УДК 519.853.6 DOI: 10.30748/soi.2024.179.03

В.М. Задачин

Харківський національний економічний університет ім. С. Кузнеця, Харків

КВАЗІ-НЬЮТОНОВСЬКИЙ КОМБІНОВАНИЙ МЕТОД ДЛЯ РОЗВ'ЯЗАННЯ ЗАДАЧ БЕЗУМОВНОЇ ОПТИМІЗАЦІЇ В MACHINE LEARNING

Представлено метод квазі-ньютоновського типу для розв'язання погано обумовлених або вироджених задач безумовної оптимізації, який є комбінацією двох методів: квазі-ньютоновського методу та методу спряжених градієнтів. На кожній ітерації весь простір представляється як ортогональна сума двох підпросторів на основі спектрального розкладання наближення гессіану по формулі BFGS та параметра "регуляризації" чисельного методу. На одному ортогональному підпросторі застосовується квазі-ньютоновський метод, а на іншому – метод спряжених градієнтів. На кожному ортогональному підпросторі застосовується окремий одномірний пошук по відповідному напрямку. Для зменшення трудомісткості процесу спектрального розкладання поточної матриці на кожній ітерації методу представлено алгоритм його перерахунку через попередній розклад. Ефективність представленого методу підтверджується чисельними експериментами, які були проведені на загальноприйнятих тестових функціях для задач безумовної оптимізації в порівнянні з процедурами оптимізації відомих математичних пакетів R, Python, Scilab, MATLAB. При чисельній реалізації методу в середовищі Python застосовувалось обчислення градієнта цільової функції за допомогою пакета TensorFlow.

Ключові слова: *безумовна оптимізація; квазі-ньютоновські методи; метод спряжених градієнтів; погано обумовлені та вироджені задачі; спектральне розкладання матриці; Machine Learning.*

Вступ

Постановка проблеми. Методи безумовної оптимізації мають велике значення в машинному навчанні. Зараз при навчанні нейронних мереж застосовуються в основному методи градієнтного спуску (наприклад, ADAM [1]), тому що вони менш трудомісткі та потребують менше пам'яті. Але у методів градієнтного спуску є суттєвий недолік – дуже низька швидкість збіжності [1–3], особливо для погано обмовлених задач. Значно більшу швидкість збіжності мають квазі-ньютонівські методи оптимізації, але вони потребують більше пам'яті та більшого об'єму обчислень, хоча, для доволі великих розмірностей, є варіанти квазі-ньютонівських методів з обмеженим застосуванням пам'яті (L-BFGS [1]). На думку автора, по мірі збільшення продуктивності обчислювальної техніки у майбутньому все ж таки будуть застосовуватись саме квазі-ньютонівські методи оптимізації, як у машинному навчанні взагалі, так для навчання нейронних мереж також [1; 4–11].

При розв'язанні практичних задач у машинному навчанні, наприклад, при налаштуванні регресійних нелінійних моделей, точка екстремуму обраного критерію оптимальності нерідко виявляється виродженою, що значно ускладнює її пошук. Тому саме вироджені задачі є найбільш складними в оптимізації. Є ще більший клас задач оптимізації, в яких цільова функція є погано обумовленою в околиці точки мінімуму. Ці задачі, хоча і не є формально виродженими, але існуючі чисельні методи теж мають при цьому невелику швидкість збіжності.

Аналіз останніх досліджень і публікацій. Відомі чисельні методи розв'язання загальної задачі безумовної оптимізації, до другого порядку включно, мають дуже низьку швидкість збіжності в разі розв'язання вироджених задач [2–3]. Це пояснюється тим, що для істотного підвищення швидкості збіжності в цьому випадку необхідно використання в методі похідних більш високого порядку, ніж другий [12; 34], але це робить чисельні методи дуже трудомісткими.

Незважаючи на те, що були розроблені досить ефективні ньютонівські та квазі-ньютонівські методи безумовної оптимізації [2–3], інтерес до них не згасає й досі [13–23]. При цьому в разі розв'язання погано обумовлених та вироджених задач безумовної оптимізації застосовується підхід, пов'язаний з регуляризацією чисельних методів [13–22]. Суть регуляризації чисельного методу ньютонівського чи квазі-ньютонівського типу полягає в тому, щоб зробити матрицю Гессе, або її наближення, додатньо визначеною. Це дозволяє працювати методу у виродженому випадку, але не надає можливості розв'язати задачу з великою точністю.

Також, у зв'язку з все більш зростаючим інтересом до методів машинного навчання, останні роки все більш уваги приділяється методам спряжених градієнтів [24–31]. Ці методи в Machine Learning є найперспективнішою альтернативою методам градієнтного спуску, що мають дуже низьку швидкість збіжності, особливо на погано обумовлених та вироджених задачах оптимізації.

Метою статті є розроблення ефективного методу квазі-ньютоновського типу для розв'язання погано обумовлених та вироджених задач безумовної оптимізації, ідея якого (на відміну від регуляризації) полягає в поділі всього простору на суму двох ортогональних підпросторів [32]. Поділ простору на кожній ітерації методу засновано на спектральному розкладанні матриці, що наближує гессіан цільової функції по формулі Broyden–Fletcher–Goldfarb–Shanno (BFGS) [2]. На кожному підпросторі є своя поведінка цільової функції, і тому на ньому застосовується відповідний метод мінімізації, а саме застосовується комбінація квазі-ньютоновського методу і методу спряжених градієнтів. Відмітимо, що замість методу спряжених градієнтів можна застосовувати і інші методи, наприклад, ADAM, градієнтний спуск і т.д. В роботі [33] розглядалася комбінація ньютонівського методу та методу градієнтного спуску, і тому дана стаття може розглядатися як практичний розвиток ідеї комбінації декількох методів при розв'язанні погано обумовлених та вироджених задач безумовної оптимізації. Ця ж ідея застосовувалась і в статті [34], де розглядалася комбінація ньютонівського методу та методу, в якому обчислюються похідні по напрямку 4-го порядку. В даній статті, додатково до [32–34], також розглянуті питання практичної реалізації цієї групи методів.

