КІБЕРНЕТИКА та КОМП'ЮТЕРНІ ТЕХНОЛОГІЇ

УДК 519.85 DOI:10.34229/2707-451X.24.1.3

П.І. СТЕЦЮК, О.М. ХОМ'ЯК, О.О. ДАВИДОВ

ВИКОРИСТАННЯ МЕТОДУ ЕЛІПСОЇДІВ ДЛЯ ЗАДАЧІ СИЛЬВЕСТРА ТА ЇЇ УЗАГАЛЬНЕННЯ

Стаття присвячена дослідженню застосування алгоритму ***emshor*** *– алгоритму методу еліпсоїдів для розв'язання задачі Сильвестра про найменшу обмежувальну гіперсферу та її узагальнення. Узагальнена задача Сильвестра полягає у знаходженні центру кулі в n-вимірному просторі, яка має мінімальний радіус та містить скінченний набір n-вимірних куль, заданих їх центрами та радіусами. У статті побудовано алгоритми* ***sylvester1*** *та* ***sylvester2****, які є застосуванням алгоритму* ***emshor*** *для розв'язання відповідних задачам Сильвестра мінімаксних задач опуклого програмування. Обидва алгоритми демонструють високу точність знаходження центрів куль та хороші перспективи для їх застосування під час розв'язання задач в n-вимірних просторах, де* $n = 2 \div 30$*. Це підтверджується результатами обчислювальних експериментів.*

Ключові слова: *метод еліпсоїдів, опукла функція задача Сильвестра, кусково-гладка функція, мінімаксна задача.*

Вступ. Задача про найменше обмежувальне коло – це задача про побудову круга найменшого радіуса, який містить заданий набір точок евклідової площини. В n-вимірному просторі їй відповідає задача про найменшу обмежувальну гіперкулю, яка полягає у знаходженні кулі мінімального радіуса, що містить скінченний набір точок. Вперше задачу про найменше обмежувальне коло сформулював англійський математик Джеймс Джозеф Сильвестр у 1857 році [1].

Цю задачу досить часто необхідно вирішувати для $n = 2$ та $n = 3$. Так, наприклад, потрібно розмістити пункт швидкої допомоги таким чином, щоб максимально швидко доїхати до всіх пацієнтів. При дослідженні космічних проблем, де виникає потреба розташовувати супутники таким чином, щоб своїм радіусом дії вони охоплювали всі необхідні для спостережень об'єкти. Крім того, сигнал, що надходить з супутників має максимально швидко передаватися на інші станції. Задачі такого плану зводяться до побудови круга або кулі мінімального радіуса, що охоплює всі потрібні об'єкти.

Мета статті – дослідження застосування методу еліпсоїдів для розв'язання задач опуклого програмування, що є еквівалентними задачі знаходження кулі мінімального радіуса та її узагальненню.

Матеріал статті викладений у 3 розділах. У розділі 1 описано алгоритм методу еліпсоїдів та наведено теореми про його збіжність, описана Octave реалізація алгоритму. У розділі 2 описано застосування алгоритму для розв'язання задачі мінімізації опуклої кусково-квадратичної функції, яка є еквівалентною задачі знаходження кулі мінімального радіуса. У розділі 3 описано застосування алгоритму для розв'язання узагальнення задачі знаходження кулі мінімального радіуса на скінченний набір куль з заданими їх центрами та радіусами.

1. Метод еліпсоїдів для мінімізації опуклої функції

Класичний метод еліпсоїдів запропонували в 1976 р. Д.Б. Юдін та А.С. Немировський [2]. Вони отримали цей метод зі схеми послідовних відсікань та назвали його модифікованим методом центрованих перерізів. Незалежно від цього метод еліпсоїдів був відкритий і Н.З. Шором у 1977 р. [3]. У цій праці метод представлено як окремий випадок субградієнтних методів з розтягом простору, які були запропоновані Н.З. Шором наприкінці шістдесятих років [4, 5]. Зауважимо, що в 1970 р. Н.З. Шор був за крок до відкриття класичного методу еліпсоїдів. У цьому легко переконатися, якщо порівняти доведення теореми про збіжність методу еліпсоїдів, наведеної в [3], з доведенням теореми 2 у [5]. Далі опишемо метод еліпсоїдів для задачі мінімізації опуклої функції, який умовимося називати алгоритмом **emshor** (**e**llipsoid **m**ethod of **Shor**).

1.1. Опис алгоритму emshor. Нехай $f(x)$ – опукла функція, $x \in \mathbb{R}^n$. Позначимо її мінімальне значення як $f^* = f(x^*)$ та припустимо, що точка мінімуму x^* – єдина. Субградієнт $g(x)$ задовольняє таку умову:

$$\left\langle x - x^*, g(x) \right\rangle \geq f(x) - f^* \geq 0, \ \forall x \in \mathbb{R}^n. \tag{1}$$

Тут і далі $\langle x, y \rangle$ – скалярний добуток векторів $x \in \mathbb{R}^n$ і $y \in \mathbb{R}^n$.

Алгоритм **emshor** дозволяє знайти таку точку x_ε^*, для якої $f(x_\varepsilon^*) - f^* \leq \varepsilon$ і $\varepsilon > 0$ – досить мале. Він має такий вигляд.

Крок 0. Вибираємо точку $x_0 \in \mathbb{R}^n$ і радіус r_0 такими, щоб $\left\| x_0 - x^* \right\| \leq r_0$. Крім того, вибираємо $\varepsilon > 0$ і вважаємо, що $B_0 = I_n$ і $k := 0$.

Крок 1. Якщо $\left\| B_k^\top g(x_k) \right\| r_k \leq \varepsilon$, то ЗУПИНКА: $k^* := k$, $x_\varepsilon^* := x_k$.

Крок 2. Обчислюємо $x_{k+1} := x_k - h_k B_k \xi_k$, де $\xi_k := \dfrac{B_k^\top g(x_k)}{\left\| B_k^\top g(x_k) \right\|}$, $h_k := \dfrac{1}{n+1} r_k$.

Крок 3. Обновлюємо $B_{k+1} := B_k + \left(\sqrt{\dfrac{n-1}{n+1}} - 1 \right) (B_k \xi_k) \xi_k^\top$ і $r_{k+1} := \dfrac{n}{\sqrt{n^2 - 1}} r_k$.

Крок 4. Установлюємо $k := k + 1$ і переходимо до кроку 1.

На кожній ітерації алгоритму **emshor** обновлюється матриця B_k, яка пов'язана із заміною змінних $x = B_k y$, де $y = A_k x$ – образ точки x в перетвореному просторі змінних, $A_k = B_k^{-1}$. Очевидно, що обновлення B-матриці (див. крок 3) вимагає $O(n^2)$ операцій. Це обумовлено використанням оператора розтягу простору $R_\alpha(\xi) : \mathbb{R}^n \to \mathbb{R}^n$, що визначається як

$$R_\alpha(\xi) := I_n + (\alpha - 1) \xi \xi^\top, \tag{2}$$

де $\xi \in \mathbb{R}^n$, $\| \xi \| = 1$ – напрям розтягу, а $I_n \in \mathbb{R}^{n \times n}$ – одинична матриця. Властивості оператора розтягу простору детально досліджені в [7]. Вважаючи, що $\beta := 1/\alpha$ і позначаючи оператор «оберненого» розтягу як $R_\alpha^{-1}(\xi)$, маємо $R_\alpha^{-1}(\xi) = R_\beta(\xi)$ і $B_{k+1} = B_k R_\beta(\xi_k)$. Для обновлення A-матриць використовуємо співвідношення

$$A_{k+1} = R_\alpha(\xi_k) A_k, \tag{3}$$

яке випливає з ланцюжка рівностей

$$A_{k+1} = B_{k+1}^{-1} = \left(B_k R_\beta(\xi_k)\right)^{-1} = R_\beta^{-1}(\xi_k) B_k^{-1} = R_{1/\beta}(\xi_k) A_k = R_\alpha(\xi_k) A_k.$$

Теорема 1. Послідовність точок $\{x_k\}$, яку генерує алгоритм **emshor**, задовольняє нерівність

$$\left\| A_k (x_k - x^*) \right\| = \left\| B_k^{-1} (x_k - x^*) \right\| \le r_k, \quad k \in \mathbb{N}. \tag{4}$$

Детальне доведення теореми 1 наведено далі у пункті 1.2.

Множина точок x, яка задовольняє нерівність $\left\| B_k^{-1}(x_k - x) \right\| \le r_k$, є еліпсоїдом $\mathcal{E}_k := \left\{ x \,|\, \left\| B_k^{-1}(x_k - x) \right\| \le r_k \right\}$, що містить точку x^*. Еліпсоїд $\mathcal{E}_k$ має об'єм

$$\mathrm{vol}(\mathcal{E}_k) = \frac{v_0 r_k^n}{\det A_k}, \tag{5}$$

де v_0 – об'єм одиничної n-вимірної кулі, $\det A_k$ – визначник матриці A_k.

Отже, швидкість збіжності алгоритму **emshor** визначатиметься відношенням об'єму еліпсоїда $\mathcal{E}_{k+1}$, що локалізує точку x^* на $(k+1)$-й ітерації, до об'єму еліпсоїда $\mathcal{E}_k$, що локалізує x^* на k-й ітерації.

