Інститут кібернетики імені В.М. Глушкова НАН України
Відділ методів негладкої оптимізації

Робочі матеріали
Випуск 19-01

П.І. Стецюк, Т.В. Бєлих, О.І. Криворучко

ТЕОРІЯ ТА ПРОГРАМНІ РЕАЛІЗАЦІЇ *r*-АЛГОРИТМІВ ШОРА

Київ – 2019

УДК 519.8

Автори
П.І. Стецюк, Т.В. Бєлих, О.І. Криворучко

Теорія та програмні реалізації *r*-алгоритмів Шора[1]. – Київ: 2019. – 28 с. Робочі матеріали. Випуск 19-01 / Інститут кібернетики імені В.М. Глушкова НАН України, відділ методів негладкої оптимізації.

Розглядаються три обчислювальні форми r-алгоритмів, які відрізняються обсягом обчислень на одній ітерації. Наведено результати про збіжність граничного варіанта r-алгоритма для опуклих гладких функцій і $r_\mu(\alpha)$-алгоритма для опуклих кусочно-гладких функцій. Обговорюються практичні аспекти $r(\alpha)$-алгоритмів з постійним α, коефіцієнтом розтягу простору, та адаптивним способом регулювання кроку у напрямку нормованого антисубградієнта в перетвореному просторі змінних. Описана дослідницька програма **ralgb5a**, яка призначена для мінімізації гладких та негладких опуклих функцій.

Іл. 1. Табл. 1. Бібліогр.: с. 26–27 (22 найменування).

UDC 519.8

Authors
P.I. Stetsyuk, T.V. Belykh, O.I. Kryvoruchko

Theory and program realizations of Shor's r-algorithms. – Kyiv: 2019. – 28 p. Working papers. Issue 19-01 / V.M. Glushkov Institute of Cybernetics of NAS of Ukraine, Department of non-smooth optimization methods.

Three computational forms of r-algorithms that differ in the number of computations at iteration are considered. The results of convergence of the limit version of r-algorithm for convex smooth functions and $r_\mu(\alpha)$-algorithm for convex piecewise-smooth functions are presented. The practical aspects of $r(\alpha)$-algorithms with constant coefficient of space dilation α, and adaptive adjustment of step coefficient in the direction of the normalized antisubgradient in the transformed space of variables are discussed. The research program **ralgb5a** to minimize smooth and non-smooth convex functions is described.

Fig. 1. Tab. 1. Ref.: p. 26–27 (22 references).

[1] Робота виконана за фінансової підтримки НАН України (проект № 0117U000327 – перший і другий автори, проект № 0118U005227 – другий і третій автори) та Volkswagen Foundation (грант No 90 306 – перший автор).

ВСТУП

Субградієнтні методи з розтягом простору в напрямку різниці двох послідовних субградієнтів отримали назву r-алгоритмів [1, 2, 3]. Програмні реалізації r-алгоритмів виявилися достатньо конкурентоспроможними як за часом обчислень, так і за точністю результатів з найбільш ефективними методами розв'язання гладких погано обумовлених задач. Прискорену збіжність r-алгоритмів при мінімізації негладких опуклих функцій забезпечує тісний взаємозв'язок двох принципів у чисельних методах оптимізації.

Перший принцип полягає у використанні процедури найшвидшого спуску в напрямку антисубградієнта опуклої функції в перетвореному просторі змінних. Якщо пошук мінімуму функції здійснюється точно, то перший принцип гарантує монотонність значень опуклої функції для точок мінімізуючої послідовності, яка будується за допомогою r-алгоритмів. Якщо пошук мінімуму функції здійснюється наближено, то «монотонність» по мінімізуючій функції замінюється на «майже монотонність». Однак найшвидший спуск для негладких функцій може зациклитися, тому для запобігання цьому використовується другий метод.

Другий принцип спрямований на зменшення ступеня витягнутості поверхонь рівня яружних функцій у перетвореному просторі змінних. Він полягає у використанні операції розтягу простору в напрямку різниці двох послідовних субградієнтів, де другий субградієнт обчислений у точці мінімуму функції в напрямку першого антисубградієнта. В результаті цього розтягу зменшуються поперечні складові субградієнтів уздовж напрямку точки мінімуму, що забезпечує достатньо швидку збіжність субградієнтного процесу з розтягом простору.

Комбінації цих принципів при певному регулюванні кроку найшвидшого спуску (точного чи наближеного) і відповідному виборі коефіцієнта розтягу простору забезпечують прискорену збіжність конкретних варіантів r-алгоритмів і гарантують їхню монотонність (або майже монотонність) за значеннями мінімізуючої функції. Це підтверджується результатами численних застосувань r-алгоритмів у задачах лінійного і нелінійного програмування, блочних задачах з різними схемами декомпозиції, при вирішенні мінімаксних і матричних задач оптимізації, для обчислення двоїстих лагранжевих оцінок у багатоекстремальних і комбінаторних задачах оптимізації [4, 5, 6, 7].

1. ТРИ ОБЧИСЛЮВАЛЬНІ ФОРМИ *r*-АЛГОРИТМІВ

Розглядається задача мінімізації опуклої функції $f(x)$, де $x \in E^n$ – вектор із n змінних. Мінімальне значення функції будемо позначати $f^* = f(x^*)$, $x^* \in X^*$. Також припустимо, що $f(x)$ має обмежену множину мінімумів X^*, тобто виконується умова $\lim_{\|x\|\to\infty} f(x) = +\infty$. Ця умова забезпечує коректність регулювання кроку в r-алгоритмах. Нехай $\{\sim_k\}_{k=0}^{\infty}$ – набір коефіцієнтів розтягу простору, таких що $\sim_k > 1$.

r-Алгоритмом для мінімізації $f(x)$ називається ітеративна процедура знаходження послідовності n-вимірних векторів $\{x_k\}_{k=0}^{\infty}$ і послідовності $n \times n$-матриць $\{B_k\}_{k=0}^{\infty}$ за наступним правилом:

$$x_{k+1} = x_k - h_k B_k \P_k, \quad B_{k+1} = B_k R_{\ k}(¦_k), \quad k = 0,1,2,\dots, \tag{1}$$

де

$$\P_k = \frac{B_k^T g_f(x_k)}{\left\| B_k^T g_f(x_k) \right\|}, \quad h_k \ge h_k^* = \arg\min_{h \ge 0} f(x_k - hB_k \P_k), \tag{2}$$

$$_k = \frac{1}{\sim_k} < 1, \quad ¦_k = \frac{B_k^T r_k}{\| B_k^T r_k \|}, \quad \text{де} \quad r_k = g_f(x_{k+1}) - g_f(x_k), \tag{3}$$

де x_0 – початкова точка; $B_0 = I_n$ – одинична $n \times n$-матриця; h_k^* – величина кроку за умови мінімуму функції $f(x)$ у напрямку нормованого антисубградієнта у перетвореному просторі змінних; $R\ (¦) = I_n + (\ \ -1)¦\,¦^T$ – оператор стиснення простору субградієнтів у нормованому напрямку $¦$ з коефіцієнтом $\ = \frac{1}{\sim} < 1$; $g_f(x_k)$ і $g_f(x_{k+1})$ – субградієнти функції $f(x)$ у точках x_k і x_{k+1}.

Якщо на ітерації k для процесу (1)–(3) виконуються певні критерії (умови) зупинки, то покладемо $k^* = k$, $x_k^* = x_k$ і закінчимо роботу алгоритму.

На кожній ітерації r-алгоритмів реалізується субградієнтний спуск для опуклої функції $\ddot{}\,(y) = f(B_k y)$ у перетвореному просторі змінних $y = A_k x$, де $A_k = B_k^{-1}$.

Якщо ж обидві частини формули $x_{k+1} = x_k - h_k B_k \P_k$ помножити зліва на матрицю A_k, то отримаємо

$$y_{k+1} = A_k x_{k+1} = A_k x_k - h_k \P_k = y_k - h_k \frac{B_k^T g_f(x_k)}{\|B_k^T g_f(x_k)\|} = y_k - h_k \frac{g_{¨}(y_k)}{\|g_{¨}(y_k)\|}, \qquad (4)$$

де вектор $g_{¨}(y_k) = B_k^T g_f(x_k)$ є субградієнтом функції $¨(y) = f(B_k y)$ у точці $y_k = A_k x_k$ простору змінних $y = A_k x$. Це випливає з того, що субградієнт функції $f(x)$ у точці x_k задовольняє нерівність

$$f(x) \geq f(x_k) + (g_f(x_k))^T (x - x_k) \quad \forall x \in E^n,$$

звідки, виконавши заміну змінних $x = B_k y$, одержимо

$$¨(y) \geq ¨(y_k) + (B_k^T g_f(x_k))^T (y - y_k) = ¨(y_k) + (g_{¨}(y_k))^T (y - y_k) \quad \forall y \in E^n.$$

Якщо $h_k = h_k^*$, то формула (4) означає точний пошук мінімуму функції $¨(y) = f(B_k y)$ у напрямку нормованого антисубградієнта в перетвореному просторі змінних $y = A_k x$; якщо ж $h_k \approx h_k^*$, то (4) означає наближений пошук.