Виклад основного матеріалу

Розглядається задача безумовної оптимізації:

$$\min f(x),\ x \in R^n, \qquad (1)$$

де $f(x)$ – двічі диференційована функція, для якої є $x^* \in R^n$ – локальна точка мінімуму функції $f(x)$, матриця Гессе $f^{(2)}\left(x^*\right)$ або вироджена ($rank\left(f^{(2)}\left(x^*\right)\right) < n$), або погано обумовлена ($cond\left(f^{(2)}\left(x^*\right)\right) \gg 1$).

1. Квазі-ньютоновський комбінований метод

Квазі-ньютоновський комбінований метод для розв'язання задачі (1) будує ітераційну послідовність наближень точки мінімуму за формулою:

$$x^{(k+1)} = x^{(k)} + \alpha_{k1} u_1^{(k)} + \alpha_{k2} u_2^{(k)},\ k = 0,1,2,\ldots, \qquad (2)$$

де $x^{(0)}$ – початкове наближення точки мінімуму, $u_1^{(k)}, u_2^{(k)}$ – ортогональні вектори, $\alpha_{k1} > 0$, $\alpha_{k2} > 0$ – шагові множники вздовж, відповідно, напрямків $u_1^{(k)}, u_2^{(k)}$. Ортогональні вектори $u_1^{(k)}, u_2^{(k)}$ визначаються наступним чином.

На кожній k-й ітерації методу обчислюються вектор $g^k = f^{(1)}\left(x^k\right)$ та матриця H_k, що є наближенням матриці Гессе $f^{(2)}\left(x^k\right)$ за формулою BFGS [3]:

$$H_{k+1} = H_k + \frac{y_k y_k^T}{y_k^T s_k} - \frac{H_k s_k (H_k s_k)^T}{s_k^T H_k s_k},\ H_0 = I, \qquad (3)$$

де $s_k = x^{(k)} - x^{(k-1)}$, $y_k = g^{(k)} - g^{(k-1)}$, I – одинична матриця.

Так як матриця H_k є симетричною, то відповідно до спектрального розкладання її можна представити у вигляді

$$H_k = Q_k \Lambda_k Q_k^T, \qquad (4)$$

де Q_k – ортогональна матриця, $\Lambda_k = diag\left(\lambda_i^{(k)}\right)$, а $\lambda_i^{(k)} (i = 1,\ldots,n)$ – власні значення матриці H_k, що впорядковані за спаданням за абсолютною величиною.

Представимо тепер діагональну матрицю Λ_k в блочному вигляді

$$\Lambda_k = \begin{bmatrix} \Lambda_{k1} & 0 \\ 0 & \Lambda_{k2} \end{bmatrix},$$

де $\Lambda_{k1} = diag\left(\lambda_i^{(k)}\right)$, $\left|\lambda_i^{(k)}\right| / \left|\lambda_1^{(k)}\right| > \varepsilon_k\ (i = 1,\ldots,r_k)$, $r_k \le n$, $\varepsilon_k > 0$ – параметр чисельного методу на k-й ітерації;

$$\Lambda_{k2} = diag\left(\lambda_i^{(k)}\right), \left|\lambda_i^{(k)}\right| / \left|\lambda_1^{(k)}\right| \le \varepsilon_k,$$
$$(i = (r_k + 1),\ldots,n).$$

Тоді матрицю Q_k також запишемо у блочному вигляді $Q_k = \begin{bmatrix} Q_{k1} & Q_{k2} \end{bmatrix}$, де Q_{k1} – блок розмірності $n \times r_k$, а Q_{k2} – блок розмірності $n \times (n - r_k)$.

Тепер матрицю H_k у зв'язку з (4) можна представити наступним чином:

$$H_k = \begin{bmatrix} Q_{k1} & Q_{k2} \end{bmatrix} \begin{bmatrix} \Lambda_{k1} & 0 \\ 0 & \Lambda_{k2} \end{bmatrix} \begin{bmatrix} Q_{k1}^T \\ Q_{k2}^T \end{bmatrix} =$$
$$= Q_{k1} \Lambda_{k1} Q_{k1}^T + Q_{k2} \Lambda_{k2} Q_{k2}^T = H_{k\varepsilon} + E_{k\varepsilon},$$

де $H_{k\varepsilon} = Q_{k1} \Lambda_{k1} Q_{k1}^T$, $E_{k\varepsilon} = Q_{k2} \Lambda_{k2} Q_{k2}^T = H_k - H_{k\varepsilon}$.

Далі будуємо ортогональні проектори: $P_k = I - Q_{k1} Q_{k1}^T$, $P_k^{\perp} = I - P_k = Q_{k1} Q_{k1}^T$ на підпростір $Ker(H_{k\varepsilon}) = \left\{x \in R^n \mid H_{k\varepsilon} x = 0\right\}$ та ортогональне доповнення до нього, відповідно. Відмітимо, що з ортогональності матриці Q_k випливає, що $P_k = Q_{k2} Q_{k2}^T$. Тепер на підпросторах $Ker(H_{k\varepsilon})$ і ортогональному доповненню до нього можна засто-

сувати різні методи оптимізації в залежності під поведінки функції $f(x)$ на них. Відмітимо, що при цьому параметр чисельного методу $\varepsilon_k > 0$ є деяким критерієм, за яким виконується розподіл простору на ортогональну суму двох підпросторів на кожній ітерації. Алгоритм підбору цього параметру буде описано нижче.

