахуємо матрицю H_{k+1} і радіус r_{k+1}

$$H_{k+1} := H_k + \left(\frac{1}{\alpha^2} - 1\right)\frac{H_k g_k g_k^T H_k}{g_k^T H_k g_k},\ r_{k+1} := \frac{1}{2}\left(\alpha + \frac{1}{\alpha}\right) r_k.$$

Крок 4. Переходимо до $(k+1)$-ї ітерації з x_{k+1}, r_{k+1} і H_{k+1}.

Теорема 2 [3]. Послідовність точок $\{x_k\}_{k=0}^{\infty}$, що генеруються H-формою узагальненого методу еліпсоїдів, задовольняє нерівностям

$$(x_k - x^*)^T H_k^{-1}(x_k - x^*) \le r_k^2, \quad k = 0,1,2,\ldots, \tag{3}$$

а відношення об'ємів еліпсоїдів

$$E_k = \{x : (x_k - x)^T H_k^{-1}(x_k - x) \le r_k^2\} \text{ і}$$

$$E_{k+1} = \{x : (x_{k+1} - x)^T H_{k+1}^{-1}(x_{k+1} - x) \le r_{k+1}^2\},$$

що локалізують точку x^*, є величиною сталою і рівною

$$q_n(\alpha) = \frac{vol(E_{k+1})}{vol(E_k)} = \frac{1}{\alpha}\left(\frac{1}{2}\left(\alpha + \frac{1}{\alpha}\right)\right)^n < 1, \quad k = 0,1,2,\ldots. \tag{4}$$

Як за оперативною памяттю, так і за трудоємністю (кількістю обчислень на ітерації) H-форма узагальненого методу еліпсоїдів економніша, ніж B-форма. Наприклад, за оперативною пам'яттю вона економніша майже в два рази, оскільки потрібно зберігати симетричну матрицю. Проте H-форма узагальненого методу еліпсоїдів обчислювально менш стійка до накопичення помилок, ніж B-форма. Це пов'язано з тим, що при реалізації методу в H-формі необхідно контролювати, щоб матриця H_k була додатньо визначеною на кожній ітерації. Для методу у B-формі такий контроль не потрібен, оскільки на ітерації обчислення пов'язані з додатньо визначеною матрицею $H_k \quad B_k B_k$.

У теоремах 1 і 2 співвідношення (2) і (4) означають, що метод еліпсоїдів збігається (за об'ємом локалізації точки x^*) зі швидкістю геометричної прогресії зі знаменником $q_n(\alpha) < 1$. Величина знаменника залежить від вибраного значення α, що задовільняє нерівності $\alpha + \frac{1}{\alpha} < 2\sqrt[n]{\alpha}$. Найменший знаменник прогресії реалізується в класичному методі еліпсоїдів Юдіна–Немировського–Шора. Йому відповідає коефіцієнт розтягу $\alpha_1 = \sqrt{\frac{n+1}{n-1}}$, і досягається він у точці мінімуму функції $q_n(\alpha)$ по α.

Узагальнений метод еліпсоїдів можна застосувати для знаходження розв'язку безумовної задачі мінімізації опуклої функції $f(x)$, де $x \in R^n$. Мінімальне значення $f(x)$ будемо позначати як $f^* = f(x^*)$ і, не обмежуючи загальності, вважатимемо, що точка x^* – єдина точка мінімуму. Нехай є апріорна інформація, що точка x^* розташована в кулі $S(x_0, R)$. Тоді, якщо векторне поле визначити за формулою $g(x) = g_f(x)$, де $g_f(x)$ – субградієнт функції $f(x)$ в точці x, то для нього буде виконуватися нерівність

$$\left(x - x^*, g(x)\right) = \left(x - x^*, g_f(x)\right) \ge f(x) - f(x^*) =$$
$$= f(x) - f^* \ge 0, \quad \forall x \in E^n. \tag{5}$$

Отже, для знаходження точки x^* можна використовувати метод еліпсоїдів, вказавши стартову точку x_0, початковий радіус $r_0 = R$ і матрицю $B_0 = I_n$, де I_n – одинична $n \times n$-матриця. Як критерій зупинки можна використовувати умову $r_k \left\| B_k^T g_f(x_k) \right\| \le \varepsilon$, яка за довільного малого ε дає змогу знайти точку $x_\varepsilon^* = x_k$, для якої $f(x_\varepsilon^*) - f^* \le \varepsilon$. Це випливає з нерівності

$$r_k \ge \left\| B_k^{-1}(x_k - x^*) \right\| \ge \left(B_k^{-1}(x_k - x^*), \frac{B_k^T g_f(x_k)}{\left\| B_k^T g_f(x_k) \right\|} \right) =$$

$$= \frac{(x_k - x^*, g_f(x_k))}{\left\| B_k^T g_f(x_k) \right\|} \ge \frac{f(x_k) - f^*}{\left\| B_k^T g_f(x_k) \right\|},$$

яка справедлива для опуклої функції $f(x)$ з урахуванням умови (5).