Теорема 2. Нехай x_k і x_{k+1} згенеровані алгоритмом **emshor**. Тоді $x^* \in \mathcal{E}_k$ для всіх $k \in \mathbb{N}$, а відношення об'ємів еліпсоїдів $\mathcal{E}_{k+1}$ і $\mathcal{E}_k$ не залежить від k і дорівнює

$$q_n := \frac{\mathrm{vol}(\mathcal{E}_{k+1})}{\mathrm{vol}(\mathcal{E}_k)} = \sqrt{\frac{n-1}{n+1}} \left(\frac{n}{\sqrt{n^2-1}} \right)^n < \exp\left\{ -\frac{1}{2(n+1)} \right\} < 1. \tag{6}$$

Якщо для ітерації k^* виконується нерівність $\left\| B_k^\top g(x_{k^*}) \right\| r_{k^*} \le \varepsilon$, то $f(x_{k^*}) - f^* \le \varepsilon$.

Доведення. Згідно з (3) маємо $A_{k+1} = R_\alpha(\xi_k) A_k$ і для визначників матриць A_{k+1} та A_k справедливо $\det A_{k+1} = \det R_\alpha(\xi_k) \det A_k$. Враховуючи (5) і те, що $\det R_\alpha(\xi) = \alpha$, знаходимо для (6) коефіцієнт зменшення об'єму

$$q_n = \frac{\mathrm{vol}(\mathcal{E}_{k+1})}{\mathrm{vol}(\mathcal{E}_k)} = \frac{v_0 r_{k+1}^n \det A_k}{v_0 r_k^n \det A_{k+1}} = \left(\frac{r_{k+1}}{r_k} \right)^n \frac{\det A_k}{\det R_\alpha(\xi_k) \det A_k} = \sqrt{\frac{n-1}{n+1}} \left(\frac{n}{\sqrt{n^2-1}} \right)^n.$$

Доведення нерівності в (6) базується на справедливості наступного співвідношення

$$\sqrt{\frac{n-1}{n+1}} \left(\frac{n}{\sqrt{n^2-1}} \right)^n = \sqrt{\frac{n-1}{n+1} \cdot \left(\frac{n^2}{n^2-1} \right)^n} = \sqrt{\left(\frac{n}{n+1} \right)^2 \cdot \left(\frac{n^2}{n^2-1} \right)^{n-1}} = \left(1 - \frac{1}{n+1} \right) \cdot \left(1 + \frac{1}{n^2-1} \right)^{(n-1)/2}.$$

Використовуючи елементарну нерівність $1 + t \le e^t$, яка є строгою для всіх $t \ne 0$, ми отримуємо

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

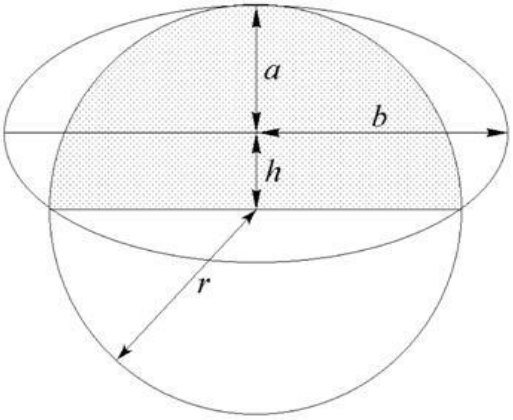
Той факт, що умова $r_k \left\| B_k^T g(x_{k^*}) \right\| \le \varepsilon$ дозволяє знайти точку x_{k^*}, для якої $f(x_{k^*}) - f^* \le \varepsilon$, випливає з нерівності

$$r_k \geq \left\| B_k^{-1}(x_k - x^*) \right\| \geq \left\langle B_k^{-1}(x_k - x^*), \frac{B_k^T g(x_k)}{\left\| B_k^T g(x_k) \right\|} \right\rangle = \frac{\left\langle B_k B_k^{-1}(x_k - x^*), g(x_k) \right\rangle}{\left\| B_k^T g(x_k) \right\|} = \frac{\left\langle x_k - x^*, g(x_k) \right\rangle}{\left\| B_k^T g(x_k) \right\|} \geq \frac{f(x_k) - f^*}{\left\| B_k^T g(x_k) \right\|},$$

яка, враховуючи нерівність (1), виконується для всіх $x_k \in \mathbb{R}^n$, згенерованих алгоритмом **emshor**.

Теорема 2 доведена.

Алгоритм **emshor** має просту геометричну інтерпретацію, яка базується на використанні еліпсоїда мінімального об'єму, що містить півкулю радіуса r в $\mathbb{R}^n$ ($n \geq 2$). Такий еліпсоїд має зжату форму в напрямі нормалі до гіперплощини, що визначає півкулю. Параметри еліпсоїда мінімального об'єму показані на рис. 1, де a – довжина малої півосі еліпсоїда, b – довжина великих півосей еліпсоїда (кількість таких півосей дорівнює $n-1$), h – відстань від центру кулі до центру еліпсоїда в напрямі його малої півосі.



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

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

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

РИС. 1. Еліпсоїд мінімального об'єму, що містить півкулю в $\mathbb{R}^n$

Об'єм еліпсоїда мінімального об'єму дорівнює $v_e = v_0 ab^{n-1}$, об'єм кулі дорівнює $v_b = v_0 r^n$, де v_0 – об'єм одиничної кулі в $\mathbb{R}^n$. Отже, коефіцієнт зменшення об'єму дорівнює

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

Щоб перетворити еліпсоїд мінімального об'єму в нову кулю, достатньо простір змінних розтягнути у напрямку малої півосі з коефіцієнтом $\alpha = b/a = \sqrt{(n+1)/(n-1)}$. Це реалізується за допомогою оператора $R_\alpha(\xi)$, де напрям ξ в формулі (2) збігається із напрямком малої півосі еліпсоїда.

Якщо $X = \mathbb{R}^n$ є початковим простором змінних, то у перетвореному просторі змінних $Y = R_\alpha(\xi)X$, ми отримаємо нову кулю радіуса b, яка містить розв'язок задачі, якщо його містила куля у початковому просторі. Повторюючи цю процедуру, але уже для нової кулі в перетвореному просторі отримуємо метод еліпсоїдів. Для цього на кроці 2, у перетвореному просторі $Y_k = B_k^{-1}X$ розраховується напрямок малої півосі еліпсоїда мінімального об'єму, що містить кулю радіуса r_k, і виконується перехід до його центру. Обчислений напрямок використовується для наступного роз-

тягування простору, яке здійснюється на кроці 3 за допомогою оновлення матриці B_{k+1}. В результаті розтягу простору отримаємо кулю з радіусом r_{k+1} в наступному перетвореному просторі $Y_{k+1} = B_{k+1}^{-1}X$.

1.2. Доведення теореми 1. Доведення проведемо за аналогією з доведенням, яке для методу еліпсоїдів було зроблено Н.З. Шором [4]. Для доведення будемо використовувати співвідношення

$$R_\alpha^T(\xi)R_\alpha(\xi) = R_{\alpha^2}(\xi), \tag{7}$$

яке випливає з властивостей (2) – оператора розтягу простору (див. [6], стор. 68–69).

Доведення теореми проведемо методом індукції по k . Для $k = 0$ нерівність (4) переходить у $\left\|B_0^{-1}(x_0 - x^*)\right\| \le r_0$ та виконується за припущенням. Припустимо, що нерівність (4) виконується для $k = \bar{k}$. Доведемо її справедливість для $k = \bar{k} + 1$.

Враховуючи співвідношення (7) і те, що з (3) випливає $A_{\bar{k}+1} = R_\alpha(\xi_{\bar{k}})A_{\bar{k}}$, маємо ланцюжок рівностей:

$$\| A_{\bar{k}+1}(x_{\bar{k}+1} - x^*)\|^2 = \left\langle A_{\bar{k}+1}(x_{\bar{k}+1} - x^*), A_{\bar{k}+1}(x_{\bar{k}+1} - x^*)\right\rangle = \left\langle R_\alpha(\xi_{\bar{k}})A_{\bar{k}}(x_{\bar{k}+1} - x^*), R_\alpha(\xi_{\bar{k}})A_{\bar{k}}(x_{\bar{k}+1} - x^*)\right\rangle =$$

$$= \left\langle A_{\bar{k}}(x_{\bar{k}+1} - x^*), R_\alpha^T(\xi_{\bar{k}})R_\alpha(\xi_{\bar{k}})A_{\bar{k}}(x_{\bar{k}+1} - x^*)\right\rangle = \left\langle A_{\bar{k}}(x_{\bar{k}+1} - x^*), R_{\alpha^2}(\xi_{\bar{k}})A_{\bar{k}}(x_{\bar{k}+1} - x^*)\right\rangle =$$

$$= \left\langle A_{\bar{k}}(x_{\bar{k}+1} - x^*), A_{\bar{k}}(x_{\bar{k}+1} - x^*)\right\rangle + (\alpha^2 - 1)\left\langle \xi_{\bar{k}}, A_{\bar{k}}(x_{\bar{k}+1} - x^*)\right\rangle^2 =$$

$$= \| A_{\bar{k}}(x_{\bar{k}+1} - x^*)\|^2 + (\alpha^2 - 1)\left\langle \xi_{\bar{k}}, A_{\bar{k}}(x_{\bar{k}+1} - x^*)\right\rangle^2,$$

який запишемо у вигляді співвідношення

$$\| A_{\bar{k}+1}(x_{\bar{k}+1} - x^*)\|^2 = \| A_{\bar{k}}(x_{\bar{k}+1} - x^*)\|^2 + (\alpha^2 - 1)\left\langle \xi_{\bar{k}}, A_{\bar{k}}(x_{\bar{k}+1} - x^*)\right\rangle^2. \tag{8}$$