В методі (2) на ортогональному доповненню до підпростору $Ker(H_{k\varepsilon})$ застосуємо квазі-ньютоновський алгоритм, тобто вектор $u_1^{(k)}$ будемо обчислювати за формулою:

$$u_1^{(k)} = -H_{k\varepsilon}^{+} P_k^{\perp} g^{(k)} = -Q_{k1} \Lambda_{k1}^{-1} Q_{k1}^T P_k^{\perp} g^{(k)}, \quad (5)$$

де $H_{k\varepsilon}^{+}$ – псевдообернена матриця до матриці $H_{k\varepsilon}$.

На підпросторі $Ker(H_{k\varepsilon})$ застосуємо метод спряжених градієнтів, тобто вектор $u_2^{(k)}$ будемо визначати за формулою:

$$u_2^{(k)} = \begin{cases} -P_k g^{(k)}, & \text{при } k\%n = 0; \\ -P_k g^{(k)} + \beta_k u_2^{(k-1)}, & \text{при } k\%n > 0; \end{cases} \quad (6)$$

де β_k обчислюється за формулою CD [24]:

$$\beta_k = \beta_k^{CD} = \frac{P_k g^{(k)2}}{-\left(u_2^{(k-1)}\right)^T \left(P_k g^{(k)}\right)} \quad (7)$$

Обчислювання крокових множників. В методі (2), (5)–(7) треба визначати два крокових множника: $\alpha_{k1} > 0$ та $\alpha_{k2} > 0$, вдовж, відповідно, напрямків $u_1^{(k)}$ та $u_2^{(k)}$, тобто в різних ортогональних підпросторах. З одного боку це потребує додаткових обчислень, а з іншого, це надає можливість просування методу на підпросторі ядра матриці $H_{k\varepsilon}$. Так як матриця H_k є наближенням матриці Гессе $f^{(2)}\left(x^{(k)}\right)$, то в і напрямках $u_1^{(k)}$ та $u_2^{(k)}$ є відхилення від оптимальних, тому у випадку одночасного пошуку по напрямку $u^{(k)} = u_1^{(k)} + u_2^{(k)}$ напрямок $u_1^{(k)}$ буде домінуючим, і просування по напрямку $u_2^{(k)}$ буде дуже малим, або і зовсім відсутнім.

Як показали результати чисельних експериментів, найбільш ефективним алгоритмом визначення крокових множників $\alpha_{k1} > 0$, $\alpha_{k2} > 0$ виявився алгоритм, що є аналогічним методу покоординатного спуску [1]. Тобто, спочатку визначається кроковий множник $\alpha_{k1} > 0$, як точка наближеного мінімуму функції $\phi_1(\alpha) = f(x^{(k)} + \alpha\, u_1^{(k)})$, а потім – множник $\alpha_{k2} > 0$, як точка наближеного мінімуму функції $\phi_2(\alpha) = f(x^{(k)} + \alpha_{k1} u_1^{(k)} + \alpha\, u_2^{(k)})$.

Урахування негативних власних значень матриці $H_{k\varepsilon}$. У випадку, коли деякі елементи діагональної матриці Λ_{k1} є негативними, можна застосувати наступний алгоритм для визначення вектора $u_1^{(k)}$ замість формули (5).

Матриця Λ_{k1} розкладається на суму діагональних матриць $\Lambda_{k1} = \Lambda'_{k1} + \Lambda''_{k1}$, де у матриці Λ'_{k1} залишаються тільки додатні елементи, а у матриці Λ''_{k1} – тільки негативні елементи.

Тоді вектор $u_1^{(k)}$ визначається за формулою:

$$u_1^{(k)} = u_{11}^{(k)} + \rho_k\, u_{12}^{(k)}, \quad (8)$$

де

$$u_{11}^{(k)} = -Q_{k1} \Lambda'^{-1}_{k1} Q_{k1}^T P_k^{\perp} g^{(k)},$$
$$u_{12}^{(k)} = Q_{k1} \Lambda''^{-1}_{k1} Q_{k1}^T P_k^{\perp} g^{(k)}, \quad (9)$$

а параметр $\rho_k > 0$ визначається як точка наближеного мінімуму функції $\phi_3(\alpha) = f(x^{(k)} + \alpha\, u_{12}{}^{(k)})$.

Перерахунок наближення матриці Гессе за формулою BFGS. Найбільш трудомісткою процедурою в описаному квазі-ньютоновському комбінованому методі (2), (5)–(7) є спектральне розкладання (4) матриці H_k. Тому для зменшення обчислень при програмній реалізації методу можна застосувати наступний алгоритм, який об'єднує в собі і спектральне розкладання (4), і перерахунок наближення матриці Гессе за формулою BFGS (3).

На початку ітераційного процесу $H_0 = I$, тобто маємо $H_0 = Q_0 \Lambda_0 Q_0^T$, де $\Lambda_0 = I$, $Q_0 = I$. Таким чином, на k–й ітерації є матриці Λ_k, Q_k, що відповідають спектральному розкладанню (4). Далі обчислюємо Λ_{k+1}, Q_{k+1}, що відповідають розкладанню $H_{k+1} = Q_{k+1} \Lambda_{k+1} Q_{k+1}^T$, наступним чином.