Для знаходження точки x_ε^* – наближення до x^* – точки мінімуму опуклої функції $f(x)$ нижче розглянемо алгоритм **emshor** (**e**llipsoid **m**ethod of **Shor**), який використовує алгоритмічну реалізацію класичного методу еліпсоїдів у B-формі, запропонованій Н. З. Шором.

2. Алгоритм emshor для знаходження x_ε^* та його Octave-реалізація

Вхідним параметром алгоритму є величина ε_f – точність, з якою потрібно знайти точку x_ε^*, для якої $f(x_\varepsilon^*) - f^* \le \varepsilon_f$, де $f^* = f(x^*)$, $g(x_k)$ – субградієнт функції $f(x)$ в точці x_k.

Алгоритм **emshor** для знаходження x_ε^* має такий вигляд.

Ініціалізація. Виберемо стартову точку $x_0 \in R^n$ та радіус r_0 так, щоб $\left\| x_0 - x^* \right\| \le r_0$. Введемо до розгляду $n \times n$-матрицю B та покладемо $B_0 := I_n$, де I_n – одинична $n \times n$-матриця. Перейдемо до першої ітерації з x_0, r_0 та B_0.

Нехай на ітерації k знайдено $x_k \in R^n$, r_k та B_k. Перехід до ітерації $(k+1)$ полягає у виконанні такої послідовності дій.

Крок 1. Обчислимо $f(x_k)$ та $g(x_k)$. Якщо $\left\| B_k^T g(x_k) \right\| r_k \le \varepsilon_f$, то «Зупинка: $k^* = k$ і $x_\varepsilon^* = x_k$». Інакше переходимо до кроку 2.

Крок 2. Обчислимо чергову точку

$$x_{k+1} := x_k - h_k B_k \xi_k,$$

$$\text{де} \quad \xi_k := \frac{B_k^T g(x_k)}{\left\| B_k^T g(x_k) \right\|}, \quad h_k = \frac{1}{n+1} r_k.$$

Крок 3. Обчислимо

$$B_{k+1} := B_k + \left(\sqrt{\frac{n-1}{n+1}} - 1 \right) \left(B_k \xi_k \right) \xi_k^T$$

$$\text{та} \quad r_{k+1} := r_k \frac{n}{\sqrt{n^2 - 1}}.$$

Крок 4. Перейдемо до ітерації $(k + 1)$ зі значеннями x_{k+1}, r_{k+1}, B_{k+1}.

Теорема 3. Послідовність точок $\{x_k\}_{k=0}^{k^*}$, що генерується алгоритмом **emshor**, задовольняє нерівностям

$$\left\| B_k^{-1}(x_k - x^*) \right\| \le r_k, \quad k = 0, 1, 2, \ldots, k^*.$$

На кожній ітерації k, де $1 \le k \le k^*$, відношення об'ємів еліпсоїдів $E_k = \left\{ x : \left\| B_k^{-1}(x_k - x) \right\| \le r_k \right\}$ та $E_{k-1} = \left\{ x : \left\| B_k^{-1}(x_{k-1} - x) \right\| \le r_{k-1} \right\}$, що локалізують x^*, є величина стала та рівна

$$q_n = \frac{vol(E_k)}{vol(E_{k-1})} = \frac{n}{n+1} \left(\frac{n}{\sqrt{n^2 - 1}} \right)^{n-1} < \exp\left\{ -\frac{1}{2n} \right\} < 1.$$