Далі, розшифруємо обидва доданки у правій частині (8), для чого використаємо співвідношення

$$A_{\bar{k}}(x_{\bar{k}+1} - x^*) = A_{\bar{k}}(x_{\bar{k}} - x^*) - h_{\bar{k}}\xi_{\bar{k}}, \tag{9}$$

яке з урахуванням того, що $A_{\bar{k}} = B_{\bar{k}}^{-1}$ і чергова точка в алгоритмі 1 обчислюється за формулою з кроку 2, випливає з ланцюжка рівностей

$$A_{\bar{k}}(x_{\bar{k}+1} - x^*) = A_{\bar{k}}(x_{\bar{k}} - h_{\bar{k}}B_{\bar{k}}\xi_{\bar{k}} - x^*) = A_{\bar{k}}(x_{\bar{k}} - x^*) - h_{\bar{k}}A_{\bar{k}}B_{\bar{k}}\xi_{\bar{k}} = A_{\bar{k}}(x_{\bar{k}} - x^*) - h_{\bar{k}}\xi_{\bar{k}}.$$

Перший доданок у правій частині (9) можна записати у вигляді рівності

$$\| A_{\bar{k}}(x_{\bar{k}+1} - x^*)\|^2 = \| A_{\bar{k}}(x_{\bar{k}} - x^*)\|^2 - 2h_{\bar{k}}\left\langle A_{\bar{k}}(x_{\bar{k}} - x^*), \xi_{\bar{k}}\right\rangle + h_{\bar{k}}^2, \tag{10}$$

яка з урахуванням (9) і того, що $\| \xi_{\bar{k}}\| = 1$, випливає з ланцюжка рівностей

$$\| A_{\bar{k}}(x_{\bar{k}+1} - x^*)\|^2 = \| A_{\bar{k}}(x_{\bar{k}} - x^*) - h_{\bar{k}}\xi_{\bar{k}}\|^2 = \left\langle A_{\bar{k}}(x_{\bar{k}} - x^*) - h_{\bar{k}}\xi_{\bar{k}}, A_{\bar{k}}(x_{\bar{k}} - x^*) - h_{\bar{k}}\xi_{\bar{k}}\right\rangle =$$

$$= \left\langle A_{\bar{k}}(x_{\bar{k}} - x^*), A_{\bar{k}}(x_{\bar{k}} - x^*)\right\rangle - 2h_{\bar{k}}\left\langle A_{\bar{k}}(x_{\bar{k}} - x^*), \xi_{\bar{k}}\right\rangle + h_{\bar{k}}^2\left\langle \xi_{\bar{k}}, \xi_{\bar{k}}\right\rangle =$$

$$= \| A_{\bar{k}}(x_{\bar{k}} - x^*)\|^2 - 2h_{\bar{k}}\left\langle A_{\bar{k}}(x_{\bar{k}} - x^*), \xi_{\bar{k}}\right\rangle + h_{\bar{k}}^2\| \xi_{\bar{k}}\|^2 = \| A_{\bar{k}}(x_{\bar{k}} - x^*)\|^2 - 2h_{\bar{k}}\left\langle A_{\bar{k}}(x_{\bar{k}} - x^*), \xi_{\bar{k}}\right\rangle + h_{\bar{k}}^2.$$

Враховуючи співвідношення (9), для квадрата скалярного добутку в другому доданку правої частини (8) маємо такий ланцюжок рівностей:

$$\left\langle A_{\bar{k}}(x_{\bar{k}+1}-x^*),\xi_{\bar{k}}\right\rangle^2=\left\langle A_{\bar{k}}(x_{\bar{k}}-x^*)-h_{\bar{k}}\xi_{\bar{k}},\xi_{\bar{k}}\right\rangle^2=\left(\left\langle A_{\bar{k}}(x_{\bar{k}}-x^*),\xi_{\bar{k}}\right\rangle-h_{\bar{k}}\left\langle \xi_{\bar{k}},\xi_{\bar{k}}\right\rangle\right)^2=$$

$$=\left(\left\langle A_{\bar{k}}(x_{\bar{k}}-x^*),\xi_{\bar{k}}\right\rangle-h_{\bar{k}}\|\xi_{\bar{k}}\|^2\right)^2=\left(\left\langle A_{\bar{k}}(x_{\bar{k}}-x^*),\xi_{\bar{k}}\right\rangle-h_{\bar{k}}\right)^2=$$

$$=\left\langle A_{\bar{k}}(x_{\bar{k}}-x^*),\xi_{\bar{k}}\right\rangle^2-2h_{\bar{k}}\left\langle A_{\bar{k}}(x_{\bar{k}}-x^*),\xi_{\bar{k}}\right\rangle+h_{\bar{k}}^2.$$

Отже, квадрат зазначеного скалярного добутку можна записати у вигляді

$$\left\langle A_{\bar{k}}(x_{\bar{k}+1}-x^*),\xi_{\bar{k}}\right\rangle^2=\left\langle A_{\bar{k}}(x_{\bar{k}}-x^*),\xi_{\bar{k}}\right\rangle^2-2h_{\bar{k}}\left\langle A_{\bar{k}}(x_{\bar{k}}-x^*),\xi_{\bar{k}}\right\rangle+h_{\bar{k}}^2. \quad (11)$$

Підставляючи (10) і (11) у (8) маємо

$$A_{\bar{k}+1}(x_{\bar{k}+1}-x^*)\|^2=\|A_{\bar{k}}(x_{\bar{k}+1}-x^*)\|^2+(\alpha^2-1)\left\langle \xi_{\bar{k}},A_{\bar{k}}(x_{\bar{k}+1}-x^*)\right\rangle^2=\|A_{\bar{k}}(x_{\bar{k}}-x^*)\|^2-$$

$$-2h_{\bar{k}}\left\langle A_{\bar{k}}(x_{\bar{k}}-x^*),\xi_{\bar{k}}\right\rangle+h_{\bar{k}}^2+(\alpha^2-1)\left(\left\langle A_{\bar{k}}(x_{\bar{k}}-x^*),\xi_{\bar{k}}\right\rangle^2-2h_{\bar{k}}\left\langle A_{\bar{k}}(x_{\bar{k}}-x^*),\xi_{\bar{k}}\right\rangle+h_{\bar{k}}^2\right)=$$

$$=\|A_{\bar{k}}(x_{\bar{k}}-x^*)\|^2-2\alpha^2h_{\bar{k}}\left\langle A_{\bar{k}}(x_{\bar{k}}-x^*),\xi_{\bar{k}}\right\rangle+(\alpha^2-1)\left\langle A_{\bar{k}}(x_{\bar{k}}-x^*),\xi_{\bar{k}}\right\rangle^2+\alpha^2h_{\bar{k}}^2=$$

$$=\|A_{\bar{k}}(x_{\bar{k}}-x^*)\|^2-\left\langle A_{\bar{k}}(x_{\bar{k}}-x^*),\xi_{\bar{k}}\right\rangle\left(2\alpha^2h_{\bar{k}}-(\alpha^2-1)\left\langle A_{\bar{k}}(x_{\bar{k}}-x^*),\xi_{\bar{k}}\right\rangle\right)+\alpha^2h_{\bar{k}}^2,$$

звідки з урахуванням $h_k=\dfrac{1}{n+1}r_k$ і $\alpha=\sqrt{\dfrac{n+1}{n-1}}$ маємо

$$\|A_{\bar{k}+1}(x_{\bar{k}+1}-x^*)\|^2=\|A_{\bar{k}}(x_{\bar{k}}-x^*)\|^2-\frac{2}{n-1}\left\langle A_{\bar{k}}(x_{\bar{k}}-x^*),\xi_{\bar{k}}\right\rangle\left(r_{\bar{k}}-\left\langle A_{\bar{k}}(x_{\bar{k}}-x^*),\xi_{\bar{k}}\right\rangle\right)+\frac{1}{n^2-1}r_{\bar{k}}^2. \quad (12)$$

Далі, для оцінювання знака добутку

$$\left\langle A_{\bar{k}}(x_{\bar{k}}-x^*),\xi_{\bar{k}}\right\rangle\left(r_{\bar{k}}-\left\langle A_{\bar{k}}(x_{\bar{k}}-x^*),\xi_{\bar{k}}\right\rangle\right),$$

що входить до правої частини співвідношення (12), оцінимо знаки обох його співмножників. Перший співмножник буде невід'ємним. Оскільки маємо $\left\langle x-x^*,g(x)\right\rangle\geq 0$ для всіх $x\in\mathbb{R}^n$, його легко оцінити так:

$$\left\langle A_{\bar{k}}(x_{\bar{k}}-x^*),\xi_{\bar{k}}\right\rangle=\left\langle A_{\bar{k}}(x_{\bar{k}}-x^*),\frac{B_{\bar{k}}^Tg(x_{\bar{k}})}{\|B_{\bar{k}}^Tg(x_{\bar{k}})\|}\right\rangle=\frac{1}{\|B_{\bar{k}}^Tg(x_{\bar{k}})\|}\left\langle A_{\bar{k}}(x_{\bar{k}}-x^*),B_{\bar{k}}^Tg(x_{\bar{k}})\right\rangle=$$

$$=\frac{1}{\|B_{\bar{k}}^Tg(x_{\bar{k}})\|}\left\langle B_{\bar{k}}A_{\bar{k}}(x_{\bar{k}}-x^*),g(x_{\bar{k}})\right\rangle=\frac{1}{\|B_{\bar{k}}^Tg(x_{\bar{k}})\|}\left\langle x_{\bar{k}}-x^*,g(x_{\bar{k}})\right\rangle\geq 0.$$