З ортогональності матриці Q_k та формул (3), (4) випливає:

$$H_{k+1} = Q_k \Lambda_k Q_k^T + Q_k Q_k^T \frac{y_k y_k^T}{y_k^T s_k} Q_k Q_k^T - Q_k Q_k^T \times$$
$$\times \frac{H_k s_k (H_k s_k)^T}{s_k^T H_k s_k} Q_k Q_k^T =$$
$$= Q_k (\Lambda_k + Q_k^T \frac{y_k y_k^T}{y_k^T s_k} Q_k - Q_k^T \frac{H_k s_k (H_k s_k)^T}{s_k^T H_k s_k} Q_k) Q_k^T =$$
$$= Q_k (\Lambda_k + \frac{y'_k (y'_k)^T}{(y'_k)^T s'_k} - \frac{\Lambda_k s'_k (\Lambda_k s'_k)^T}{(s'_k)^T \Lambda_k s'_k}) Q_k^T, \quad (10)$$

де $y'_k = Q_k^T y_k$, $s'_k = Q_k^T s_k$. При цьому,

$$Q_k^T H_k s_k = Q_k^T Q_k \Lambda_k Q_k^T s_k = \Lambda_k Q_k^T s_k = \Lambda_k s'_k ,$$

$$\left(y'_k\right)^T s'_k = {y_k}^T s_k , \quad \left(s'_k\right)^T \Lambda_k s'_k = {s_k}^T H_k s_k .$$

Таким чином, якщо для матриці (10)

$$B = \Lambda_k + \frac{y'_k (y'_k)^T}{\left(y'_k\right)^T s'_k} - \frac{\Lambda_k s'_k (\Lambda_k s'_k)^T}{\left(s'_k\right)^T \Lambda_k s'_k} \qquad (11)$$

виконано спектральне розкладання $B = S\,\Lambda_{k+1} S^T$, то з (10) маємо, що

$$H_{k+1} = Q_k B Q_k^T = Q_k \left(S\,\Lambda_{k+1} S^T\right) Q_k^T = \\ = Q_{k+1} \Lambda_{k+1} Q_{k+1}^T , \qquad (12)$$

де $Q_{k+1} = Q_k\ S$.

Відмітимо, що для матриці B з (11) спектральне розкладання здійснити значно менш трудомістко, чим для матриці H_{k+1}, якби вона обчислювалась за формулою (3), тому, що в матриці B домінуючою складовою є діагональна матриця Λ_k, в якій діагональні елементи вже відсортовані за спаданням за абсолютною величиною.

Підбір параметра регуляризації квазі-ньютоновського комбінованого методу. Як вже відмічалось раніше, параметр "регуляризації" $\varepsilon_k > 0$ квазі-ньютоновського комбінованого методу (2), (5)–(7) є деяким критерієм, за яким виконується розподіл простору на ортогональну суму двох підпросторів на кожній k-й ітерації. Зрозуміло, що від його значення дуже сильно залежить хід ітераційного процесу. При чисельній реалізації методу (2), (5)–(7) було застосовано наступний алгоритм підбору параметра $\varepsilon_k > 0$.

Перед застосуванням методу задається дискретний діапазон можливих значень $\varepsilon_k > 0$, наприклад, з трьох значень: $\varepsilon_{\min} < \varepsilon_m < \varepsilon_{\max}$ (при чисельних експериментах бралось $10^{-11} < 10^{-7} < 10^{-3}$). Спочатку, для $k \geq 0$ береться $\varepsilon_k = \varepsilon_{\min}$. Потім, коли ітераційний процес методу (2), (5)–(7) сповільнюється, тобто $\left\| x^{(k+1)} - x^{(k)} \right\| / \left(1 + \left\| x^{(k+1)} \right\|\right) \leq tool_{arg}$, де $tool_{arg} > 0$ – заданий параметр (наприклад, $tool_{arg} = 10^{-10}$), ε_k збільшується таким чином, щоб на наступній ітерації ранг матриці $H_{(k+1),\varepsilon}$ став меншим, тобто $r_{k+1} < r_k$. Для цього достатньо взяти $\varepsilon_{k+1} = 2\dfrac{\left|\lambda_{r_k}^{(k)}\right|}{\left|\lambda_1^{(k)}\right|} \leq \varepsilon_m$. Знову продовжується ітераційний процес методу (2), (5)–(7). Коли ітераційний процес сповільнюється, треба перейти до більшого значення ε_m, якщо їх декілька. Повністю ітераційний процес методу (2), (5)–(7) завершується, коли на k–й ітерації $\dfrac{\left|\lambda_{r_k}^{(k)}\right|}{\left|\lambda_1^{(k)}\right|} > \varepsilon_{\max}$. З цього алгоритму видно, що якщо $cond(H_k) < 1/\varepsilon_{\max}$, то описаний метод співпадає зі звичайним квазі-ньютоновським методом.

2. Чисельна реалізація методу та результати чисельних експериментів

Чисельна реалізація запропонованого методу. Квазі-ньютоновський комбінований метод (2), (5)–(7) було чисельно реалізовано в середовищі Python.

Для стійкості методу до розходження у випадку від'ємних діагональних елементів $\lambda_i^{(k)}$ $(i \leq r_k)$ матриці Λ_{k1} в (5) вектор $u_1^{(k)}$ (замість формули (5)) обчислювався за формулами (8), (9).

Обчислювання крокових множників проводилось за алгоритмом, що було описано вище в розділі 1.