Наведений алгоритм реалізовано однойменною програмою **emshor** на мові Octave [6; 2]. Вона використовує octave-функцію вигляду **function [f, g] = calcfg (x)**, яка обчислює значення $f = f(x)$ та субградієнт $g = g(x)$ у точці x. Ця функція готується користувачем та може мати довільне ім'я, яке підтримує синтаксис Octave. Код програми **emshor**, що включає короткі коментарі, наведено нижче.

```
# Вхідні параметри:
#   calcfg – ім'я функції для обчислення f та g
#   x0 – стартова точка, x0(1:n)
#   rad – радіус кулі, що локалізує точку мінімуму
#   epsf, maxitn – параметри зупинки (точн., макс. ітер.)
#   intp – інтервал друку (через кожні intp ітерацій)
# Вихідні параметри:
#   x – знайдене наближення до точки мінімуму, x(1:n)
#   f – значення функції f в точці x
#   itn – кількість виконаних ітерацій
#   ist – код завершення (1 = epsf, 4 = maxitn)
function [x,f,itn,ist] = emshor(calcfg,x0,rad,           #row01
                                epsf,maxitn,intp);
dn=double(length(x0)); beta=sqrt((dn-1.d0)/(dn+1.d0));   #row02
x=x0; radn=rad; B=eye(length(x));                        #row03
for (itn = 0:maxitn)                                     #row04
   [f, g1] = calcfg(x); g=B'*g1; dg=norm(g);             #row05
   if(radn*dg < epsf) ist = 1;  return; endif            #row06
```

```
    xi=(1.d0/dg)*g; dx = B * xi;                        #row07
    hs=radn/(dn+1.d0); x -= hs * dx;                    #row08
    B += (beta - 1) * B * xi * xi';                     #row09
    radn=radn/sqrt(1.d0-1.d0/dn)/sqrt(1.d0+1.d0/dn);    #row10
    if(mod(itn,intp)==0)                                #row11
      printf("itn %4d  f %14.6e\n",itn,f);              #row12
    endif                                               #row13
  endfor                                                #row14
  ist = 4;                                              #row15
  endfunction                                           #row16
```

Програма **emshor** завершується виконанням однієї з двох умов: 1) знайдена точка x_ε^* – така, що $f(x_\varepsilon^*) - f^* \le \varepsilon_f$ (**ist=1**), 2) **maxitn** ітерацій виявилося недостатньо (**ist=4**).

Роботу програми **emshor** продемонструємо на тестовому прикладі, який полягає у мінімізації яружної кусочно-лінійної функції вигляду

$$f(x) = \sum_{i=1}^{n} 2^{i-1}\left|x_i - 1\right|, \quad f^* = f(x^*) = 0,$$
$$x^* = \left(1,1,\ldots,1\right)^T, \qquad (6)$$

де $|a|$ – абсолютна величина числа a, а коефіцієнти за $|x_i - 1|$ утворюють геометричну прогресію з показником $q = 2$. Яружність (ступінь витягнутості поверхонь рівня) для функції (6) визначається відношенням максимального коефіцієнта за $|x_i - 1|$ до мінімального. Якщо $n = 20$, то мінімальний коефіцієнт дорівнює $(2)^0=1$, а максимальний – $(2)^{19} \approx 5.24288e{+}05$.

У таблиці наведено результати роботи програми **emsho**r для трьох значень радіусів кулі, в якій локалізована точка мінімуму, r_0 = 5, 500, 50000. Як стартова точка вибиралась точка $x_0 = (0,0,...,0)^T$. Були проведені розрахунки для n = 5, 10, 15, 20 при трьох значеннях $\varepsilon_f = 10^{-3}$, 10^{-6}, 10^{-9}. Обчислення проводили на комп'ютері Pentium 2.5GHz у системі Windows XP за допомогою GNU Octave версії 3.0.0.