З урахуванням того, що

$$0\leq\left\langle A_{\bar{k}}(x_{\bar{k}}-x^*),\xi_{\bar{k}}\right\rangle\leq\left\|A_{\bar{k}}(x_{\bar{k}}-x^*)\right\|\leq r_{\bar{k}},$$

другий співмножник оцінюється так:

$$r_{\bar{k}}-\left\langle A_{\bar{k}}(x_{\bar{k}}-x^*),\xi_{\bar{k}}\right\rangle\geq 0.$$

З невід'ємності обох співмножників випливає, що

$$\frac{2}{n-1}\left\langle A_{\bar{k}}(x_{\bar{k}}-x^*),\xi_{\bar{k}}\right\rangle\left(r_{\bar{k}}-\left\langle A_{\bar{k}}(x_{\bar{k}}-x^*),\xi_{\bar{k}}\right\rangle\right)\geq 0. \tag{13}$$

Далі, врахувавши (13) і те, що $\| A_{\bar{k}}(x_{\bar{k}}-x^*)\|\leq r_{\bar{k}}$, співвідношення (12) перепишемо у вигляді

$$\| A_{\bar{k}+1}(x_{\bar{k}+1}-x^*)\|^2\leq\| A_{\bar{k}}(x_{\bar{k}}-x^*)\|^2+\frac{1}{n^2-1}r_{\bar{k}}^2\leq r_{\bar{k}}^2+\frac{1}{n^2-1}r_{\bar{k}}^2=\left(\frac{n^2}{n^2-1}\right)r_{\bar{k}}^2,$$

звідки маємо нерівність

$$\| A_{\bar{k}+1}(x_{\bar{k}+1}-x^*)\|^2=\| B_{\bar{k}+1}^{-1}(x_{\bar{k}+1}-x^*)\|^2\leq\left(\frac{n}{\sqrt{n^2-1}}\right)^2 r_{\bar{k}}^2=r_{\bar{k}+1}^2,$$

з якої випливає справедливість нерівності (4) для $k=\bar{k}+1$.

Теорема доведена.

1.3. Octave програма emshor. Алгоритм **emshor** реалізований за допомогою однойменної програми на мові Octave [7–9]. Вона використовує octave-функцію вигляду **function [f, g] = calcfg (x)**, яка обчислює значення функції $f=f(x)$ та її субградієнт $g=g(x)$ в точці x. Ця функція підготовлюється користувачем і може мати довільне ім'я, яке підтримує синтаксис Octave. Код програми **emshor** з короткими коментарями наведений далі.

```
# Вхідні параметри:
# calcfg - ім'я функції для обчислення f та g
# x0 - початкова точка, x0(1:n)
# rad - радіус кулі, що містить точку мінімуму
# eps, maxitn - параметри зупинки (точність, 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 зупинки (1 = epsf, 4 = maxitn)
function [x,f,itn,ist] = emshor(calcfg,x0,r0,eps,        #row01
                                maxitn,intp);
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) * 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
```

В програмі ітераційний процес виконується в циклі **for** (рядки 04–14), де рядки 05–09 виконують крок 1 алгоритму **emshor**, рядки 10–11 – крок 2, а рядки 12–13 – крок 3. Після кожних intp ітерацій в циклі **for** виводяться проміжні результати (див. рядки 06–08). Програма **emshor** закінчується виконанням одної з двох умов:

1) знайдена точка x_ε^* – така, що $f(x_\varepsilon^*) \le f^* + \varepsilon$ (**ist = 1,** див. рядок 09);

2) досягається **maxitn** – максимальна кількість ітерацій (**ist = 4,** див. рядки 04 і 15).

Програму **emshor** можна успішно застосовувати для мінімізації опуклих функцій, якщо кількість змінних $n = 2 \div 30$, що підтверджують результати обчислювальних експериментів для гладких та негладких функцій [8]. Для зменшення в 10 разів об'єму еліпсоїда, в якому локалізовано точку x^*, потрібно зробити K ітерацій, де $K = -\ln 10 / \ln q_n \approx (2 \ln 10) n \approx 4.6n$, тобто, щоб на порядок покращити відхилення знайденого рекордного значення функції $f(x)$ від її оптимального значення f^* потрібно зробити $4.6n^2$ ітерацій. Якщо $n = 30$, то використання досить малих значень ε вимагає великої кількості ітерацій, наприклад, для використання $\varepsilon = 10^{-20}$ максимальна кількість ітерацій оцінюється величиною $92n^2 = 92 \times 30 \times 30 = 82800$. У цьому випадку значення **maxitn** потрібно вибирати не меншим, ніж **100000** ітерацій.

2. Метод еліпсоїдів для задачі Сильвестра

2.1. Задача Сильвестра. Задача про найменшу обмежувальну сферу – це задача про найменшу гіперкулю, яка містить усі точки заданої множини. В n-вимірному евклідовому просторі $\mathbb{R}^n$ задана множина точок $A_m = \{a_j,\ j = 1, \ldots, m\}$, де $a_j \in \mathbb{R}^n$. Потрібно побудувати гіперкулю мінімального радіуса, яка охоплює всі точки множини A_m. Цю задачу умовимось називати задачею Сильвестра, так як вперше для площини $\mathbb{R}^2$ таку задачу сформулював англійський математик Джеймс Джозеф Сильвестр [1].

На рис. 2 показано декілька розв'язків задачі Сильвестра для $\mathbb{R}^2$, які наведені в [10]. Найменше обмежувальне коло для множини точок на площині можна визначити максимум за трьома точками з цієї множини, які лежать на границі кола. Якщо коло визначається лише двома точками, то хорда, що з'єднує ці точки, є діаметром найменшого обмежувального кола. Якщо коло визначається трьома точками, то трикутник з вершинами в цих точках не може бути тупокутним.

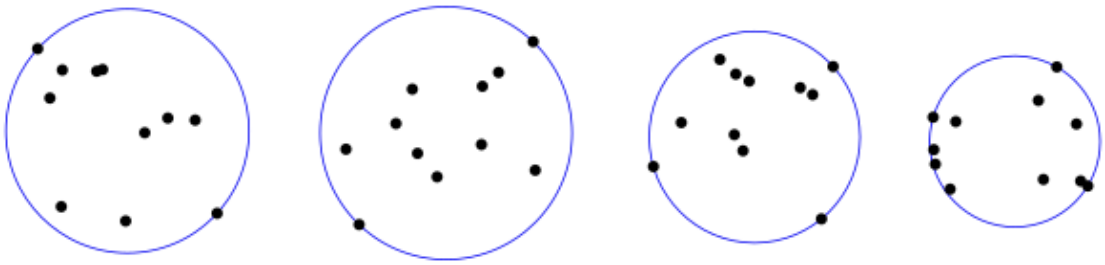


РИС. 2. Найменші обмежувальні кола, що охоплюють чотири множини точок в $\mathbb{R}^2$

Задача Сильвестра може бути сформульована як задача опуклого програмування [11]: знайти

$$f_1^* = f_1(x_S^*) = \min_{x \in R^n} \left\{ f_1(x) = \max_{j=1,\ldots,m} \left\| x - a_j \right\|^2 \right\}, \tag{14}$$

за обмежень

$$\left\| x - x_0 \right\| \le r_0, \tag{15}$$

де $x \in \mathbb{R}^n$ – n-вимірний вектор невідомих координат центру гіперкулі, $x_0 = \frac{1}{m}\sum_{j=1}^{m} a_j$ – центр кулі радіуса $r_0 = \sqrt{\max_{j=1,\ldots,m} \left\| x_0 - a_j \right\|^2}$, в якій локалізовано x_S^* – розв'язок задачі (14), (15). Оптимальне значення $f_1^* = f_1(x_S^*)$ визначає R_S^* – мінімальний радіус гіперкулі, яка охоплює всі точки множини A_m, він буде рівним $\sqrt{f_1^*}$.

Для розв'язання задачі (14), (15) може бути застосований алгоритм **emshor**, оскільки вона є задачею мінімізації опуклої кусково-квадратичної функції $f_1(x)$, для якої точка мінімуму x_S^* знаходиться всередині n-вимірної кулі радіуса $r_0 = \max_{j=1,\ldots,m} \left\| x_0 - a_j \right\|$ з центром у точці $x_0 = \frac{1}{m}\sum_{j=1}^{m} a_j$. Для знаходження точки x_S^* далі наведемо алгоритм **sylvester1**, який побудований за допомогою алгоритму **emshor**, де стартова точка x_0 та радіус r_0 вибираються такими, як вказано вище.

2.2. Алгоритм sylvester1. Вхідним параметром алгоритму є величина $\varepsilon_1 > 0$ – точність, з якою необхідно знайти $f_1^* = f_1(x_S^*)$.

Ініціалізація. Розглянемо $n \times n$-матрицю B і покладемо $B_0 := I_n$, де I_n – одинична $n \times n$-матриця. Перейдемо до першої ітерації зі значеннями $x_0 = \frac{1}{m}\sum_{j=1}^{m} a_j$, $r_0 = \max_{j=1,\ldots,m} \left\| x_0 - a_j \right\|$ і B_0.