Якщо функція $f(x)$ є недиференційовною в точці x_k, то є можливим випадок $h_k = h_k^* = 0$, з яким пов'язані основні проблеми з умовою зупинки r-алгоритмів для негладких функцій. Якщо $h_k = 0$, то це не означає, що в точці x_k процес спуску потрібно зупинити.

У процесі виконання серії ітерацій з нульовим кроком наступна точка $x_{k+1} = x_k$ не змінюється, але змінюється матриця B_{k+1} і наступний субградієнт $g_f(x_{k+1})$. Це означає, що за рахунок послідовних розтягувань простору в напрямку різниці двох субградієнтів (обидва субградієнти отримані в точці x_k) здійснюється пошук потрібного напрямку спадання функції з точки x_k.

Монотонність по мінімізуючій функції для r-алгоритмів забезпечує саме розтяг простору в напрямку різниці двох послідовних субградієнтів. Саме цим r-алгоритми принципово відрізняються від субградієнтних методів з розтягом простору в напрямку субградієнта, які для негладких опуклих функцій у принципі не можуть бути монотонними по мінімізуючій функції. Покажемо це на прикладі кусково-гладкої функції (див. рисунок).

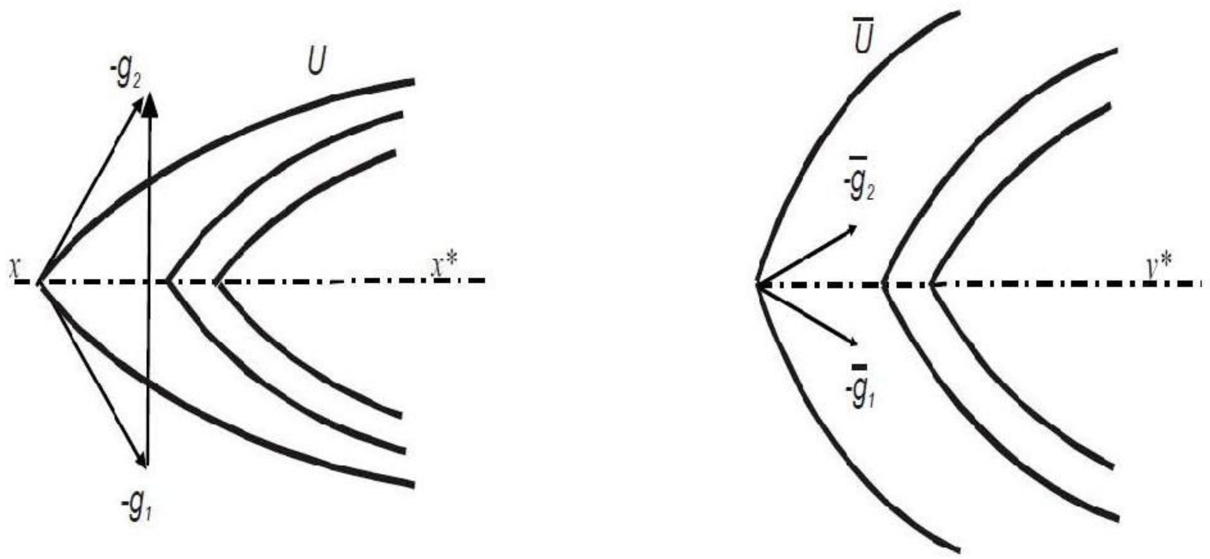

РИСУНОК. Після розтягу простору за різницею двох субградієнтів (ліворуч) антисубградієнти стають напрямками спадання функції (праворуч)

Нехай ми знаходимося на межі двох «кусків» кусково-гладкої поверхні рівня, а градієнти до цих гладких «кусків», обчислені в даній точці, утворюють тупий кут (рисунок ліворуч). Жодний розтяг простору в напрямку градієнтів не може перетворити цей кут на гострий, а може лише наблизити його до $®/2$, залишаючи тупим. Тому, застосовуючи розтяг простору в напрямку субградієнта, неможливо отримати напрямок спадання функції у вигляді антиградієнта до одного з «кусків» у розтягнутому просторі.

Слід зауважити, що розтяг простору в напрямку різниці двох наведених градієнтів з достатнім коефіцієнтом розтягу перетворює тупий кут між градієнтами на гострий, тобто відповідні образи цих антиградієнтів у розтягнутому просторі стають напрямками спадання функції (рисунок праворуч).

Отже, сімейство r-алгоритмів визначається послідовністю коефіцієнтів розтягу простору $\{\sim_k\}_{k=0}^{\infty}$, послідовністю величин кроків $\{h_k\}_{k=0}^{\infty}$ та критеріями зупинки. При цьому величина кроку обирається з умови точного (наближеного) мінімуму функції у напрямку антисубградієнта в перетвореному просторі змінних, завдяки чому визначаються два послідовних субградієнти, розтяг по різниці яких покращує властивості яружної функції у перетвореному просторі змінних. При певному регулюванні кроку та коефіцієнтів розтягу простору r-алгоритми є монотонними (чи майже монотонними) по мінімізуючій функції.

Метод (1)–(3) називають B-формою r-алгоритмів, на кожній його ітерації коригується матриця, пов’язана з заміною змінних $x = By$. Ітерація методу потребує порядку $5n^2$ арифметичних операцій множення, які визначають обчислювальну трудоємність ітерації (операціями додавання знехтуємо у зв’язку з їхнім малим внеском у трудоємність ітерації). З них $3n^2$ множень необхідні для обчислення векторів $B_k \xi_k$, $B_k^T g_f(x_k)$ та $B_k^T r_k$ (множення матриці на вектор), а $2n^2$ множень необхідні для однорангової корекції матриці $B_{k+1} = B_k R_{\beta_k}(\eta_k)$. Справді,

$$B_{k+1} = B_k R_{\beta_k}(\eta_k) = B_k(I_n + (\beta_k - 1)\eta_k \eta_k^T) = B_k + (\beta_k - 1)(B_k \eta_k)\eta_k^T,$$

звідки легко побачити, що обчислення вектора $\eta = B_k \eta_k$ потребує стільки ж n^2 множень, скільки і побудова однорангової матриці $\eta \eta_k^T$.

У r-алгоритмів є ще одна B-форма, яка порівняно з методом (1)–(3), дозволяє заощадити n^2 операцій множення на кожній ітерації. Економні r-алгоритми описуються ітеративною процедурою знаходження послідовності векторів $\{x_k\}_{k=0}^{\infty}$ і матриць $\{B_k\}_{k=0}^{\infty}$ за таким правилом:

$$x_{k+1} = x_k - h_k B_k \frac{\tilde{g}_k}{\|\tilde{g}_k\|}, \quad B_{k+1} = B_k R_{\beta_k}(\eta_k), \quad k = 0, 1, 2, \ldots, \tag{5}$$

де

$$h_k \geq h_k^* = \arg\min_{h \geq 0} f(x_k - hB_k \frac{\tilde{g}_k}{\|\tilde{g}_k\|}), \quad g_{k+1}^* = B_k^T g_f(x_{k+1}), \tag{6}$$

$$\beta_k = \frac{1}{\alpha_k}, \quad \eta_k = \frac{g_{k+1}^* - \tilde{g}_k}{\| g_{k+1}^* - \tilde{g}_k \|}, \quad \tilde{g}_{k+1} = R_{\beta_k}(\eta_k) g_{k+1}^*, \tag{7}$$

де x_0 – початкова точка, така, що $x_0 \neq x^*$; $\tilde{g}_0 = B_0^T g_f(x_0)$, B_0 – невироджена $n \times n$-матриця; $g_f(x_k)$ і $g_f(x_{k+1})$ – субградієнти функції $f(x)$ в точках x_k і x_{k+1}.

Якщо на ітерації k виконуються деякі критерії зупинки, то покладемо $k^* = k$, $x_k^* = x_k$ і закінчимо роботу алгоритму.