При обчисленні наступного наближення матриці Гессе за формулою BFGS виконувався перерахунок поточної матриці за алгоритмом (10)–(12).

Підбір параметра регуляризації квазі-ньютоновського комбінованого методу виконувався за алгоритмом, що було описано в розділі 1.

Обчислення градієнта. Похідні $g^{(k)} = f^{(1)}\left(x^{(k)}\right)$ обчислювались наступним чином.

Якщо на попередній ітерації $\left\| g^{(k-1)} \right\| > 10^{-3}$, то градієнт $g^{(k)} = f^{(1)}\left(x^{(k)}\right)$ обчислювався чисельно за симетричною формулою [1–2]:

$$\frac{\partial f(x)}{\partial x_i} \approx \frac{f(x \mid x_i + h_i) - f(x \mid x_i - h_i)}{2h_i}$$

з кроком чисельного диференціювання

$$h_i = h_0 \max\left(1, \left|x_i\right|\right), \text{ де } h_0 = 10^{-7} .$$

В іншому випадку, тобто, якщо $\left\| g^{(k-1)} \right\| \leq 10^{-3}$, градієнт $g^{(k)} = f^{(1)}\left(x^{(k)}\right)$ обчислювався за допомогою пакета TensorFlow.

Такий підхід було застосовано тому, що хоча TensorFlow обчислює градієнт точніше, але витрачає на це більше часу. Точність обчислення градієнта якраз важливіша в околиці точки мінімуму, бо від неї залежить і точність обчислення матриці H_k, що є наближенням гессіану.

Результати чисельних експериментів. Метод (2), (5)–(7) було протестовано на тестових функціях

для задач безумовної оптимізації з колекції, представленої в роботі [35]. Всього в цій колекції представлено 75 функцій, тому для тестування методу були відібрані тільки задачі оптимізації або з виродженою точкою мінімуму, або з великим числом обумовленості (*cond*) матриці $f^{(2)}\left(x^*\right)$. Нумерація застосованих тестових функцій в нижче наведених табл. 1–13 відповідає нумерації в колекції [35].

Додатково для тестування були взяті ще наступні функції:

77. **Середнє-квадратична апроксимація за допомогою поліномів**:

$$f(x)=\sum_{j=1}^{101}\left[\sum_{i=1}^{n}x_i\,0{,}01(j-1)^{i-1}-\sum_{i=1}^{n}x_i^*\,0{,}01(j-1)^{i-1}\right]^2,$$

початкове наближення – $x^{(0)}$ = (2, 2, …, 2), точка мінімуму – x^* = (1, 1, …, 1), значення цільової точки в точці мінімуму – $f\left(x^*\right)=0$, $rank\left(f^{(2)}\left(x^*\right)\right)=n-1$.

78. **Моя функція 1**:

$$f(x)=1000\left(x_1-1000\right)^2+0{,}001x_2^4+\sum_{i=3}^{n}\left(x_i-i\right)^2,$$

початкове наближення – $x^{(0)}$ = (100, …, 100), точка мінімуму – x^* = (1000, 0, 3, 4,…, *n*), значення цільової точки в точці мінімуму – $f\left(x^*\right)=0$, $rank\left(f^{(2)}\left(x^*\right)\right)=n-1$.

79. **Моя функція 2**:

$$f(x)=x_1^2+x_1x_2^2+x_2^4+\sum_{i=3}^{n}x_i^2,$$

початкове наближення $x^{(0)}$ = (10, 14, 10, …, 10), точка мінімуму x^* = (0, 0, …, 0, 0), значення цільової точки в точці мінімуму – $f\left(x^*\right)=0$, $rank\left(f^{(2)}\left(x^*\right)\right)=n-1$.

Серед розглянутих 79 функцій задача (1) є виродженою ($rank\left(f^{(2)}\left(x^*\right)\right)<n$) для функцій: 15, 21, 38, 45, 58, 78, 79, а погано обумовленою ($cond\left(f^{(2)}\left(x^*\right)\right)\gg 1$) – для функцій: 3, 4, 17, 37, 46, 77.

Чисельні експерименти майже для всіх функцій проводились для розмірності $n=4$ та $n=100$, тільки для функції 77 – для розмірності $n=5$ та $n=100$. Розмірності 4 та 5 взяті тому, що це класичні розмірності для цих функцій. Розмірність 100 взята для аналізу продуктивності процедур оптимізації на задачах з не низькою розмірністю, бо на простих і складних задачах, як правило, результати дуже сильно відрізняються.

Порівняння проводилось з процедурами квазіньютоновського типу математичних пакетів: R, Python, Scilab, MATLAB. В табл. 1–13 застосовуються наступні позначення цих процедур:

R – процедура optim математичного пакету R версії 4.1.3 (метод 'BFGS');

Py – процедура minimize пакету scipy версії 1.11.4 Python (метод 'BFGS');

Sc – процедура optim математичного пакету Scilab версії 6.1.0 (метод 'gn');

M – процедура fminunc математичного пакету MATLAB Online (метод 'quasi-newton');

QN&CG – процедура, що реалізує метод (2), (5)–(7) в Python.

Результати чисельних експериментів наведені в табл. 1–13, де:

Dx – евклідова норма $\left\|\tilde{x}-x^*\right\|$, де $\tilde{x}$ – отримане процедурою оптимізації наближення розв'язку задачі; *Df* = $\left|f\left(\tilde{x}\right)-f\left(x^*\right)\right|$; *Nit* – виконана кількість ітерацій; *Nf* – виконана кількість обчислень цільової функції; *Ngr* – виконана кількість обчислень градієнта цільової функції; *NormGr* – евклідова норма $f^{(1)}(\tilde{x})$; *code* – код закінчення обчислень, що повернула відповідна процедура оптимізації.