Нехай на k-й ітерації знайдені значення $x_k \in R^n$, r_k, B_k. Перехід до $(k+1)$-ї ітерації полягає у такій послідовності дій.

Крок 1. Обчислимо $f_1(x_k)$ та $g_{f_1}(x_k)$ у точці x_k за формулою $g_{f_1}(x_k) = 2(x_k - a_{j^*})$, де j^* таке, що $\left\| x_k - a_{j^*} \right\|^2 = f_1(x_k)$. Якщо $r_k \left\| B_k^T g_{f_1}(x_k) \right\| \le \varepsilon_1$, то "ЗУПИНКА: $k^* = k$ і $x_S^* = x_k$". Інакше переходимо до кроку 2.

Крок 2. Покладемо $\xi_k := \dfrac{B_k^T g_{f_1}(x_k)}{\left\| B_k^T g_{f_1}(x_k) \right\|}$.

Крок 3. Обчислимо чергову точку

$$x_{k+1} := x_k - h_k B_k \xi_k, \quad \text{де} \quad h_k = \frac{1}{n+1} r_k.$$

Крок 4. Обчислимо

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

Крок 5. Переходимо до $(k+1)$-ї ітерації алгоритму **sylvester1** зі значеннями x_{k+1}, r_{k+1}, B_{k+1}.

Збіжність алгоритму забезпечує наступна теорема.

Теорема 3. Послідовність точок $\{x_k\}_{k=0}^{k^*}$, яку генерує алгоритм **sylvester1**, задовольняє нерівність

$$\left\|B_k^{-1}(x_k - x_S^*)\right\| \le r_k, \quad k = 0,1,2,\ldots,k^*. \tag{16}$$

На кожній ітерації $k>0$ величина зменшення об'єму еліпсоїда $\mathcal{E}_k = \left\{x \in \mathbb{R}^n : \left\|B_k^{-1}(x_k - x)\right\| \le r_k\right\}$, локалізуючого x_S^*, – величина стала і рівна

$$q_n := \frac{vol(\mathcal{E}_{k+1})}{vol(\mathcal{E}_k)} = \sqrt{\frac{n-1}{n+1}}\left(\frac{n}{\sqrt{n^2-1}}\right)^n < \exp\left\{-\frac{1}{2(n+1)}\right\} < 1. \tag{17}$$

Якщо для ітерації k^* алгоритму **sylvester1** виконується нерівність $\left\|B_k^{\top} g(x_{k^*})\right\| r_{k^*} \le \varepsilon_1$, то

$$f_1(x_{k^*}) - f_1^* \le \varepsilon_1. \tag{18}$$

Теорема 3 включає три нерівності, де перша нерівність (16) доводиться аналогічно доведенню нерівності (4) теореми 1, а друга (17) та третя (18) доводяться аналогічно доведенню нерівностей у теоремі 2.

Якщо $n = 2 \div 30$, то алгоритм **sylvester1** можна успішно застосовувати для знаходження x_S^*. Продемонструємо це за допомогою програми **emshor**, де значення опуклої кусково-квадратичної функції $f_1(x)$ та її субградієнта в точці $x \in R^n$ обчислюються за допомогою octave-функції **`fgf1(x)`**, яка має такий вигляд:

```
function [f,g] = fgf1(x)
global A m n
temp = ones(m,1)*x' - A;
temp1 = sum(temp'.*temp');
[tmax imax] = max(temp1);
f = tmax;
g = 2.0*temp(imax,:)';
end
```

Тут множина точок $A_m = \{a_j,\ j = 1,\ldots,m\}$, де $a_j \in \mathbb{R}^n$, передається в функцію **`fgf1(x)`** за допомогою глобальних параметрів **`A`**, **`m`** та **`n`**, де **`A`** – $m \times n$-матриця, яка має **`m`** рядків, що містять **`n`**-вимірні точки.

2.3. Обчислювальний експеримент. Для перевірки роботи алгоритму **sylvester1** будемо використовувати тестовий приклад з $m = n+1$ точками, де n точок є вершинами n вимірного симп-

лекса $\sum_{i=1}^{n} x_i = 1,\ x_i \ge 0,\ i = 1,\ldots,n$, а $n+1$ точка співпадає з початком координат. Для тестового прикладу матриця A, x_S^* та R_S^* мають такий вигляд:

$$A(n+1,n) = \begin{pmatrix} 1 & 0 & \vdots & 0 & 0 \\ 0 & 1 & \vdots & 0 & 0 \\ \cdots & \cdots & \ddots & \cdots & \cdots \\ 0 & 0 & \vdots & 1 & 0 \\ 0 & 0 & \vdots & 0 & 1 \\ 0 & 0 & \cdots & 0 & 0 \end{pmatrix}, \quad x_S^*(n) = \frac{1}{n}\begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{pmatrix}, \quad R_S^* = \sqrt{1-\frac{1}{n}}, \quad n = 1,2,\ldots,30,\ldots. \qquad (19)$$

При цьому оптимальне значення цільової функції в задачі (14) – (15) є рівним $f_1^* = \left(R_S^*\right)^2 = 1 - 1/n$.

Тестовий приклад за формулою (19) для перевірки роботи алгоритму **sylvester1** при різних значеннях $\varepsilon_1 \in \left[10^{-30} \div 10^{-2}\right]$ та $n = 30$ реалізує наведений далі Octave-код.

```
printf(" Тест 1: мінімізація функції f1 програмою emshor\n\n");
global A m n
printf(" Підготовка тестового прикладу\n");
n = 30; m = n+1;
A = [eye(n); zeros(1,n)];
xstar = ones(n,1)/n;
n, m, A, x_star = xstar',
printf("\n Вибираємо x0, r0, maxitn та intp\n");
x0 = sum(A)'/m; [f0, g1] = fgf1(x0);
r0 = sqrt(f0); maxitn = 150000; intp=150001;
x_0 = x0', r0, maxitn, intp,
printf("\n Розв'язуємо задачу (14)-(15) для 15 різних значень eps1\n");
eps1 = 1.e-2; ntest = 15;
for (in = 1:ntest)
  tstart = time();
  [xr1,fr1,itn1,ist1] = emshor(@fgf1,x0,r0,eps1,maxitn,intp);
  time1 = time()-tstart;
     printf("eps1  itn1  time1  f1  dx1  %7.1e  %6d  %5.1f %20.17f %9.6e\n",
        eps1,itn1,time1,fr1,norm(xr1-xstar));
  eps1 = eps1/100.0;
endfor
printf("\n Друкуємо розв'язок (центр кулі мінімального радіусу)\n");
x_center = xr1',
```

Обчислювальні експерименти проводилися на персональному комп'ютері із процесором AMD Ryzen 5 4500U 2.38 GHz, 16 GB у системі Windows 10 за допомогою GNU Octave версії 6.2.0. Результати розрахунку за вищеприведеним кодом для 15 різних значень ε_1 наведені в табл. 1. Тут k^* – кількість ітерацій для пошуку точки x_{k^*}, t_1 – час в секундах, затрачений на розв'язання задачі (14), (15) при конкретному значенні ε_1, $f_1(x_{k^*})$ – значення функції $f_1(x)$ на ітерації k^*.

ТАБЛИЦЯ 1. Результати роботи алгоритму **sylvester1** для тесту (19) при $n=30$ та $\varepsilon_1 \in \left[10^{-30} \div 10^{-2}\right]$

ε_1	k^*	t_1	$f_1(x_{k^*})$	$\left\|x_{k^*} - x_S^*\right\|$
1.0e-02	9248	0.6	0.96691514094205810	1.419648e-03
1.0e-04	17344	1.1	0.96666675076144826	4.038524e-05
1.0e-06	25522	1.6	0.96666669634550528	8.408335e-06
1.0e-08	33675	2.1	0.96666666694991410	1.177884e-06
1.0e-10	41800	2.5	0.96666666666926282	5.148086e-08
1.0e-12	49954	3.1	0.9666666666666**6**778	4.682847e-09
1.0e-14	58115	3.5	0.9666666666666**6**790	1.519273e-09
1.0e-16	65514	4.2	0.9666666666666**6**712	1.009356e-11
1.0e-18	67685	4.4	0.9666666666666**6**712	3.935437e-12
1.0e-20	69468	4.5	0.9666666666666**6**701	7.548093e-12
1.0e-22	78832	5.1	0.9666666666666**6**712	5.063893e-13
1.0e-24	83439	5.4	0.9666666666666**6**690	4.869797e-13
1.0e-26	87876	5.8	0.9666666666666**6**690	5.051377e-13
1.0e-28	95098	6.2	0.9666666666666**6**701	5.061942e-13
1.0e-30	95921	6.3	0.9666666666666**6**712	5.064286e-13

З табл. 1 видно, що алгоритм **sylvester1** досить швидко знаходить досить точні наближення до x_S^* – єдиної точки мінімуму функції $f_1(x)$. Про це свідчать результати з останньої колонки $\left\|x_{k^*} - x_S^*\right\|$. Так, наприклад, для того, щоб знайти наближення до точки x_S^* з точністю 10^{-8} досить зробити 41800 ітерацій, для чого потрібно всього 2.5 секунди, а для точності 10^{-12} досить зробити 67685 ітерацій, затративши всього 4.4 секунди. З передостанньої колонки таблиці видно, що для знаходження значення цільової функції задачі (14), (15) алгоритм **sylvester1** демонструє високу точність, яку характеризує 14–15 значущих цифр. З табл. 1 видно, що при використанні $\varepsilon_1 \in \left[10^{-30} \div 10^{-12}\right]$ досягнуті значення цільової функції не зменшуються монотонно. Це характерно для методу еліпсоїдів, який забезпечує монотонне зменшення об'єму еліпсоїда, що локалізує точку x_S^*, але не забезпечує монотонного зменшення значення функції, що мінімізується.