Ітерація методу (5)–(7) потребує $4n^2$ операцій множення. З них $2n^2$ множень потрібні для обчислення векторів $B_k \tilde{g}_k$ і $B_k^T g_f(x_{k+1})$, а $2n^2$ множень – для однорангової корекції матриці $B_{k+1} = B_k R_{\ k}(¦_k)$. Економія n^2 множень пов’язана з тим, що $\tilde{g}_{k+1} = B_{k+1}^T g_f(x_{k+1})$, субградієнт у просторі змінних $y = A_{k+1} x$ перераховується з урахуванням уже обчисленого $g_{k+1}^* = B_k^T g_f(x_{k+1})$ – субградієнта в просторі змінних $y = A_k x$. Перерахунок субградієнта $\tilde{g}_{k+1}$ обчислюється за формулою

$$\tilde{g}_{k+1} = B_{k+1}^T g_f(x_{k+1}) = R_{\ k}(¦_k) B_k^T g_f(x_{k+1}) = R_{\ k}(¦_k) g_{k+1}^* =$$

$$= (I_n + (\ _k - 1) ¦_k ¦_k^T) g_{k+1}^* = g_{k+1}^* + (\ _k - 1)(¦_k^T g_{k+1}^*) ¦_k,$$

яка не потребує операції множення матриці на вектор. При перерахунку виникає більше помилок при обчисленні нормованого субградієнта в перетвореному просторі за формулою $¶_k = \frac{\tilde{g}_k}{\|\tilde{g}_k\|}$, ніж при обчисленні нормованого субградієнта $¶_k = \frac{B_k^T g_f(x_k)}{\|B_k^T g_f(x_k)\|}$, як у методі (1)–(3).

r-Алгоритми можна записати в H-формі за допомогою симетричної матриці $H_k = B_k B_k^T$ (за аналогією з методами змінної метрики). Їм відповідає ітеративна процедура знаходження послідовності векторів $\{x_k\}_{k=0}^{\infty}$ і симетричних матриць $\{H_k\}_{k=0}^{\infty}$ за таким правилом:

$$x_{k+1} = x_k - h_k \frac{H_k g_f(x_k)}{\sqrt{\left(g_f(x_k)\right)^T H_k g_f(x_k)}},$$
$$H_{k+1} = H_k + ({}_k^2 - 1) \frac{H_k r_k r_k^T H_k}{r_k^T H_k r_k}, \; k = 0,1,\ldots, \tag{8}$$

де

$$h_k \geq h_k^* = \arg\min_{h \geq 0} f\left(x_k - h \frac{H_k g_f(x_k)}{\sqrt{\left(g_f(x_k)\right)^T H_k g_f(x_k)}} \right),$$
$${}_k = \frac{1}{\sim_k} < 1, \quad r_k = g_f(x_{k+1}) - g_f(x_k), \tag{9}$$

де x_0 – початкова точка; $H_0 = I_n$ – одинична $n \times n$-матриця; h_k – величина кроку, яка не менша, ніж h_k^*; $g_f(x_k)$ і $g_f(x_{k+1})$ – субградієнти функції $f(x)$ в точках x_k і x_{k+1}. Якщо на ітерації k виконуються умови зупинки, то покладемо $k^* = k$, $x_k^* = x_k$ і завершуємо роботу алгоритму.

В H-формі r-алгоритмів формула для перерахунку наступного наближення x_{k+1} випливає із справедливості такого ланцюжка співвідношень

$$B_k \frac{B_k^T g_f(x_k)}{\left\| B_k^T g_f(x_k) \right\|} = \frac{B_k B_k^T g_f(x_k)}{\sqrt{\left(B_k^T g_f(x_k)\right)^T B_k g_f(x_k)}} =$$
$$= \frac{H_k g_f(x_k)}{\sqrt{\left(g_f(x_k)\right)^T B_k B_k^T g_f(x_k)}} = \frac{H_k g_f(x_k)}{\sqrt{\left(g_f(x_k)\right)^T H_k g_f(x_k)}}.$$

Формула для перерахунку симетричної матриці H_{k+1} випливає з такого ланцюжка перетворень

$$H_{k+1} = B_{k+1} B_{k+1}^T = B_k R_{\ k}(¦_k) \left(B_k R_{\ k}(¦_k) \right)^T = B_k R_{\ k}(¦_k) R_{\ k}^T(¦_k) B_k^T =$$
$$= B_k R_{\ k}(¦_k) R_{\ k}(¦_k) B_k^T = B_k R_{\ k^2}(¦_k) B_k^T = B_k (I_n + ({}_k^2 - 1) ¦_k ¦_k^T) B_k^T =$$
$$= B_k B_k^T + ({}_k^2 - 1) B_k ¦_k ¦_k^T B_k^T = H_k + ({}_k^2 - 1) \frac{B_k B_k^T r_k r_k^T B_k B_k^T}{\left\| B_k^T r_k \right\|^2} =$$
$$= H_k + ({}_k^2 - 1) \frac{H_k r_k r_k^T H_k}{\left(B_k^T r_k\right)^T B_k^T r_k} = H_k + ({}_k^2 - 1) \frac{H_k r_k r_k^T H_k}{r_k^T B_k B_k^T r_k} = H_k + ({}_k^2 - 1) \frac{H_k r_k r_k^T H_k}{r_k^T H_k r_k}.$$

H-форма r-алгоритмів економніша, ніж B-форма. За оперативною пам’яттю – майже у два рази, оскільки потрібно зберігати симетричну матрицю, а за трудоємністю – мінімум у 1,66 разів. Навіть якщо симетричну матрицю H_k зберігати як повну матрицю розмірністю $n \times n$, то ітерація методу (8)–(9) потребує $3n^2$ операцій множення. $2n^2$ множень необхідно для обчислення векторів $H_k g_f(x_k)$ і ¦ $= H_k r_k$, а n^2 множень – для обчислення однорангової матриці ¦ ¦$^T = H_k r_k r_k^T H_k$, яка використовується при перерахунку матриці H_{k+1}. Проте, H-форма r-алгоритмів обчислювально менш стійка, ніж B-форма.

Так, при реалізації методу (8)–(9) необхідно контролювати, щоб матриця H_k була додатньо визначеною. У методах (1)–(3) і (5)–(7) такий контроль не потрібен, оскільки обчислення пов’язані з додатньо визначеною матрицею $H_k = B_k B_k^T$.

Обчислювальні характеристики описаних форм r-алгоритмів за пам’яттю і трудоємністю наведені в таблиці. Хоча теоретично всі три форми r-алгоритмів однакові, обчислювальна стійкість їхніх комп’ютерних реалізацій, що подана в останньому стовпчику таблиці, відрізняється.

Для обох B-форм вона позначена як «хороша», але перевагу віддано методу (1)–(3) і відзначено знаком «+». Це обумовлено тим, що перерахунок субградієнта при переході в наступний простір змінних, що використовується в економній B-формі, сприяє збільшенню помилок щодо обчислення цього ж субградієнта в методі (1)–(3).

Обчислювальна стійкість r-алгоритмів в H-формі відзначена як «середня». Це означає, що за допомогою них взагалі неможливо знайти такі наближення до точки мінімуму, які можна знайти з r-алгоритмів у B-формі. r-Алгоритми в H-формі можна використовувати, якщо не вимагається високої точності для знаходження мінімуму функції $f(x)$.

ТАБЛИЦЯ. Обчислювальні характеристики трьох форм r -алгоритмів

Форма r -алгоритмів	Вид методу	Оперативна пам’ять	Трудомісткість ітерації	Стійкість методу
B -форма	метод (1)–(3)	$\sim n^2$	$\sim 5n^2$	хороша (+)
економна B -форма	метод (5)–(7)	$\sim n^2$	$\sim 4n^2$	хороша
H -форма	метод (8)–(9)	$\sim n^2 / 2$	$\sim 3n^2$	середня

2. ТРИ ТЕОРЕМИ ПРО ЗБІЖНІСТЬ *r*-АЛГОРИТМІВ

Для опуклих функцій теоретичні результати про збіжність r -алгоритмів пов’язані з їхніми модифікаціями: з граничним варіантом r -алгоритмів для гладких функцій [1] та $r_«(\sim)$ -алгоритмом для негладких функцій [8].

Перший результат полягає в тому, що граничний варіант r -алгоритмів є проективним методом спряжених градієнтів. Другий результат означає, що для задачі мінімізації опуклої двічі неперервно-диференційовної функції $f(x)$ граничний варіант r -алгоритмів з відновленням володіє квадратичною швидкістю збіжності за деяких умов гладкості та регулярності $f(x)$. Третій результат полягає в тому, що за певних умов $r_«(\sim)$ -алгоритм збігається до точки мінімуму для кусково-гладких опуклих функцій. Що це за результати та якими умовами вони визначаються, буде показано далі.