В процедурі QN&CG код закінчення обчислень *code* приймає значення: 0 – досягнута задана точність по градієнту (задавалося 10^{-20}); 1 – досягнута точність по аргументу (задавалося 10^{-10}); 4 – maximum number of iterations is reached (задавалося 3000).

В процедурі optim (пакет Scilab) код закінчення обчислень (err) приймає значення: 1 – "Norm of projected gradient lower than..."; 5 – "Optim stops: maximum number of iterations is reached"; 9 – "End of optimization, successful completion".

В процедурі scipy.minimize (Python) код закінчення обчислень (message) приймає значення: 0 – "Optimization terminated successfully"; 1 – "Warning: Desired error not necessarily achieved due to precision loss"; 2 – "Warning: CG iterations didn't converge. The Hessian is not positive definite"; 4 – "Warning: Maximum number of iterations has been exceeded".

В процедурі fminunc (пакет MATLAB) код закінчення обчислень (exitflag) приймає значення: 5 – "Predicted decrease in the objective function was less than the FunctionTolerance tolerance" (задавалося 10–40).

Результати чисельних експериментів представлені в табл. 1–13. Основним критерієм для порів-

няння ефективності роботи процедур тут виступає значення *Df*, бо цікавила саме максимальна можлива точність розв'язку задачі (1).

Як видно з табл. 1–13, найгірший результат на всіх тестових функціях дають процедура optim математичного пакету R та процедура fminunc математичного пакету MATLAB, тому є сенс порівнювати між собою лише процедури: minimize пакету scipy (Py), optim математичного пакету Scilab (Sc) та QN&CG.

На більшості тестовий функцій процедура optim математичного пакету Scilab (Sc) дає гірший результат, а на функції 46 при $n = 100$ і просто поганий. Також можна зауважити, що ця процедура і процесорного часу застосовує більше, ніж всі інші процедури, хоча в табл. 1–13 цей час і не приводиться, бо середовище для обчислень було різне.

Найкращий результат дають процедура minimize пакету scipy (Py) і процедура QN&CG.

Якщо порівнювати процедуру QN&CG з процедурою minimize пакету scipy (Py), то можна сказати, що процедура QN&CG дозволяє знайти достатньо точний розв'язок задачі за значно меншу кількість ітерацій і меншій кількості обчислень градієнта цільової функції. Правда, при цьому кількість обчислень значень цільової функції, як правило, більша.

Таблиця 1

Чисельні результати для функції **3** [35] ($cond(f^{(2)}(x^*)) \approx 2{,}1e6$)

	n=4							*n*=100						
	Dx	*Df*	*Nit*	*Nf*	*Ngr*	*NormGr*	*Code*	*Dx*	*Df*	*Nit*	*Nf*	*Ngr*	*NormGr*	*Code*
R	6,1e–04	7,6e–08		155	46	5,6e–02		3,1e–03	1,9e–06		631	222	0,27	
Py	6,3e–12	8,1e–24	62	90	80	2,6e–13	1	3,4e–11	2,3e–22	660	715	706	5,3e–12	1
Sc	8,2e–09	3,6e–17	72	593		2,0e–07	9	1,0e–07	2,6e–15	234	897		2,0e–07	9
M	1,0e–05	4,0e–11	43	350		1,0e–07	5	6,7e–05	9,1e–10	43	6161		6,3e–05	5
QN&CG	0,0	0,0	22	116	23	0,0	0	1,7e–13	8,9e–27	70	508	71	2,4e–12	1–

Джерело: розроблено автором.

Таблиця 2

Чисельні результати для функції **4** [35] ($cond(f^{(2)}(x^*)) \approx 4{,}1e12$)

	n=4							*n*=100						
	Dx	*Df*	*Nit*	*Nf*	*Ngr*	*NormGr*	*Code*	*Dx*	*Df*	*Nit*	*Nf*	*Ngr*	*NormGr*	*Code*
R	3,6e–03	1,3e–06		167	55	0,25		0,02	4,0e–05		10952	331	1,2	
Py	4,0e–11	1,6e–22	74	105	94	1,3e–13	1	2,2e–10	4,8e–21	100	1203	1193	7,2e–11	1
Sc	1,0e–07	1,6e–15	97	1045		1,3e–06	9	7,0e–07	5,4e–14	64	175		7,0e–07	9
M	3,9e–05	1,5e–10	36	215		2,3e–06	5	1,9e–04	3,8e–09	36	4747		2,7e–06	5
QN&CG	0,0	0,0	34	145	35	0,0	0	0,0	0,0	244	1090	245	0,0	1–

Джерело: розроблено автором.

Таблиця 3

Чисельні результати для функції **15** [35] ($rank(f^{(2)}(x^*) = n-1)$

	n=4							*n*=100						
	Dx	*Df*	*Nit*	*Nf*	*Ngr*	*NormGr*	*Code*	*Dx*	*Df*	*Nit*	*Nf*	*Ngr*	*NormGr*	*Code*
R	2,6e–04	9,8e–15		83	66	6,1e–07		4,3e–10	3,1e–25		43	32	1,8e–12	
Py	4,5e–09	8,6e–34	56	73	73	1,0e–21	0	6,0e–05	1,2e–18	48	63	61	1,4e–12	1
Sc	6,1e–09	2,9e–33	50	165		4,4e–16	9	1,4e–06	3,1e–25	274	3023		2,1e–15	9
M	1,0e–05	2,5e–16	25	1704		1,4e–08	5	5,0e–05	6,3e–15	25	3636		7,4e–08	5
QN&CG	6,5e–08	3,6e–29	34	192	35	2,2e–21	0	6,1e–08	2,0e–30	46	294	47	1,4e–22	0–

Джерело: розроблено автором.