Таблиця

$r_0 = 5$									
	$\varepsilon_f = 10^{-3}$			$\varepsilon_f = 10^{-6}$			$\varepsilon_f = 10^{-9}$		
n	*itn*	$f(x_{itn})$	*time*	*itn*	$f(x_{itn})$	*time*	*itn*	$f(x_{itn})$	*time*
5	519	6.1e-06	0.18	873	1.1e-07	0.28	1201	1.2e-10	0.40
10	2484	8.7e-05	0.79	3829	7.2e-08	1.26	5246	7.9e-11	1.76
15	6561	6.5e-06	2.10	9667	6.0e-08	3.23	12786	1.5e-11	4.36
20	13101	4.8e-05	4.28	18714	3.5e-09	6.34	23416	2.0e-11	8.11
$r_0 = 500$									
	$\varepsilon_f = 10^{-3}$			$\varepsilon_f = 10^{-6}$			$\varepsilon_f = 10^{-9}$		
n	*itn*	$f(x_{itn})$	*time*	*itn*	$f(x_{itn})$	*time*	*itn*	$f(x_{itn})$	*time*
5	747	1.1e-05	0.25	1080	1.6e-07	0.35	1392	1.7e-10	0.46
10	3429	9.0e-05	1.08	4810	9.3e-08	1.58	6185	6.3e-11	2.06
15	8615	5.6e-05	2.78	11704	6.5e-08	3.90	14805	2.4e-11	5.02
20	16729	1.8e-06	5.49	22404	4.4e-08	7.64	27161	1.4e-11	9.38
$r_0 = 50000$									
	$\varepsilon_f = 10^{-3}$			$\varepsilon_f = 10^{-6}$			$\varepsilon_f = 10^{-9}$		
n	*itn*	$f(x_{itn})$	*time*	*itn*	$f(x_{itn})$	*time*	*itn*	$f(x_{itn})$	*time*
5	951	1.7e-04	0.32	1323	1.9e-08	0.42	1658	1.4e-10	0.55
10	4323	8.4e-05	1.38	5736	6.4e-08	1.88	7093	7.2e-11	2.36
15	10663	6.0e-05	3.45	13772	1.8e-08	4.60	16860	3.6e-11	5.70
20	20417	4.7e-05	6.75	26039	3.5e-08	8.80	30772	1.6e-11	10.54

Із таблиці бачимо, що за допомогою програми **emshor** можна знаходити досить точні наближення до точки мінімуму яружної опуклої функції від декількох десятків змінних. Наприклад, якщо n = 20, то для цього потрібно декілька секунд на сучасних персональних ЕОМ з використанням GNU Octave версій 3.0.0 та вище. При цьому кількість ітерацій слабо залежить від заданої точності ε_f та r_0 – радіусу кулі, у якій локалізована точка мінімуму. Це означає, що швидкість збіжності алгоритму **emshor** майже не залежить від яружності (ступеня витягнутості поверхонь рівня) опуклої функції.

Висновки

У роботі розглянуто узагальнений метод еліпсоїдів та властивості його двох алгоритмічних реалізацій. Перша базується на корекції несиметричної матриці B, як у методі еліпсоїдів Шора, а друга – на корекції симетричної матриці $H = BB^T$, як у методі еліпсоїдів Юдіна – Немировського. На основі класичного методу еліпсоїдів у B-формі побудовано алгоритм **emshor** (**e**llipsoid **m**ethod of **Shor**) та його програмну реалізацію мовою Octave для знаходження розв'язку задачі безумовної мінімізації опуклої функції. Показано, що алгоритм **emshor** дає змогу знаходити досить точні наближення до точки мінімуму яружної опуклої функції та для функцій від двадцяти змінних витрачає на це декілька секунд на сучасних ПЕОМ. Ці властивості алгоритму дають змогу використовувати його для розв'язання негладких підзадач при реалізації методів декомпозиції для блочних задач лінійного програмування з малою кількістю зв'язуючих змінних або обмежень.

Алгоритм **emshor** та його програмну реалізацію планується використати для розробки двоїстого методу розв'язання двоетапної транспортної задачі [3], коли кількість проміжних пунктів не більша, ніж двадцять. Обчислювальна складність двоїстого методу визначається складністю обчислення значення двоїстої функції та її суперградієнта, для чого потрібно знайти мінімальні елементи в двох одновимірних масивах, довжини яких m та n відповідають кількостям постачальників та споживачів. Це означає, що двоїстий метод може бути зорієнтований на випадок великих m та n (тисячі, десятки тисяч), для яких розв'язання відповідних двоетапній транспортній задачі задач лінійного програмування за допомогою програм загального призначення є неможливим або вимагає значних обчислювальних ресурсів.

Робота виконана за фінансової підтримки Volkswagen Foundation (грант No 90 306 – перший та другий автори).