У граничному варіанті r -алгоритмів припускається, що коефіцієнт розтягу простору нескінченний (при цьому $\;_k = 0$, $k = 0,1,\ldots$), а крок обирається з умов мінімуму функції $f(x)$ за напрямком антисубградієнта $(h_k = k_k^*, k = 0,1,\ldots)$. Якщо $\;_k = 0$, то оператор $R_{\;_k}(¦_k)$ визначається формулою

$$R_0(¦_k) = I_n - ¦_k ¦_k^T, \quad \text{де} \quad ¦_k = \frac{B_k r_k}{\|B_k r_k\|}, \quad k = 1, 2, \ldots, n. \tag{10}$$

Формула (10) означає, що оператор $R_0(\text{¦}_i)$ є оператором проектування на підпростір, що є ортогональним вектору ¦_i. Добуток операторів $\prod_{i=1}^{k} R_0(\text{¦}_i)$ не залежить від порядку співмножників, є самоспряженим оператором і виконує проекцію на підпростір, що є ортогональним доповненням до лінійної оболонки взаємно ортогональних векторів ¦_i, $i = 1, 2, \ldots, k$. Звідси випливає така теорема.

Теорема 1. Для граничного варіанта r-алгоритмів на деякій ітерації $k^* \le n$ обов'язково виконується умова $B_{k^*}^T g_f(x_{k^*}) = 0$.

Граничний варіант r-алгоритмів є проективним методом спряжених градієнтів. Для невід'ємно визначеної квадратичної функції він знаходить точку мінімуму x^* за кількістю ітерацій, що не перевищує n-розмірності вектора змінних. Із теореми 1 випливає, що після k^* ітерацій для граничного варіанта r-алгоритмів продовжувати обчислення неможливо, оскільки напрямок руху в перетвореному просторі перетворюється на нульовий.

Отже, для мінімізації гладких функцій $f(x)$ у граничному варіанті r-алгоритму після виконання умови $B_{k^*}^T g_f(x_{k^*}) = 0$ необхідно застосовувати «відновлення», тобто після деякої кількості ітерацій, що не перевищує n, потрібно «відновлювати» матрицю B_k, замінюючи її одиничною матрицею I_n. Побудований таким чином алгоритм називають граничним варіантом r-алгоритмів з відновленням. Справедлива наступна теорема.

Теорема 2. Нехай функція $f(x)$, визначена в E^n, двічі неперервно-диференційовна в деякому околі S точки мінімуму x^*, причому в цьому околі матриця інших похідних $H(x)$ задовольняє умові

$$\|H(x) - H(x')\| \le L\|x - x'\|, \quad x, x' \in S. \tag{11}$$

Крім того, $H(x^*)$ – додатньо визначена матриця. Тоді для точки x^* знайдеться такий окіл $S' \subseteq S$, що при $x_0 \in S'$ знайдеться таке число $c > 0$, що

$$\left\| x_n - x^* \right\| \le c \left\| x_n - x^* \right\|^2 ,$$

де x_n – точка, отримана після n кроків роботи вищенаведеного алгоритму (якщо для деякого $k^* < n$ отримали $B_{k^*}^T g_f(x_{k^*}) = 0$, то припускаємо $x_n = x_{k^*}$).

Отже, для задачі мінімізації опуклої двічі неперервно-диференційовної функції $f(x)$ граничний варіант r-алгоритмів з відновленням матриці B_k після кожних n ітерацій має квадратичну швидкість збіжності при умовах гладкості та регулярності $f(x)$, наведених у теоремі 2.

Для мінімізації гладких опуклих функцій r-алгоритми займають проміжне місце між методом найшвидшого спуску та алгоритмами квазіньютонівського типу зі змінною метрикою. Якщо $_k = 1$ і $h_k = k_k^*$, то розтяг простору не потрібний, і r-алгоритм переходить в метод найшвидшого спуску. Якщо $_k = 0$ і $h_k = k_k^*$, то отримуємо граничний варіант r-алгоритмів, що є проективним методом спряжених градієнтів і для квадратичної опуклої функції збігається не більше, ніж за n ітерацій. Якщо $_k = \quad < 1$ і $h_k = k_k^*$, то отримаємо варіант r-алгоритмів, де відновлення матриці B_k не потрібне, а алгоритм буде збігатися швидше, ніж метод найшвидшого спуску. Цим у значній мірі пояснюється чудова властивість r-алгоритмів, яка полягає в тому, що їх конкретні реалізації показують дуже хороші результати при мінімізації яружних опуклих функцій.

У роботах Н.З. Шора $r_{«}(\sim)$-алгоритм створювався для мінімізації майже-диференційовних функцій [9], клас яких є ширшим, ніж клас опуклих функцій. У зв'язку з цим Б. Мордухович, М. Солодов і М. Тодд у передмові до спецвипуску журналу «Optimization Methods and Software» [10], присвяченому Н.З. Шору, відмітили: *«У 1972 році Н.З. Шор ввів фундаментальне поняття узагальненого диференціала для локально ліпшицевих функцій, який він назвав «множина майже-градієнтів». Цей диференціал визначено як сукупність граничних точок звичайних градієнтів ліпшицевої неперервної функції, яка є майже всюди диференційовною за класичною теоремою Радемахера. Згодом ця гранична множина широко застосовувалася під назвою B-градієнта* *та B-якобіана вектор-функцій при розробці негладких версій ньютонівського методу. Варто відмітити, що в тій же статті 1972 року Шор також ввів та використав поняття опуклої оболонки множини майже-градієнтів, яку він назвав «множиною узагальнених майже-градієнтів». Ця множина була перевідкрита Ф. Кларком і стала широко відомою в недиференційовній оптимізації як узагальнений градієнт Кларка для ліпшицевих функцій».*

Результати про збіжність $r_\alpha(\beta)$-алгоритма для мінімізації майже-диференційовних функцій детально викладені в роботах [8] та [4, с. 102–113]. Тому далі опишемо тільки те, до чого призводять ці результати для задачі мінімізації кусково-гладких функцій вигляду

$$f(x) = \max_{1 \le i \le m} f_i(x), \quad \text{де} \quad f_i(x) \text{ – гладкі опуклі функції}, \quad i = 1, 2, \ldots, m. \qquad (12)$$

Функція (12) опукла. Скористаємося для її мінімізації $r_\alpha(\beta)$-алгоритмом. Позначимо $G_f(x) = \left\{ \bigcup_{i \in I(x)} g_{f_i}(x) \right\}$, де $I(x) = \left\{ i \,\middle|\, f_i(x) = f(x) \right\}$, $g_{f_i}(x)$ – градієнт функції $f_i(x)$ в точці x.

Нехай $\beta > 1$ – коефіцієнт розтягу простору; α – константа, така, що $0 \le \alpha < 1$; x_0 – початкова точка; $g_f(x_0)$ – субградієнт функції f в точці x_0 (не будь-який субградієнт, а обраний із множини $G_f(x_0)$, тобто $g_f(x_0) \in G_f(x_0)$); B_0 – невироджена $n \times n$-матриця. $r_\alpha(\beta)$-Алгоритмом є ітераційна процедура побудови послідовностей векторів $\{x_k\}_{k=0}^{\infty}$ та матриць $\{B_k\}_{k=0}^{\infty}$, де перехід від k-ої ітерації до $(k+1)$-ої ітерації відбувається за таким правилом:

1. Обчислимо наступну точку виду

$$x_{k+1} = x_k - h_k B_k g_{\varphi_k}(y_k), \qquad (13)$$

де $g_{\varphi_k}(y_k) = B_k^T g_f(x_k)$, а величина кроку h_k обирається з умов:

a) на відрізку $\left[0, h_k\right]$ функція $\varphi_k(h) = f(x_{k+1}(h))$ не зростає, (13.a)

b) існує $g \in G_f(x_{k+1})$ такий, що $\dfrac{\left(B_k^T g\right)^T g_{\varphi_k}(y_k)}{\left\|B_k^T g\right\| \left\|g_{\varphi_k}(y_k)\right\|} \le \alpha$. (13.b)

2. Перерахуємо матрицю перетворення простору

$$B_{k+1} = B_k R\ (¦_k) = B_k + (\quad -1)(B_k ¦_k) ¦_k^T , \tag{14}$$

де

$$¦_k = \frac{B_k^T r_k}{\| B_k^T r_k \|}, \quad r_k = g_f(x_{k+1}) - g_f(x_k), \qquad = \frac{1}{\sim} < 1. \tag{15}$$

3. Здійснимо перехід до наступної ітерації з x_{k+1}, B_{k+1} та $g_f(x_{k+1}) = g$.

Зауваження. Якщо обидві частини формули (13) помножити зліва на матрицю $A_k = B_k^{-1}$, то отримаємо

$$y_{k+1} = A_k x_{k+1} = A_k x_k - °_k g_{¨_k}(y_k) = y_k - °_k g_{¨_k}(y_k).$$

Тож формула (13) фактично реалізує крок субградієнтного спуску для функції $¨_k(y) = f(B_k y)$, де $°_k$ – невід’ємний крок на k-ій ітерації (може дорівнювати нулю). При $« = 0$ величина $°_k^*$ – це крок найшвидшого спуску в напрямку антисубградієнта. Він пов’язаний з h_k^* (кроком найшвидшого спуску в напрямку нормованого антисубградієнта) за формулою $h_k^* = °_k^* \left\| B_k^T g_f(x_k) \right\|$.