Таблиця 4

Чисельні результати для функції **17** [35] ($cond(f^{(2)}(x^*)) \approx 9{,}2e6$)

	n=4							*n*=100						
	Dx	*Df*	*Nit*	*Nf*	*Ngr*	*NormGr*	*Code*	*Dx*	*Df*	*Nit*	*Nf*	*Ngr*	*NormGr*	*Code*
R	2,6e–04	4,7e–11		53	17	5,0e–03		4,3e–10	1,4		432	128	3,4 e–02	
Py	4,5e–09	6,7e–28	12	64	53	1,3e–13	1	6,0e–05	1,1e–25	124	181	173	5,0e–13	1
Sc	6,1e–09	9,4e–22	20	27		6,0e–15	9	1,4e–06	7,9e–20	129	290		7,9e–20	9

M	1,3e–08	4,0e–14	15	80		2,4e–11	5	5,0e–05	2,0e–12	44	5959		9,7e–07	5
QN&CG		6,1e–32	12	45	13	3,7e–15	1		4,0e–27	108	342	109	4,9e–13	1–

Джерело: розроблено автором.

Таблиця 5

Чисельні результати для функції **21** [35] ($rank(f^{(2)}(x^*)) = n/2$)

	n=4							*n*=100						
	Dx	*Df*	*Nit*	*Nf*	*Ngr*	*NormGr*	*Code*	*Dx*	*Df*	*Nit*	*Nf*	*Ngr*	*NormGr*	*Code*
R	5,0e–04	2,7e–13		320	168	2,3e–06		1,5e–03	4,8e–13		1699	545	5,7e–06	
Py	7,4e–07	3,8e–25	95	112	100	2,0e–12	1	1,4e–05	8,8e–21	1850	1940	1928	5,2e–15	1
Sc	5,0e–09	6,3e–34	87	109		2,5e–23	1	4,6e–06	3,5e–23	356	492		3,9e–17	1
M	1,3e–03	1,1e–11	39	230		2,5e–05	5	6,7e–03	2,9e–10	39	4848		1,2e–04	5
QN&CG	5,2e–09	8,9e–34	84	633	85	1,3e–16	1	2,1e–06	1,9e–24	631	5886	632	6,7e–17	1

Джерело: розроблено автором.

Таблиця 6

Чисельні результати для функції **37** [35] ($cond(f^{(2)}(x^*)) \approx 2,8e3$)

	n=4							*n*=100						
	Dx	*Df*	*Nit*	*Nf*	*Ngr*	*NormGr*	*Code*	*Dx*	*Df*	*Nit*	*Nf*	*Ngr*	*NormGr*	*Code*
R	1,4e–05	9,1e–10		203	43	5,3e–02		1,0e–04	2,9e–08		1020	298	0,26	1
Py	2,0e–13	8,6e–26	92	135	129	1,6e–12	1	3,8e–12	8,0e–24	878	1002	993	5,1e–12	1
Sc	5,4e–10	1,2e–18	74	115		3,5e–09	9	3,5e–09	3,9e–17	86	178		1,0e–08	9
M	4,3e–07	5,7e–13	42	250		5,5e–12	5	2,1e–06	1,4e–11	55	6464		3,5e–06	5
QN&CG	0,0	0,0	43	172	44	0,0	0	1,4e–13	7,6e–27	287	1136	288	3,7e–13	1

Джерело: розроблено автором.

Таблиця 7

Чисельні результати для функції **38** [35] ($rank(f^{(2)}(x^*)) = n-1$)

	n=4							*n*=100						
	Dx	*Df*	*Nit*	*Nf*	*Ngr*	*NormGr*	*Code*	*Dx*	*Df*	*Nit*	*Nf*	*Ngr*	*NormGr*	*Code*
R	5,4e–09	1,3e–18		313	120	3,8e–08		8,4e–08	2,7e–16		1688	502	1,2e–07	
Py	0,0	0,0	114	153	153	0,0	0	2,4e–09	2,2e–19	935	1095	1090	1,9e–10	1
Sc	2,4e–13	2,2e–27	94	151		3,9e–12	9	5,8e–09	1,3e–18	644	971		4,7e–10	9
M	5,5e–06	3,8e–11	71	500		3,7e–04	5	6,0e–07	6,9e–10	103	12928		3,8e–06	5
QN&CG	0,0	0,0	14	154	15	0,0	0	5,9e–11	2,3e–22	36	421	37	1,0e–09	1

Джерело: розроблено автором.

Таблиця 8

Чисельні результати для функції **45** [35] ($rank(f^{(2)}(x^*)) \ll n$)

	n=4							*n*=100						
	Dx	*Df*	*Nit*	*Nf*	*Ngr*	*NormGr*	*Code*	*Dx*	*Df*	*Nit*	*Nf*	*Ngr*	*NormGr*	*Code*
R	2,6e–04	4,8e–15		62	48	3,1e–07		5,8e–04	1,1e–13		21346	3248	6,6e–07	
Py	1,9e–08	1,4e–31	142	218	206	2,9e–17	1	2,0e–05	1,1e–19	2562	2626	2614	3,0e–14	1
Sc	1,1e–13	1,5e–52	184	239		2,8e–39	1	4,1e–08	1,0e–30	3001	3097		1,1e–22	1
M	7,1e–04	1,5e–13	51	260		2,5e–09	5	1,3e–03	1,3e–12	226	23129		5,7e–09	5
QN&CG	6,6e–09	1,4e–33	58	435	59	3,9e–18	1	1,6e–06	4,5e–24	582	3148	583	1,5e–17	1

Джерело: розроблено автором.