Зауважимо, що для розв'язання задачі (14), (15) при $n=30$ значення **maxitn** було вибрано рівним 150000 ітерацій, хоча максимальна кількість ітерацій не перевищила 100000. Це відповідає швидкості збіжності алгоритму **emshor** за теоремою 2, оскільки для зменшення в 10 разів об'єму еліпсоїда, в якому локалізована точка x_S^*, потрібно зробити $K = -\ln 10/\ln q_n \approx (2\ln 10)n \approx 4.6n$ ітерацій. Якщо $n=30$, то при використанні дуже малих значень $\varepsilon_1 = 10^{-30}$ максимальна кількість ітерацій оцінюється величиною $138n^2 = 138\times 30\times 30 = 124200$. Для цього випадку значення **maxitn** потрібно вибирати не меншим, ніж **125000** ітерацій. Якщо кількість невідомих n буде меншою, то для $\varepsilon_1 = 10^{-30}$ буде потрібна менша кількість ітерацій. Так, наприклад, при $n=20$ кількість ітерацій не перевищує 51000, при цьому затрачено 3.2 секунди, при $n=10$ – не більше 13000 ітерацій за 0.8 секунд, при $n=5$ – не більше 3000 ітерацій за 0.2 секунди.

3. Метод еліпсоїдів для узагальненої задачі Сильвестра

3.1. Узагальнена задача Сильвестра. Задачу Сильвестра можна розширити на випадок, коли множина точок замінюється множиною куль із заданими координатами центрів та радіусів. Тоді задача про найменшу обмежувальну сферу буде задачею про найменшу гіперкулю, яка містить усі кулі заданої множини. В n-вимірному евклідовому просторі $\mathbb{R}^n$ задана множина куль $S_m = \{S_j,\ j=1,\ldots,m\}$ з їх центрами $A_m = \{a_j,\ j=1,\ldots,m\}$, де $a_j \in \mathbb{R}^n$, та радіусами $r_j,\ j=1,\ldots,m$. Потрібно побудувати гіперкулю мінімального радіуса, яка охоплює всі кулі множини S_m. Цю задачу будемо називати узагальненою задачею Сильвестра. Якщо $r \equiv 0$, то узагальнена задача Сильвестра переходить у звичайну задачу Сильвестра, яка описана в підрозділі 2.1.

Для чотирьох прикладів на рис. 3 показано розв'язки узагальненої задачі Сильвестра для $\mathbb{R}^2$. На першому (крайній зліва) рисунку зображено найменше обмежувальне коло для множини п'яти кругів різних радіусів, на другому – найменше обмежувальне коло для множини трьох кругів та двох точок (круги з нульовими радіусами). На третьому рисунку зображено найменше обмежувальне коло для множини трьох однакових кругів, центри яких відповідають тестовому прикладу (19). На четвертому рисунку представлено обмежувальне коло для двох нерівних кругів, що перетинаються, та трьох точок, дві з яких належать обмежувальному колу.

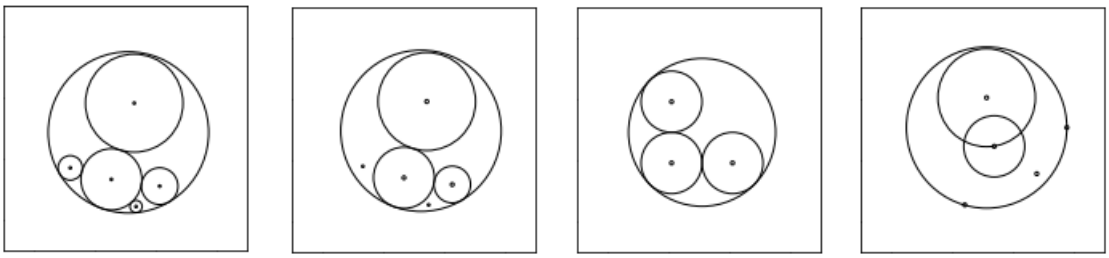


РИС. 3. Найменші обмежувальні кола, що охоплюють множини кругів в $\mathbb{R}^2$

Узагальнена задача Сильвестра може бути сформульована як така задача опуклого програмування: знайти

$$f_2^* = f_2(x^*) = \min_{x \in R^n} \left\{ f_2(x) = \max_{j=1,\ldots,m} \left\{ \left\| x - a_j \right\| + r_j \right\} \right\}, \tag{20}$$

за обмежень

$$\left\| x - x_0 \right\| \le r_0, \tag{21}$$

де $x \in \mathbb{R}^n$ – n-вимірний вектор невідомих координат центру гіперкулі, $x_0 = \frac{1}{m}\sum_{j=1}^{m} a_j$ – центр кулі радіуса $r_0 = \sqrt{\max_{j=1,\ldots,m} \left\{ \left\| x_0 - a_j \right\| + r_j \right\}}$, в якій локалізовано x_U^* – розв'язок задачі (20), (21). Оптимальне значення $f_2^* = f_2(x_U^*)$ є рівним R_U^* −мінімальному радіусу гіперкулі, яка охоплює всі кулі мно-

жини S_m. Якщо $r \equiv 0$, то задача (20), (21) переходить в задачу Сильвестра (14), (15), де достатньо опуклу цільову функцію $f_1(x) = \max\limits_{j=1,\ldots,m} \left\| x - a_j \right\|^2$ замінити на опуклу функцію $f(x) = \max\limits_{j=1,\ldots,m} \left\| x - a_j \right\|$.

Для розв'язання задачі (20), (21) може бути застосований алгоритм **emshor**, оскільки вона є задачею мінімізації опуклої функції $f_2(x)$, для якої точка мінімуму x_U^* знаходиться всередині n-вимірної кулі радіуса $r_0 = \sqrt{\max\limits_{j=1,\ldots,m} \left\{ \left\| x_0 - a_j \right\| + r_j \right\}}$ з центром у точці $x_0 = \frac{1}{m} \sum\limits_{j=1}^{m} a_j$. Для знаходження точки x_U^* далі наведемо алгоритм **sylvester2**, який побудований за допомогою алгоритму **emshor**, де стартова точка x_0 та радіус r_0 вибирається такими, як вказано вище.

3.2. Алгоритм sylvester2. Вхідним параметром алгоритму є величина $\varepsilon_2 > 0$ – точність, з якою необхідно знайти $f_2^* = f_2(x_U^*)$.

Ініціалізація. Розглянемо $n \times n$-матрицю B і покладемо $B_0 := I_n$, де I_n – одинична $n \times n$-матриця. Перейдемо до першої ітерації зі значеннями $x_0 = \frac{1}{m} \sum\limits_{j=1}^{m} a_j$, $r_0 = \sqrt{\max\limits_{j=1,\ldots,m} \left\{ \left\| x_0 - a_j \right\| + r_j \right\}}$ і B_0.

Нехай на k-й ітерації знайдені значення $x_k \in R^n$, r_k, B_k. Перехід до $(k+1)$-ї ітерації полягає у такій послідовності дій.

Крок 1. Обчислимо $f_2(x_k)$ та $g_{f_2}(x_k)$ у точці x_k за формулою $g_{f_2}(x_k) = (x_k - a_{j^*}) \Big/ \left\| x_k - a_{j^*} \right\|$, де j^* таке, що $\left\| x_k - a_{j^*} \right\| + r_{j^*} = f_2(x_k)$. Якщо $r_k \left\| B_k^T g_{f_2}(x_k) \right\| \le \varepsilon_2$, то "ЗУПИНКА: $k^* = k$ і $x_U^* = x_k$". Інакше переходимо до кроку 2.

Крок 2. Покладемо $\xi_k := \dfrac{B_k^T g_{f_2}(x_k)}{\left\| B_k^T g_{f_2}(x_k) \right\|}$.

Крок 3. Обчислимо чергову точку

$$x_{k+1} := x_k - h_k B_k \xi_k, \quad \text{де} \quad h_k = \frac{1}{n+1} r_k.$$

Крок 4. Обчислимо

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

Крок 5. Переходимо до $(k+1)$-ї ітерації алгоритму **sylvester2** зі значеннями x_{k+1}, r_{k+1}, B_{k+1}.

Збіжність алгоритму забезпечує наступна теорема.

Теорема 4. Послідовність точок $\{x_k\}_{k=0}^{k^*}$, яку генерує алгоритм **sylvester2**, задовольняє нерівність

$$\left\| B_k^{-1} (x_k - x_U^*) \right\| \le r_k, \quad k = 0,1,2,\ldots,k^*. \tag{22}$$

На кожній ітерації $k>0$ величина зменшення об'єму еліпсоїда $\mathcal{E}_k=\left\{x\in\mathbb{R}^n:\left\|B_k^{-1}(x_k-x)\right\|\le r_k\right\}$, який локалізує точку x_U^*, є величиною сталою і рівною

$$q_n:=\frac{vol(\mathcal{E}_{k+1})}{vol(\mathcal{E}_k)}=\sqrt{\frac{n-1}{n+1}}\left(\frac{n}{\sqrt{n^2-1}}\right)^n<\exp\left\{-\frac{1}{2(n+1)}\right\}<1. \tag{23}$$

Якщо для ітерації k^* алгоритму **sylvester2** виконується нерівність $\left\|B_k^{\top}g(x_{k^*})\right\|r_{k^*}\le\varepsilon_2$, то

$$f_2(x_{k^*})-f_2^*\le\varepsilon_2. \tag{24}$$

Зауважимо, що нерівність (22) доводиться аналогічно доведенню нерівності (4) теореми 1, а нерівності (23) та третя (24) доводяться аналогічно доведенню нерівностей у теоремі 2.