Введення константи $« \geq 0$ дає можливість розглянути алгоритми, в яких пошук мінімуму за напрямком відбувається приблизно і так, щоб наступна точка знаходилася ближче, ніж точка мінімуму функції за напрямком. Зауважимо, що в r-алгоритмах із розділу 1 це було неможливим.

Справді, якщо $« \geq 0$, то умова (13.b) означає обмеження на косинус гострого кута між двома послідовними субградієнтами, за різницею яких реалізується розтяг простору.

Справедлива наступна теорема.

Теорема 3. Нехай $f(x)$ – функція виду (12), така, що $\lim\limits_{\|x\| \to \infty} f(x) = +\infty$, і послідовність $\{x_k\}_{k=0}^{\infty}$, що генерується $r_«(\sim)$-алгоритмом, задовольняє умові

$$\lim_{k \to \infty} \left\| x_{k+1} - x_k \right\| = 0. \tag{16}$$

Якщо x^* – ізольована точка мінімуму, а точка x_0 така, що опукла множина $\left\{x : f(x^*) \le f(x) \le f(x^0)\right\}$, яка містить x_0 і x^*, не містить, крім x^*, інших точок z, в яких сімейство $G_f(z)$ лінійно залежне, то послідовність $\{x_k\}_{k=0}^{\infty}$ збігається до точки x^*.

Проблема обґрунтування збіжності $r_«(\sim)$-алгоритмів для всього класу опуклих функцій наразі є відкритою. Одна з причин, через яку для негладких функцій важко довести $r_«(\sim)$-алгоритми, пов'язана з неоднозначним вибором антисубградієнта для наступного напрямку руху з точок негладкості, де субградієнти (з множини G_f) лінійно залежні та жоден із антисубградієнтів не є напрямком спадання функції. Для опуклої кусково-лінійної функції такі точки негладкості можуть бути пастками для мінімізуючої послідовності розглянутого варіанта $r_«(\sim)$-алгоритму [11].

У роботі [12] відзначені шляхи для обґрунтування $r_«(\sim)$-алгоритмів. Тут розглядаються монотонні модифікації r-алгоритмів, для яких за критерій зупинки обрана необхідна та достатня умова оптимальності для опуклих функцій – $0 \in \partial f(x)$, де $\partial f(x)$ – субдиференціал. $r_«(\sim)$-Алгоритм доповнюється наступним правилом. Якщо в даній точці мінімум за напрямком реалізується при нульовому кроці, то для наступної ітерації напрямок руху в перетвореному просторі обирається аналогічно найшвидшому спуску для опуклих функцій.

Модифікований вищенаведеним способом $r_«(\sim)$-алгоритм для майже диференційовних кусково-гладких функцій завжди гарантує вихід з будь-якої точки, в якій існує напрямок спадання функції, і, крім того, дає можливість стверджувати, що отриманий оптимум, якщо найкоротшим до опуклої оболонки майже-градієнтів виявиться нульовий вектор.

Така схема приваблива в теоретичному плані, оскільки питання збіжності при цьому фактично зводяться до визначення існування в точці напрямку спадання функції. Але їх так само, як і $r_«(\sim)$-алгоритм, можна вважати «ідеалізованими», оскільки для реалізації найшвидшого спуску з високою точністю необхідно виконати велику кількість обчислень значень функції та її субградієнта.

У практичних варіантах r-алгоритмів крок h_k обирається так, щоб виконувалася нерівність $h_k > h_k^*$, де h_k^* відповідає мінімуму функції. Для них «наближений» пошук мінімуму функції спрямований на зменшення загальної кількості обчислень значень функції та її субградієнта і реалізується таким чином, щоб на одну ітерацію алгоритму припадало в середньому два-три таких обчислення. Однією з ефективних реалізацій такої стратегії є $r(\sim)$-алгоритм з адаптивним регулюванням кроку.

3. $r(\sim)$-АЛГОРИТМ З АДАПТИВНИМ КРОКОМ ТА ЙОГО ПРОГРАМНІ РЕАЛІЗАЦІЇ

Одним з ефективних є $r(\sim)$-алгоритм з адаптивним регулюванням кроку, де α – постійний коефіцієнт розтягу простору, а величина кроку h_k налаштовується при виконанні одночасного спуску в напрямку нормованого антисубградієнта в перетвореному просторі змінних. Налаштування кроку здійснюється за допомогою чотирьох параметрів: $h_0 > 0$ – величина початкового кроку (використовується на першій ітерації, на кожній наступній уточнюється); q_1 – коефіцієнт зменшення кроку ($q_1 \le 1$), якщо умова завершення спуску за напрямком виконується за один крок; q_2 – коефіцієнт збільшення кроку ($q_2 \ge 1$); через кожні n_h кроків одномірного спуску ($n_h > 1$) величина кроку буде збільшуватись у q_2 разів. Умова завершення спуску за напрямком виконується, як тільки буде знайдена точка x_{k+1}, для якої має місце умова

$$\left(x_{k+1} - x_k\right)^T g_{k+1}(x_{k+1}) \ge 0 . \qquad (17)$$

Умова (17) легко перевіряється, оскільки в силу додатності кроку вона рівносильна виконанню нерівності

$$\left(B_k B_k^T g_f(x_k)\right)^T g_f(x_{k+1}) \le 0 . \qquad (18)$$

Це означає, що кут між двома послідовними субградієнтами в перетвореному просторі змінних буде негострим. Нерівність (18) можна записати як умову

$$\left(B_k B_k^T g_f(x_k)\right)^T g_f(x_{k+1}) \le 0, \qquad (19)$$

звідки

$$\left(g_{\cdot\cdot}(y_k)\right)^T g_{\cdot\cdot}(y_{k+1}) \le 0, \qquad (20)$$

де $g_{\cdot\cdot}(y_k) = B_k^T g_f(x_k)$ і $g_{\cdot\cdot}(y_{k+1}) = B_k^T g_f(x_{k+1})$ є субградієнтами функції $\ddot{}\,(y) = f(B_k y)$ у точках $y_k = A_k x_k$ и $y_{k+1} = A_k x_{k+1}$ перетвореного простору змінних $y = A_k x$, де $A_k = B_k^{-1}$. Оскільки передбачається, що $\lim\limits_{\|x\|\to\infty} f(x) = +\infty$, то після скінченої кількості кроків адаптивного спуску в напрямку нормованого антисубградієнта обов'язково буде виконуватись умова завершення (19)–(20).

Ітеративний процес в $r(\sim)$-алгоритмі з адаптивним регулюванням кроку продовжується до виконання деякого критерію зупинки, де визначальну роль відіграють параметри £_x та £_g. Алгоритм зупиняється в точці x_{k+1}, якщо виконано умову $\|x_{k+1} - x_k\| \le \text{£}_x$ (зупинка за аргументом) або виконано умову $\|g_f(x_{k+1})\| \le \text{£}_g$ (зупинка за нормою субградієнта, використовується для гладких функцій). Крім того, використовуються ще дві умови зупинки: стандартна зупинка, якщо перевищено максимальну кількість ітерацій, та аварійна зупинка, яка сигналізує про те, що або функція $f(x)$ необмежена знизу, або початковий крок h_0 занадто малий і його необхідно збільшити. І хоча $r(\sim)$-алгоритм з адаптивним регулюванням кроку не гарантує монотонного спадання функції, проте, як показали експерименти, зростання функції відбувається достатньо рідко.

Адаптивний спосіб регулювання кроку дозволяє збільшувати точність пошуку мінімуму функції в процесі розрахунку, щоб кількість кроків за напрямком не перевищувала в середньому двох-трьох на одну ітерацію [13].

Якщо $r(\sim)$-алгоритм з адаптивним регулюванням кроку застосувати для мінімізації негладких функцій, то рекомендується наступний вибір параметрів: $\sim\, = 2 \div 4$, $h_0 = 1.0$, $q_1 = 1.0$, $q_2 = 1.1 \div 1.2$, $n_h = 2 \div 3$. Якщо відома оцінка відстані від початкової точки x_0 до точки мінімуму x^*, то початковий крок h_0 доцільно обирати близьким до $\| x_0 - x^* \|$. При мінімізації гладких функцій рекомендовані такі ж параметри, за виключенням q_1 ($q_1 = 0.8 \div 0.95$).