Список літератури

1. Стецюк П. И. Обобщенный метод эллипсоидов / П. И. Стецюк, А. В. Фесюк, О. Н. Хомяк // Кибернетика и системный анализ. – 2018. – № 4. – С. 70–80.
2. Стецюк П. І. Метод еліпсоїдів та Octave-програма emshor / П. І. Стецюк, А. В. Івлічев // Міжнародний науковий симпозіум «ІНТЕЛЕКТУАЛЬНІ РІШЕННЯ». Теорія прийняття рішень: праці міжнар. школи-семінару, 15–20 квітня 2019 р. – Ужгород : УНУ, 2019. – С. 119–120.
3. Стецюк П. І. Двоетапна транспортна задача та її AMPL-реалізація / П. І. Стецюк, В. І. Ляшко, Г. В. Мазютинець // Наукові записки НаУКМА. Комп'ютерні науки. – 2018.– Т. 1. – С. 14–20.
4. Шор Н. З. Метод отсечения с растяжением пространства для решения задач выпуклого программирования / Н. З. Шор // Кибернетика. – 1977. – № 1. – С. 94–95.
5. Юдин Д. Б. Информационная сложность и эффективные методы решения выпуклых экстремальных задач / Д. Б. Юдин, А. С. Немировский // Экономика и математические методы. – 1976. – Вып. 2. – С. 357–369.
6. Octave [Електронний ресурс]. – Режим доступу: http://www.octave.org.

References

Judin, D. B., & Nemirovskij, A. S. (1976). Informacionnaja slozhnost' i jeffektivnye metody reshenija vypuklyh jekstremal'nyh zadach. *Jekonomika i matematicheskie metody*, *13* (3), 25–45 [in Russian].

Octave. Retrieved from http://www.octave.org.

Shor, N. Z. (1977). Metod otsechenija s rastjazheniem prostranstva dlja reshenija zadach vypuklogo programmirovanija. *Kibernetika, 13*, 94–96 [in Russian].

Stecjuk, P. I., Fesjuk, A. V., & Homjak, O. N. (2018). Obobshhennyj metod jellipsoidov. *Kibernetika i sistemnyj analiz*, *4*, 70–80 [in Russian].

Stetsyuk, P. I., & Ivlichev, A. V. (2019). Metod elipsoidiv ta Octave-prohrama emshor. *Mizhnarodnyi naukovyi sympozium "INTELEKTUALNI RISHENNIA". Teoriia pryiniattia rishen: pratsi mizhnar. shkoly-seminaru, 15–20 kvitnia 2019 r.*, Uzhhorod, 119–120 [in Ukrainain].

Stetsyuk, P. I., Liashko, V. I., & Maziutynets, H. V. (2018). Dvoetapna transportna zadacha ta yii AMPL-realizatsiia. *Naukovi zapysky NaUKMA. Kompiuterni nauky*, 1, 14–20 [in Ukrainain].

P. Stetsyuk, A. Fischer, V. Lyashko

AN ELLIPSOID METHOD FOR MINIMIZATION OF CONVEX FUNCTION

We consider the generalized ellipsoid method – an algorithm with the space dilation. For a certain choice of the dilation coefficient, this is a method of outer approximation of semi-ellipsoids by ellipsoids with a monotonous decrease in their volume. The Yudin-Nemirovski-Shor ellipsoid method is a specific case. The paper provides properties of two algorithmic realizations of the generalized ellipsoid method.

The first algorithm is based on updating nonsymmetric matrix B, as in the Shor ellipsoid method, and the second is based on updating symmetric matrix $H = BB^T$, *as in the Yudin-Nemirovski ellipsoid method.*

We present the ***Emshor*** *algorithm (****E****llipsoid* ***M****ethod of* ***Shor****) for computing a solution of the problem of unconstrained minimization of a convex function. It updates nonsymmetric matrix B and uses a stopping criterion that, for a convex function, guarantees to find a point at which the function value does not deviate more than a specified tolerance from the optimal function value. It is shown that the* ***Emshor*** *algorithm finds sufficiently accurate approximations to the minimum point of a ravine convex function, and for functions of twenty variables it takes no longer than a few seconds on a usual PC. Hence, the algorithm can be useful for solving small optimization problems.*

The ***Emshor*** *algorithm is planned to be used to develop a dual method for solving a two-stage transportation problem, when the number of intermediate points is not greater than twenty. The computational complexity of the dual method is determined by the complexity of calculating the value of the dual function and its supergradient, for which we need not more than twenty times to find the minimum elements in two one-dimensional arrays whose lengths m and n correspond to the numbers of suppliers and consumers. This means that the dual method can be oriented to the case of large m and n (thousands, tens of thousands), for which solving linear programming problems corresponding to the two-stage transportation problem by general-purpose programs is impossible or requires significant computational resources.*

Keywords: ellipsoid method, space transformation, coefficient of space dilation, convex function, subgradient, ravine non-smooth function.

Матеріал надійшов 13.05.2019