Якщо $n=2\div30$, то алгоритм **sylvester2** можна успішно застосовувати для знаходження x_U^*. Продемонструємо це за допомогою програми **emshor**, де значення функції $f_2(x)$ та її субградієнта в точці $x\in R^n$ обчислюються за допомогою octave-функції **fgf2(x)**, яка має такий вигляд:

```
function [f,g] = fgf2(x)
global A r m n
temp = ones(m,1)*x' - A;
temp1 = sum(temp'.*temp');
temp1a = sqrt(temp1);
temp2 = temp1a + r';
[tmax imax] = max(temp2);
f = tmax;
g = temp(imax,:)'/temp1a(1,imax);
end
```

Тут центри куль $A_m=\left\{a_j,\ j=1,\ldots,m\right\}$, де $a_j\in\mathbb{R}^n$, та їх радіуси $r_j,\ j=1,\ldots,m$ передаються в функцію **fgf2(x)** за допомогою глобальних параметрів **A, r, m** та **n**, де **A** $-m\times n$-матриця, яка має **m** рядків, що містять **n**-вимірні центри куль, а **r** – **m**-вимірний вектор, що містить радіуси куль.

3.3. Обчислювальний експеримент. Для перевірки роботи алгоритму **sylvester2** будемо використовувати тестовий приклад (19), де його $m=n+1$ точок будуть центрами куль, а радіуси куль виберемо однаковими та рівними $1/2$. Для модифікованого тестового прикладу матриця A, вектор r, x_U^* та R_U^* мають такий вигляд:

$$A(n+1,n)=\begin{pmatrix}1&0&\vdots&0&0\\0&1&\vdots&0&0\\\cdots&\cdots&\ddots&\cdots&\cdots\\0&0&\vdots&1&0\\0&0&\vdots&0&1\\0&0&\cdots&0&0\end{pmatrix},\ r(n)=\frac{1}{2}\begin{pmatrix}1\\1\\\vdots\\1\\1\end{pmatrix}\ x_U^*(n)=\frac{1}{n}\begin{pmatrix}1\\1\\\vdots\\1\\1\end{pmatrix},\ R_U^*=\frac{1}{2}+\sqrt{1-\frac{1}{n}},\ n=1,2,\ldots,30,\ldots. \tag{25}$$

Оптимальне значення цільової функції у задачі (20), (21) є рівним $f_2^* = R_U^* = \frac{1}{2} + \sqrt{1 - \frac{1}{n}}$. Якщо $r \equiv 0$, то $f_2^* = R_S^* = \sqrt{1 - \frac{1}{n}}$ співпадає з мінімальним радіусом гіперкулі, яка охоплює всі точки множини A_m, для задачі Сильвестра.

Тестовий приклад за формулою (25) для перевірки роботи алгоритму **sylvester2** при різних ε_1 та $n = 30$ реалізує наведений далі Octave-код.

```
printf(" Тест 2: мінімізація функції f2 програмою emshor\n\n");
global A r m n
printf(" Підготовка тестового прикладу\n");
n = 30; m = n+1;
A = [eye(n); zeros(1,n)];
r = 0.5*ones(m,1);
xstar = ones(n,1)/n;
n, m, A, x_star = xstar',
printf("\n  Вибираємо x0, r0, maxitn та intp\n");
x0 = sum(A)'/m; [f0, g1] = fgf2(x0);
r0 = f0; maxitn = 150000; intp=150001;
x_0 = x0', r0, maxitn, intp,
printf("\n Розв'язуємо задачу (20),(21) для 15 різних значень eps2\n");
eps2 = 1.e-2; ntest = 15;
for (in = 1:ntest)
  tstart = time();
  [xr2,fr2,itn2,ist2] = emshor(@fgf2,x0,r0,eps2,maxitn,intp);
  time2 = time()-tstart;
  printf("eps2  itn2  time2  f2  dx2  %7.1e  %6d  %5.1f  %20.17f  %9.6e\n",
         eps2,itn2,time2,fr2,norm(xr2-xstar));
  eps2 = eps2/100.0;
endfor
fr2,
printf("\n Друкуємо розв'язок (центр кулі мінімального радіусу)\n");
x_center = xr2',
```

Результати розрахунку за вищеприведеним кодом для 15 різних значень ε_2 наведені в табл. 2, а результати розрахунку за кодом, де оператор «`r = 0.5*ones(m,1)`» замінено на оператор «`r = 0.0*ones(m,1)`» наведені в табл. 3. Тут k^* – кількість ітерацій, затрачена на пошук точки x_{k^*}, t_2 – затрати часу в секундах на розв'язання задачі (20), (21), $f_2(x_{k^*})$ – значення функції $f_2(x)$ на ітерації k^*. Обчислювальний експеримент проводився на персональному комп'ютері із процесором AMD Ryzen 5 4500U 2.38 GHz, 16 GB у системі Windows 10 за допомогою GNU Octave версії 6.2.0.

ТАБЛИЦЯ 2. Результати роботи алгоритму **sylvester2** для тесту (25) при $n=30$ та $\varepsilon_2 \in \left[10^{-30} \div 10^{-2}\right]$

ε_2	k^*	t_2	$f_2(x_{k^*})$	$\left\|x_{k^*} - x_U^*\right\|$
1.0e-02	8776	0.7	1.48330237976035395	1.624189e-03
1.0e-04	16928	1.2	1.48319480119659541	2.089760e-04
1.0e-06	25053	1.7	1.48319208088122512	6.389005e-06
1.0e-08	33237	2.3	1.48319208054077234	2.500294e-06
1.0e-10	41375	2.8	1.48319208025169536	1.868036e-07
1.0e-12	49492	3.4	1.48319208025018057	4.624606e-09
1.0e-14	57642	3.9	1.48319208025017524	8.692244e-10
1.0e-16	65405	4.5	1.48319208025017546	1.016419e-10
1.0e-18	70597	4.8	1.48319208025017524	2.207081e-12
1.0e-20	73451	5.1	1.48319208025017524	7.020718e-13
1.0e-22	82433	5.7	1.48319208025017524	1.734658e-13
1.0e-24	92656	6.3	1.48319208025017524	1.643407e-13
1.0e-26	100652	6.9	1.48319208025017524	1.510768e-13
1.0e-28	109019	7.5	1.48319208025017524	1.497466e-13
1.0e-30	113118	7.7	1.48319208025017524	1.497466e-13

ТАБЛИЦЯ 3. Результати роботи алгоритму **sylvester2** для тесту (25) при $n=30$ та $r \equiv 0$

ε_2	k^*	t_2	$f_2(x_{k^*})$	$\left\|x_{k^*} - x_S^*\right\|$
1.0e-02	8051	0.6	0.98342957656730667	1.957242e-03
1.0e-04	16177	1.1	0.98319247196514914	1.808707e-04
1.0e-06	24323	1.7	0.98319210631926124	8.950859e-06
1.0e-08	32498	2.2	0.98319208049198403	1.215883e-06
1.0e-10	40628	2.8	0.98319208025261295	1.645955e-07
1.0e-12	48783	3.4	0.98319208025019111	5.535608e-09
1.0e-14	56918	4.0	0.98319208025017546	2.436637e-09
1.0e-16	64570	4.5	0.98319208025017513	1.271709e-10
1.0e-18	68146	4.7	0.98319208025017513	1.574935e-12
1.0e-20	71686	5.0	0.98319208025017502	9.318159e-14
1.0e-22	80167	5.5	0.98319208025017502	8.957242e-13
1.0e-24	85833	5.9	0.98319208025017502	8.684131e-13
1.0e-26	93262	6.4	0.98319208025017513	8.610223e-13
1.0e-28	100986	7.0	0.98319208025017513	8.588204e-13
1.0e-30	103076	7.1	0.98319208025017524	8.608145e-13

З табл. 2 та 3 видно, що алгоритм **sylvester2** не менш швидко, ніж алгоритм **sylvester1**, знаходить досить точні наближення до x_U^* – єдиної точки мінімуму функції $f_2(x)$. Так, згідно табл. 2, для того, щоб знайти точку x_{k^*}, для якої $\left\|x_{k^*} - x_U^*\right\| \le 10^{-9}$, досить зробити 49492

ітерацій, для чого потрібно всього 3.4 секунди, а для того, щоб знайти точку x_{k^*}, для якої $\left\|x_{k^*} - x_U^*\right\| \le 10^{-13}$, досить зробити 82433 ітерацій, затративши всього 5.7 секунди.

Вищеописане має місце і для табл. 3, в якій наведені результати розв'язання задачі (20), (21) при $r \equiv 0$. При цьому оптимальне значення цільової функції в задачі (20), (21) є рівним $f_2^* = R_S^* = \sqrt{1 - \frac{1}{n}}$ та співпадає з мінімальним радіусом гіперкулі, яка охоплює всі точки множини A_m, для задачі Сильвестра. Так, наприклад, для того, щоб знайти точку x_{k^*}, для якої $\left\|x_{k^*} - x_S^*\right\| \le 10^{-9}$, досить зробити 48783 ітерацій, для чого потрібно всього 3,4 секунди, а для того, щоб знайти точку x_{k^*}, для якої $\left\|x_{k^*} - x_S^*\right\| \le 10^{-13}$ досить зробити 80167 ітерацій, затративши всього 5.5 секунд.