Це зумовлене тим, що додаткове зменшення кроку сприяє збільшенню точності пошуку мінімуму функції за напрямком, що при мінімізації гладких функцій дає вищу швидкість збіжності. При такому виборі параметрів, як правило, число спусків за напрямком рідко буде більшим двох, а за n кроків точність по функції поліпшується у три-п’ять разів.

Параметри зупинки $\pounds_x, \pounds_g \sim 10^{-6} \div 10^{-5}$ при мінімізації опуклої функції навіть суттєво яружної структури забезпечують знаходження точки x_k^* – наближення до точки x^*, для якої значення функції, достатньо близьке до оптимального ($\frac{f(x_k^*) - f(x^*)}{|f(x^*)| + 1} \sim 10^{-6} \div 10^{-5}$ – для негладких і $\frac{f(x_k^*) - f(x^*)}{|f(x^*)| + 1} \sim 10^{-12} \div 10^{-10}$ – для гладких функцій). Це підтверджується результатами чисельних тестових і реальних розрахунків.

У наш час $r(\sim)$-алгоритм з адаптивним регулюванням кроку і його модифікації реалізовані багатьма комп’ютерними програмами на мовах програмування Фортран, C, C++, C# і Octave. В їх основу покладені B-форми r-алгоритму. За допомогою економної B-форми (5)–(7) і її модифікацій розроблені комп’ютерні програми **ralg** (Фортран, C, C++), **ralgb4** (Фортран, Octave), **SolveOpt** (Фортран, C). На основі B-форми методу (1)–(3) розроблена комп’ютерна програма **ralgb5** (Фортран, Octave, C++ і C#). Для r-алгоритмів у H-формі комп’ютерні програми не отримали широкої практики в силу того, що виявилися неефективними для достатньо точного вирішення задач оптимізації.

Історично однією з перших була фортранівська програма **ralg** (автор – М.Г. Журбенко). У 70–80 рр. минулого століття програма **ralg** активно використовувалася для оптимізації негладких функцій в Інституті кібернетики та інших організаціях. Наприклад, активними користувачами програми були Д.І. Соломон (м. Кишинів, Інститут математики) і О.М. Кісєльова (Дніпропетровський університет), які захистили докторські дисертації під керівництвом Н.З. Шора. У 1990 роки програма **ralg** стала основою для фортранівської програми **ralgb4** (автор – П.І. Стецюк), в якій використовується модифікація r-алгоритму [14]. За допомогою програми **ralg** О.В. Кунцевич розробив комплекс програм **SolveOpt** (мови – Фортран та С), де використане ускладнене адаптивне регулювання кроку та критерії зупинки $| x_{k+1}^i - x_k^i | \le \text{¢}_x \, | x_{k+1}^i |$ і $| f(x_{k+1}) - f(x_k) | \le \text{¢}_f \, | f(x_{k+1}) |$ з заданими достатньо малими ¢_x і ¢_f [15].

В кінці минулого – на початку цього століття програма **ralg** стала основою для розробки комп'ютерних програм на мовах програмування С, С++ (автори – М.Г. Журбенко і О.П. Лиховид).

У 2007–2008 рр. була розроблена фортранівська програма **ralgb5** (автор – П.І. Стецюк), в якій використовується метод (1)–(3). Її назва пов'язана з тим, що в основу програми закладена B-форма r-алгоритму, яка потребує $5n^2$ арифметичних операцій множення на кожній ітерації. У 2010 році програма **ralgb5** була переписана мовою Octave (її код наведено в [16, с. 384–385]). При розв'язанні задач для тисячі і більше змінних Octave-програма **ralgb5** виявилася швидшою за однойменну фортранівську програму. Це обумовлено тим, що бібліотека BLAS (Basic Linear Algebra Subprograms) для мови Octave дозволяє швидше виконувати матрично-векторні операції в r-алгоритмах, ніж це дозволяють оптимізуючі опції компілятора Фортрана. На даний час програма ralgb5 реалізована О.П. Лиховидом на мовах С++, С# і використовується в програмних імплементаціях алгоритмів вирішення різного роду задач нелінійного програмування.

У 2016 році по аналогії з програмою **ralgb5** була розроблена Octave-програма **ralgb4** [17]. Вона використовує метод (5)–(7), який реалізує економну B-форму r-алгоритмів. Octave-програма **ralgb4** потребує $4n^2$ арифметичних операцій множення на кожній ітерації. Її обчислювальні властивості виявилися близькими до обчислювальних властивостей програми **ralgb5.** Octave-функції **ralgb4** і **ralgb5** можна використовувати як оптимізаційні ядра при реалізації на мові **Octave** алгоритмів вирішення задач нелінійного програмування. На їх основі легко розробляти оптимізаційні ядра мовою **MATLAB** для вирішення обчислювальних задач, які зводяться до проблем мінімізації негладких випуклих функцій або гладких випуклих функцій з яружною структурою поверхонь рівня.

Octave-функції **ralgb4** і **ralgb5** можна легко переписати на мовах **Фортран** і **C**, використовуючи бібліотеку базових підпрограм лінійної алгебри BLAS (Basic Linear Algebra Subprograms) або бібліотеку математичних прикладних програм IntelR Math Kernel Library (IntelR MKL), які оптимізовані під сучасні обчислювальні машини. Це дозволить значно пришвидшити методи для розв’язання громіздких задач (тисяча і більше змінних), наприклад, за рахунок використання обчислювальних потужностей графічних процесорів, які в декілька разів перевищують обчислювальні потужності класичних процесорів. Так, значне прискорення можна отримати використовуючи для розрахунків технологію CUDA на графічних прискорювачах. Це підтверджує гібридна реалізація r-алгоритму [18], де скорочення часу, яке досягається при розв’язанні задач з 1000–8000 змінними, варіюється від 14 до 18 разів.

За допомогою програмних реалізацій $r(\alpha)$-алгоритму з адаптивним кроком можна знаходити досить точні наближення до точки мінімуму опуклої функції. Якщо коефіцієнт розтягу простору вибрати таким, щоб він добре узгоджувався з параметрами адаптивного регулювання кроку в напрямку нормованого антисубградієнта в перетвореному просторі змінних, то для виконання одних і тих самих критеріїв зупинки можна значно скоротити кількість ітерацій і кількість обчислень значення функції і субградієнта (градієнта). Це залежить від конкретного виду функції, що мінімізується, ступеня її яружності та масштабу змінних.

4. OCTAVE-ФУНКЦІЯ RALGB5A

Про програму ralgb5a [19]. Програма **ralgb5a** є спрощеною (для зручності використання) версією програми **ralgb5** [16, с. 383–386], в якій використовується метод (1)–(3) [3]. Тут абревіатура «**b5»** означає, що в основу програми покладено r-алгоритм у B-формі, де коректується $n \times n$-матриця B, а кожна ітерація методу (1)–(3) вимагає $5n^2$ арифметичних операцій множення, які визначають обчислювальну трудоємність ітерації (операції додавання не враховуються через їх малий внесок у трудоємність ітерації). З них $3n^2$ операцій множення потрібно для обчислення векторів $B_k\text{¶}_k$, $B_k^T g_f(x_k)$ і $B_k^T r_k$ (множення матриці на вектор), а $2n^2$ операцій множення вимагає однорангова корекція матриці $B_{k+1} = B_k R\ (\text{¦}_k)$. Дійсно, корекція матриці B_{k+1} виконується за формулою

$$B_{k+1} = B_k R\ (\text{¦}_k) = B_k(I_n + (\quad -1)\text{¦}_k\text{¦}_k^T) = B_k + (\quad -1)(B_k\text{¦}_k)\text{¦}_k^T,$$

звідки легко бачити, що обчислення вектора $\text{¦} = B_k\text{¦}_k$ вимагає n^2 операцій множення, і стільки ж операцій множення вимагає побудова тимчасової однорангової матриці $\text{¦}\,\text{¦}_k^T$.

У програмі **ralgb5a** зафіксовані два найбільш часто використовувані параметри $q_2 = 1.1$ і $n_h = 3$. При цьому величина початкового кроку для чергової ітерації може максимально збільшуватися в 10^6 раз. У програмі **ralgb5a** використовується параметр **intp (int**erval for **p**rint**)**, який забезпечує друк інформації про хід процесу мінімізації через кожні **intp** ітерацій. Цей параметр дозволяє скоротити протокол роботи програми при мінімізації функції для сотень і тисяч змінних, коли кількість ітерацій оцінюється тисячами і десятками тисяч. Програма використовує параметри ε_x і ε_g для зупинки ітераційного процесу в точці $x_{k^*} \in [x_k, x_{k+1}]$, де $\|x_{k+1} - x_k\| \le \varepsilon_x$ (зупинка за аргументом), або $\|g_f(x_{k^*})\| \le \varepsilon_g$ (зупинка за нормою градієнта, яка використовується для гладких функцій). Використовуються також стандартна зупинка, якщо перевищено задану максимальну кількість ітерацій **maxitn**, та аварійна зупинка, яка сигналізує про те, що або функція $f(x)$ не є обмеженою знизу, або початковий крок h_0 занадто малий, і його треба збільшити.