Висновки. У роботі досліджено застосування алгоритму **emshor** – алгоритму методу еліпсоїдів для розв'язання задачі Сильвестра про найменшу обмежувальну гіперсферу та її узагальнення на випадок скінченного набору *n*-вимірних куль, заданих їх центрами та радіусами.

На основі методу **emshor** побудовано алгоритми для знаходження розв'язків обох задач Сильвестра. Алгоритм **sylvester1** призначений для розв'язання задачі мінімізації опуклої кусково-квадратичної функції, що є еквівалентною задачі знаходження кулі мінімального радіуса для скінченного набору точок. Алгоритм **sylvester2** призначений для розв'язання задачі мінімізації опуклої функції, що є еквівалентною узагальненій задачі знаходження кулі мінімального радіуса для скінченного набору куль з заданими їх центрами та радіусами. Результати тестування алгоритмів **sylvester1** та **sylvester2** демонструють високу швидкість роботи для сучасних комп'ютерів та високу точність за оптимальним значенням цільової функції при розв'язанні задач в *n*-вимірних просторах для невеликих значень $n = 2 \div 30$.

Фінансування. Робота підтримана грантом Volkswagen Foundation (грант № 97775) та грантом ДО "ВЦП КНУ імені Тараса Шевченка при НАН України" №2М-2024.

Авторські внески. Стецюк П.І. – концептуалізація, методологія, керівництво; Хом'як О.М. – дослідження, узагальнення, формальний аналіз, програмне забезпечення, написання – оригінальна чернетка; Давидов О.О. – програмне забезпечення, візуалізація, написання – редагування.

Список літератури

1. Sylvester J.J. A question in the geometry of situation. *Quarterly Journal of Mathematics*. 1857. T. 1. P. 79.
2. Юдин Д.Б., Немировский А.С. Информационная сложность и эффективные методы решения выпуклых экстремальных задач. *Экономика и математические методы.* 1976. Вып. 2. С. 357–369.
3. Шор Н.З. Метод отсечения с растяжением пространства для решения задач выпуклого программирования. *Кибернетика*. 1977. Т. 13. № 1. С. 94–95.
4. Шор Н.З., Билецкий В.И. Метод растяжения пространства для ускорения сходимости в задачах овражного типа. *Теория оптимальных решений*. 1969. № 2. С. 3–18.
5. Шор Н.З. Использование операции растяжения пространства в задачах минимизации выпуклых функций. *Кибернетика*. 1970. Т. 6. № 1. С. 6–12.
6. Шор Н.З. Методы минимизации недифференцируемых функций и их приложения. Киев: Наукова думка, 1979. 200 с.
7. Octave. http://www.octave.org (звернення: 12.03.2024)
8. Fischer A., Khomyak O., Stetsyuk P. The ellipsoid method and computational aspects. *Commun. Optim. Theory.* 2023. No. 21. P. 1–14.
9. Стецюк П.І., Фішер А., Хом'як О.М. Уніфіковане представлення класичного методу еліпсоїдів. *Кібернетика та системний аналіз*. 2023. Т. 59. № 5. С. 113–123.

10. Задача про найменше коло. https://en.wikipedia.org/wiki/Smallest-circle_problem. (звернення: 12.03.2024)
11. Хом'як О.М., Давидов О.О. Метод еліпсоїдів для гіперкулі мінімального радіуса. Матеріали VII-ї Міжнародної наукової конференції "М*атематичне моделювання, оптимізація та інформаційні технології* (*ММОТІ-2021*)", 15–19 листопада 2021 р. С. 340–342.

Одержано 12.03.2024

Стецюк Петро Іванович,
доктор фізико-математичних наук, завідувач відділу методів негладкої оптимізації
Інституту кібернетики імені В.М. Глушкова НАН України, Київ,
https://orcid.org/0000-0003-4036-2543
stetsyukp@gmail.com

Хом'як Ольга Миколаївна,
кандидат фізико-математичних наук, старший науковий співробітник відділу математичних методів теорії надійності складних систем Інституту кібернетики імені В.М. Глушкова НАН України, Київ,
https://orcid.org/0000-0002-5384-9070
khomiak.olha@gmail.com

Давидов Олександр Олександрович,
магістр другого курсу,
Київський національний університет імені Тараса Шевченка, Київ.
https://orcid.org/0000-0001-7232-4926
davydov0508@gmail.com

УДК 519.85

П.І. Стецюк [1] [*], О.М. Хом'як [1], О.О. Давидов [2]

Використання методу еліпсоїдів для задачі Сильвестра та її узагальнення

[1] Інститут кібернетики імені В.М. Глушкова НАН України, Київ
[2] Київський національний університет імені Тараса Шевченка, Україна
[*] Листування: stetsyukp@gmail.com

Задача Сильвестра або задача про найменше обмежувальне коло – це задача про побудову круга найменшого радіуса, який містить скінченний набір точок на площині. В n-вимірному просторі їй відповідає задача про найменшу обмежувальну гіперсферу, яка можна сформулювати як задачу мінімізації опуклої кусочно-квадратичної функції.

Стаття присвячена дослідженню застосування методу еліпсоїдів для розв'язання цієї задачі та мінімаксної задачі опуклого програмування, що є еквівалентною узагальненій задачі про найменшу обмежувальну гіперсферу. Узагальнена задача полягає у знаходженні центру кулі в n-вимірному просторі, яка має мінімальний радіус та містить скінченний набір n-вимірних куль, заданих їх центрами та радіусами.

Матеріал статті викладений у 3 розділах. У розділі 1 описано алгоритм **emshor** – алгоритм методу еліпсоїдів для задачі мінімізації довільної опуклої функції, доведено теореми про його збіжність, наведено геометричну інтерпретацію алгоритму, яка базується на використанні еліпсоїда мінімального об'єму. Тут представлено octave реалізацію алгоритму **emshor**, яку можна успішно застосовувати для мінімізації негладких опуклих функцій, якщо кількість змінних $n = 2 \div 30$.

У розділі 2 побудовано алгоритм **sylvester1**, який є застосуванням алгоритму **emshor** для розв'язання задачі мінімізації опуклої кусково-квадратичної функції, що є еквівалентною задачі знаходження кулі мінімального радіуса для скінченного набору точок. У розділі 3 побудовано алгоритм **sylvester2**, який є застосуванням алгоритму **emshor** для розв'язання задачі мінімізації опуклої функції, що є еквівалентною узагальненій задачі знаходження кулі мінімального радіуса для скінченного набору куль з заданими їх центрами та радіусами. Результати тестування алгоритмів **sylvester1** та **sylvester2** демонструють високу швидкість роботи для сучасних комп'ютерів та високу точність за оптимальним значенням цільової функції при розв'язанні задач в n-вимірних просторах для невеликих значень $n = 2 \div 30$.

Ключові слова: метод еліпсоїдів, опукла функція, задача Сильвестра, кусково-гладка функція, мінімаксна задача.

UDC 519.85

Petro Stetsyuk [1 *], Olha Khomiak [1], Oleksander Davydov [2]

Using the Ellipsoid Method for Sylvester's Problem and its Generalization

[1] *V.M. Glushkov Institute of Cybernetics of the NAS of Ukraine, Kyiv*
[2] *Taras Shevchenko National University of Kyiv, Ukraine*
[*] *Correspondence: stetsyukp@gmail.com*

Sylvester's problem or the problem of the smallest bounding circle is the problem of constructing a circle of the smallest radius that contains a finite set of points on the plane. In *n*-dimensional space, it corresponds to the problem of the smallest bounding hypersphere, which can be formulated as the problem of minimizing a convex piecewise quadratic function.

The article is dedicated to study of the ellipsoid method application for solving this problem and the minimax convex programming problem, which is equivalent to the generalized problem of the smallest bounding hypersphere. The generalized problem consists in finding the center of a sphere in an *n*-dimensional space that has a minimal radius and contains a finite set of *n*-dimensional spheres given by their centers and radii.

The article consists of 3 sections. Section 1 describes the **emshor** algorithm – the algorithm of the ellipsoid method for problem of minimization of an arbitrary convex function, proves its convergence theorems, gives a geometric interpretation of the algorithm, which is based on the use of a minimum volume ellipsoid. Octave implementation of the **emshor** algorithm is presented here, which can be successfully applied for non-smooth convex functions minimization if the number of variables is $n = 2 \div 30$.

In Section 2, the **sylvester1** algorithm is built, which is an application of the **emshor** algorithm for solving the problem of minimizing a convex piecewise quadratic function, which is equivalent to the problem of finding a sphere of minimum radius for a finite set of points. In Section 3, the **sylvester2** algorithm is built, which is an application of the **emshor** algorithm for solving the problem of minimization of a convex function, which is equivalent to the generalized problem of finding a sphere of minimum radius for a finite set of spheres with given centers and radii. The results of testing the **sylvester1** and **sylvester2** algorithms demonstrate high working speed for modern computers and high accuracy in terms of optimal value of the objective function when solving problems in n-dimensional spaces for small values $n = 2 \div 30$.

Keywords: ellipsoid method, convex function, Sylvester's problem, piecewise smooth function, minimax problem.