Якщо ітераційний процес запускається зі стартової точки x_0, то параметри $r(\alpha)$-алгоритму рекомендується вибирати наступними: $\alpha \in [2,4]$, $q_1 = 1.0$ (для негладких функцій), $q_1 = 0.8 \div 0.95$ (для гладких функцій), $h_0 \approx \|x_0 - x^*\|$ – оцінка відстані від стартової точки x_0 до точки мінімуму x^*. Як правило, використовуються такі параметри зупинки: $\varepsilon_x \approx 10^{-6}$; $\varepsilon_g \approx 10^{-12}$; **maxitn** $\approx 20n$**.** Тут параметр ε_g використовується для гладких функцій, а параметр ε_x – для негладких функцій. Якщо програма **ralgb5a** завершує роботу за умовою $\|x_{k+1} - x_k\| \le \varepsilon_x = 10^{-8}$, то цього цілком достатньо, щоб на $14-15$ порядків зменшити різницю між знайденим рекордним значенням квадратичної функції f_r та її мінімальним значенням f^*.

Опис програми ralgb5a. Нижче наведемо короткий опис програми ralgb5a, який буде містити код Octave-функції ralgb5a та його застосування до тестового прикладу sabs(100,1.2), який полягає у мінімізації кусково-лінійної функції

$$f(x) = \sum_{i=1}^{100} (1.2)^{i-1} |x_i - 1|, \quad f^* = f(x^*) = 0, \quad x^* = (1,1,\ldots,1)^T, \tag{21}$$

де $|a|$ – абсолютна величина числа a. Функція (21) є яружною, оскільки коефіцієнти при $|x_i - 1|$, $i = 1,\ldots,100$ утворюють геометричну прогресію з показником $q = 1.2$, де мінімальний коефіцієнт дорівнює $(1.2)^0 = 1$, а максимальний – $(1.2)^{99} \approx 6.9015\text{e+}07$.

Код програми ralgb5a. Octave-функція ralgb5a знаходить x_r^* – наближення до точки мінімуму опуклої функції $f(x)$ від n змінних. Програма використовує Octave-функцію **function [f, g] = calcfg (x)**, яка обчислює значення функції $f = f(x)$ та її субградієнта $g = g_f(x)$ в точці x. Ця програма готується користувачем та може мати довільне ім’я, яке підтримує синтаксис Octave. Код програми, що включає і короткі англомовні коментарі для вхідних та вихідних параметрів, наведено нижче.

```
# Input parameters:
#    calcfg - name of the function calcfg(x) for calculation of f and g
#    x - the starting point, x0(1:n) (it is modified in the program)
#    alpha - the value of coefficient of space dilation
#    h0, q1 - parameters of the adaptive step adjustment
#    epsx, epsg, maxitn - stop parameters
#    intp - print information every intp iteration
# Output parameters:
#    xr - a minimum point, which was found by the program, xr(1:n)
#    fr - the value of the function f at the point xr
#    itn - the number of iterations used by the program
#    nfg - the number of function calcfg calls
#    istop - exit code (2 = epsg, 3 = epsx, 4 = maxitn, 5 = error)
function [xr,fr,itn,nfg,istop] = ralgb5a(calcfg,x,alpha,h0,q1,  #row001
                                         epsg,epsx,maxitn,intp);
itn = 0; B = eye(length(x)); hs = h0; lsa = 0; lsm = 0;         #row002
xr = x; [fr,g0] = calcfg(xr); nfg = 1;                          #row003
printf("itn %4d  f%15.6e  fr%15.6e  nfg %4d\n",itn,fr,fr,nfg);  #row004
if(norm(g0) < epsg) istop = 2;  return; endif                   #row005
for (itn = 1:maxitn)                                            #row006
   dx = B * (g1 = B' * g0)/norm(g1);                            #row007
   d = 1; ls = 0; ddx = 0;                                      #row008
   while (d > 0)                                                #row009
      x -= hs * dx; ddx += hs * norm(dx);                       #row010
      [f, g1] = calcfg(x); nfg ++;                              #row011
      if (f < fr)  fr = f; xr = x;  endif                       #row012
      if(norm(g1) < epsg) istop = 2;  return; endif             #row013
      ls ++; (mod(ls,3) == 0) && (hs *= 1);                     #row014
      if(ls > 500) istop = 5;  return; endif                    #row015
      d = dx' * g1;                                             #row016
   endwhile                                                     #row017
   (ls == 1) && (hs *= q1);  lsa=lsa+ls; lsm=max(lsm,ls);       #row018
   if(mod(itn,intp)==0)                                         #row019
     printf("itn %4d  f %14.6e  fr %14.6e", itn, f, fr);        #row020
     printf("  nfg %4d  lsa %3d  lsm %3d\n", nfg, lsa, lsm);    #row021
     lsa=0; lsm=0;                                              #row022
   endif                                                        #row023
   if(ddx < epsx) istop = 3;  return; endif                     #row024
   xi = (dg = B' * (g1 - g0) )/norm(dg);                        #row025
   B += (1 / alpha - 1) * B * xi * xi';                         #row026
   g0 = g1;                                                     #row027
endfor                                                          #row028
istop = 4;                                                      #row029
endfunction                                                     #row030
```

При мінімізації негладких функцій рекомендується вибирати: alpha = $2 \div 3$, $h0$ = 0, $q1$ = 0. При мінімізації гладких функцій рекомендується використовувати $q1$ = $0.8 \div 0.95$. При правильному підборі цих параметрів можна значно скоротити кількість ітерацій для виконання одних і тих самих критеріїв зупинки. Це залежить від конкретного виду функції, що мінімізується, ступеня її яружності та масштабу змінних.

Тестовий приклад. Для тестового прикладу використовується кусково-лінійна функція (21). Обчислення значення функції (21) та її субградієнта реалізовано Octave-функцією

```
function [f,g] = sabs(x)
global w
temp=x-ones(length(x),1); f=sum(abs(w.*temp)); g=w.*sign(temp);
endfunction
```

для якої значення коефіцієнтів w встановлюються за допомогою операторів

```
global w
n=100; temp=[0:(n-1)]'; w=2.**temp;
```

Ітераційний процес запускається зі стартової точки $x_0 = (0,0,\ldots,0)^T$, для якої значення функції $f(x_0) = 4.140899\text{e+}08$. Параметри $r(\alpha)$-алгоритма вибираються такими: $\alpha = 4$ (рекомендується $\alpha \in [2,4]$), $q_1 = 1.0$ (рекомендується для негладких функцій), $h_0 = 10$ (дорівнює $\|x_0 - x^*\|$ – відстані від стартової точки x_0 до точки мінімуму x^*). Використовуються параметри зупинки: $\varepsilon_x = 10^{-8}$, $\varepsilon_g = 10^{-12}$, **maxitn = 5000**. Тут параметр ε_g ролі не відіграє (він використовується для гладких функцій), а параметр ε_x вибраний таким, щоб программа **ralgb5a** закінчувала роботу за критерієм $\|x_{k+1} - x_k\| \le 10^{-8}$, чого цілком достатньо, щоб на 14–15 порядків зменшити різницю між знайденим рекордним значенням функції f_r та її мінімальним значенням $f^* = 0$.

Для вказаних параметрів головна Octave-програма має вигляд:

```
global w
n=100; temp=[0:(n-1)]'; w=2.**temp; # w(1,1), w(100,1),
x = zeros(n,1); alpha = 4.0, h0 = 10.0, q1 = 0,
epsx = e-8, epsg = e-12, maxitn = 5000, intp=500;
[xr,fr,itn,nfg,istop]=ralgb5a(@sabs,x,alpha,h0,q1,epsg,epsx,maxitn,intp);
printf("itn %4d  fr %23.15e  istop %d  nfg %4d\n", itn, fr, istop,nfg);
dx = norm(xr-ones(n,1)),
```

Протокол роботи програми. Обчислення проводилися на комп'ютері Pentium 3GHz в системі Windows7/32 за допомогою GNU Octave версії 3.6.4. Протокол роботи програми **ralgb5a** при **intp = 500** має вигляд**:**

```
alpha =  4  h0 =  10  q1 =  1
epsx = 0000e-008  epsg = 0000e-012  maxitn =  5000
itn    0  f  4.140899e+008  fr  4.140899e+008  nfg    1
itn  500  f  718525e+003  fr  273433e+003  nfg  532  lsa 531  lsm   4
itn 1000  f  409472e+000  fr  192802e+000  nfg 1032  lsa 500  lsm   1
itn 1500  f  258921e-003  fr  258921e-003  nfg 1532  lsa 500  lsm   1
itn 2000  f  422859e-006  fr  224438e-006  nfg 2032  lsa 500  lsm   1
itn 2046  fr  6.340398755873688e-007  istop 3  nfg 2078
dx = 9497e-008
```

Тут вхідні параметри $r(\sim)$-алгоритма зібрано у двох перших рядках. Із протоколу видно, що кількість ітерацій відповідає емпіричній оцінці, тобто, на одну ітерацію в середньому затрачено $2078/2046 < 3$ обчислень значення функції (21) та її субградієнта. Якщо вибрати параметр $q_1 = 0.95$, то кількість ітерацій зменшується до **920** при **1539** викликах функції **sabs**.

Програма **ralgb5a** працює під управлінням тих операційних систем, які допускають установку Open Source-пакета для математичних обчислень GNU Octave [20]. Програма використовує версії Octave 3.0.0 і вище. Для неї не потрібно ніяких спеціальних конфігурацій комп'ютера.

ВИСНОВКИ

У рамках сімейства субградієнтних методів з розтягом простору в напрямку різниці двох послідовних субградієнтів отримані достатньо ефективні реалізації r-алгоритмів. Розроблені модифікації r-алгоритмів можна використовувати при мінімізації опуклих негладких функцій з різних областей застосувань. Оскільки гладка функція з градієнтом, що дуже швидко змінюється, близька за своїми властивостями до негладкої функції, то r-алгоритми мають прискорену збіжність при оптимізації яружних гладких функцій. При мінімізації гладких функцій вони виявляються конкурентоспроможними з найбільш вдалими реалізаціями методів спряжених напрямків і методів квазіньютонівського типу.

За допомогою програмних реалізацій $r(\sim)$-алгоритму з адаптивним кроком можна знаходити достатньо точні наближення до точки мінімуму опуклої функції. Якщо коефіцієнт розтягу простору обрати таким, щоб він добре узгоджувався з параметрами адаптивного регулювання кроку в напрямку нормованого антисубградієнта в перетвореному просторі змінних, то для виконання одних і тих самих критеріїв зупинки можна значно скоротити кількість ітерацій і кількість обчислень функції та суградієнта (градієнта). Це залежить від конкретного виду мінімізуючої функції, ступеня її яружності та масштабу змінних [21, 22].

Однак, теоретичне обґрунтування r-алгоритмів проведене недостатньо повно. І хоча кількість ітерацій для знаходження мінімального значення f^* з точністю £ для опуклих функцій від n змінних емпірично оцінюється як $N = O\left(n \log \frac{1}{\varepsilon}\right)$, наразі актуальною виявилася цитата Н.З. Шора 1982 року: «Теория всего класса алгоритмов с растяжением пространства далека от совершенства. Нам кажется достаточно реалистичной целью построение такого алгоритма, который по своей практической эффективности не уступал бы r-алгоритму и был столь же хорошо обоснован, как метод эллипсоидов».

Дослідження в цьому напрямку активно тривають.

Список літератури

1. Шор Н.З. Методы минимизации недифференцируемых функций и их приложения: Автореф. дис. докт. физ-мат. наук. Киев, 1970. 44 с.
2. Шор Н.З., Журбенко Н.Г. Метод минимизации, использующий операцию растяжения пространства в направлении разности двух последовательных градиентов. *Кибернетика.* 1971. № 3. С. 51–59.
3. Стецюк П.И. Теория и программные реализации r-алгоритмов Шора. *Кибернетика и системный анализ*. 2017. № 5. С. 43–57.
4. Шор Н.З. Методы минимизации недифференцируемых функций и их приложения. Київ: Наукова думка, 1979. 200 с. (English transl.: Shor N.Z. Minimization Methods for Non-Differetiable Functions. Berlin: Springer-Verlag, 1985. 178 p.).
5. Shor N.Z. Nondifferentiable optimization and polynomial problems. Boston; Dordrecht; London: Kluwer Academic Publishers. 1998. 412 p.
6. Shor N.Z., Stetsyuk P.I. Lagrangian bounds in multiextremal polynomial and discrete optimization problems. *Journal of Global Optimization.* 2002. Vol. 23, N. P. 1–41.
7. Шор Н.З., Журбенко Н.Г., Лиховид А.П., Стецюк П.И. Развитие алгоритмов недифференцируемой оптимизации и их приложения. *Кибернетика и системный анализ.* 2003. № 4. С. 80–94.
8. Шор Н.З. Исследование сходимости метода градиентного типа с растяжением пространства в направлении разности двух последовательных субградиентов. *Кибернетика.* 1975. № 4. С. 48–53.
9. Шор Н.З. О классе почти-дифференцируемых функций и одном методе минимизации функций этого класса. *Кибернетика*. 1972. № 4. С. 65–70.
10. *Special issue* "Nonsmooth optimization and related topics", dedicated to the memory of professor Naum Shor, Optimization Methods and Software, 23:1, 3–4 (2008) DOI: 10.1080/10556780701652368
11. Стецюк П.И. К вопросу сходимости r-алгоритмов. *Кибернетика и системный анализ*. 1995. № 6. С. 173–177.
12. Шор Н.З. Монотонные модификации *r*-алгоритмов и их приложения. *Кибернетика и системный анализ.* 2002. № 6. С. 74–96.

13. Шор Н.З., Стеценко С.И. Квадратичные экстремальные задачи и недифференцируемая оптимизация. Киев: Наук. думка, 1989. 208 с.
14. Шор Н.З., Стецюк П.И. Использование модификации *r*-алгоритма для нахождения глобального минимума полиномиальных функций. *Кибернетика и системный анализ*. 1997. № 4. С. 28–49.
15. Kappel F., Kuntsevich A.V. An implementation of Shor's r-algorithm. *Computational Optimization and Applications*. 2000. Vol. 15, № 2. P. 193–205.
16. Стецюк П.И. Методы эллипсоидов и r-алгоритмы. Кишинэу: Эврика. 2014. 488 с.
17. Стецюк П.И. Субградиентные методы ralgb5 и ralgb4 для минимизации овражных выпуклых функций. *Вычислительные технологии*. 2017. № 2. С. 127–149.
18. Стецюк П.І., Хіміч О.М., Сидорук В.О. Реалізація *r*-алгоритму на графічних процесорах. *Компьютерная математика*. Киев: Ин-т кибернетики имени В.М. Глушкова НАН Украины. 2016. № 2. С. 100–109.
19. Стецюк П.И., Фишер А. *r*-Алгоритмы Шора и octave-функция ralgb5a. *Тези Міжнародної наукової конференції «Сучасна інформатика: проблеми, досягнення та перспективи розвитку», присвяченої 60-річчю заснування Інституту кібернетики імені В.М. Глушкова НАН України* (м. Київ, 13–15 грудня 2017 р.). К.: Ін-т кібернетики імені В.М. Глушкова НАН України, 2017. С. 143–146.
20. *Octave* [Электронный ресурс]: http://www.octave.org – Режим доступа: свободный.
21. Stetsyuk P.I. Shor's r-Algorithms: Theory and Practice. In: Optimization Methods and Applications: In Honor of the 80th Birthday of Ivan V. Sergienko. Ed. by S. Butenko, P.M. Pardalos, V. Shylo. Springer. 2017. P. 495–520.
22. Лиховид А.П., Ивличев А.В. Результаты тестирования двух реализаций *r*-алгоритма. *Теорія оптимальних рішень*. Київ: Ін-т кібернетики імені В.М. Глушкова НАН України, 2018 . С. 42–48.

ЗМІСТ

ПРОГРАМНІ РЕАЛІЗАЦІЇ *r*-АЛГОРИТМІВ ШОРА

Робочі матеріали.

Випуск 19-01

Матеріали подано в авторській редакції.

СТЕЦЮК Петро Іванович,
доктор фізико-математичних наук,
завідувач відділу методів негладкої оптимізації
Інституту кібернетики імені В.М. Глушкова НАН України,
stetsyukp@gmail.com

БЄЛИХ Тамара Василівна,
кандидат технічних наук,
інженер-програміст І категорії, відділ методів негладкої оптимізації
Інституту кібернетики імені В.М. Глушкова НАН України,
krainaz@ukr.net

КРИВОРУЧКО Олена Ігорівна,
аспірантка Інституту кібернетики імені В.М. Глушкова НАН України.
helen.krivoruchko@gmail.com

Відповідальні за випуск:

СТЕЦЮК Петро Іванович,
E-mail: stetsyukp@gmail.com

ЛИХОВИД Олексій Петрович.
E-mail: o.lykhovyd@gmail.com

Відділ методів негладкої оптимізації,
E-mail: gicd120@gmail.com

Здано до друку 12 лютого 2019 року