Таблиця 9

Чисельні результати для функції **46** [35] ($cond(f^{(2)}(x^*)) \approx 3,6e3$)

	n=4							*n*=100						
	Dx	*Df*	*Nit*	*Nf*	*Ngr*	*NormGr*	*Code*	*Dx*	*Df*	*Nit*	*Nf*	*Ngr*	*NormGr*	*Code*
R		5,6e–12		145	19	7,0e–02			8,7e–09		1541	385	0,24	
Py		1,8e–25	16	56	48	1,2e–11	1		2,3e–24	126	237	228	7,6e–12	1
Sc		2,6e–06	17	53		4,4e–09	9		5963,8	76	717		1668,7	9

M		1,1e–13	11	80		2,5e–06	5		1,1e–11	186	21917		5,3e–06	5
QN&CG		1,5e–30	11	53	12	6,1e–14	1		2,4e–29	115	405	116	3.1e–13	1

Джерело: розроблено автором.

Таблиця 10

Чисельні результати для функції **58** [35] ($rank(f^{(2)}(x^*)) = n-1$)

	n=4							*n*=100						
	Dx	*Df*	*Nit*	*Nf*	*Ngr*	*NormGr*	*Code*	*Dx*	*Df*	*Nit*	*Nf*	*Ngr*	*NormGr*	*Code*
R	1,2e–02	1,9e–06		384	106	6,8e–02		9,6	1,4e–04		1231	373	8,9e–02	
Py	1,2e–10	1,9e–22	66	95	84	2,4e–13	1	9,2	7,1e–09	3000	3713	3713	3,2e–05	4
Sc	2,0e–07	9,2e–16	83	543		1,3e–06	9	9,2	5,8e–09	3001	3897		7,0e–07	5
M	2,0e–05	5,7e–11	62	445		2,8e–04	5	9,5	4,6e–06	372	49995		3,1e–03	5
QN&CG	0,0	0,0	51	190	52	0,0	0	9,2	3,4e–09	3000	15309	3001	3,0e–05	4

Джерело: розроблено автором.

Таблиця 11

Чисельні результати для функції **77** ($cond(f^{(2)}(x^*)) \gg 1$)

	n=5							*n*=100						
	Dx	*Df*	*Nit*	*Nf*	*Ngr*	*NormGr*	*Code*	*Dx*	*Df*	*Nit*	*Nf*	*Ngr*	*NormGr*	*Code*
R	5,2e–07	1,0e–16		582	357	1,2e–08		5,3e–05	6,6e–15		78	63	8,6e–10	
Py	1,8e–13	4,1e–29	35	78	72	2,9e–14	1	1,0e–03	9,5e–17	122	187	175	1,2e–12	1
Sc	2,3e–13	3,0e–29	23	49		3,0e–14	9	1,4e–05	5,6e–17	54	134		8,9e–12	9
M	1,3e–06	9,5e–15	50	306		1,5e–13	5	1,9e–04	6,9e–13		10100		7,4e–09	5
QN&CG	7,4e–16	5,0e–30	12	77	13	2,0e–14	1	3,5e–06	4,3e–19	33	254	34	7,4e–12	1

Джерело: розроблено автором.

Таблиця 12

Чисельні результати для функції **78** ($rank(f^{(2)}(x^*)) = n-1$)

	n=4							*n*=100						
	Dx	*Df*	*Nit*	*Nf*	*Ngr*	*NormGr*	*Code*	*Dx*	*Df*	*Nit*	*Nf*	*Ngr*	*NormGr*	*Code*
R	3,7e–03	2,0e–13		155	85	1,7e–09		8,0e–06	5,8e–22		67	52	4,8e–11	
Py	1,5e–08	5,2e–35	83	175	168	2,2e–16	1	9,5e–08	1,3e–30	79	159	153	8,4e–15	1
Sc	5,5e–11	9,2e–45	113	512		4,4e–16	9	1,7e–08	7,6e–27	89	182		1,7e–13	9
M	1,0e–03	5,5e–08	55	320		2,5e–07	5	1,0e–03	5,5e–08	55	6666		2,7e–10	5
QN&CG	6,9e–08	2,2e–28	31	213	32	1,3e–21	0	7,9e–08	4,0e–32	39	255	40	2,0e–24	0

Джерело: розроблено автором.

Таблиця 13

Чисельні результати для функції **79** ($rank(f^{(2)}(x^*)) = n-1$)

	n=4							*n*=100						
	Dx	*Df*	*Nit*	*Nf*	*Ngr*	*NormGr*	*Code*	*Dx*	*Df*	*Nit*	*Nf*	*Ngr*	*NormGr*	*Code*
R	1,1e–05	1,3e–20		75	57	4,6e–09		2,3e–08	1,2e–20		58	41	2,2e–10	
Py	1,6e–12	2,2e–41	109	261	261	1,2e–21	0	1,0e–14	1,0e–41	77	78	78	6,5e–21	0
Sc	3,0e–17	6,3e–67	174	190		1,9e–34	1	2,2e–16	1,1e–61	198	235		6,7e–31	1
M	6,7e–05	2,3e–16	62	315		5,3e–08	5	1,1e–04	3,6e–15	79	8686		2,4e–07	5
QN&CG	4,5e–09	3,6e–34	45	286	46	1,3e–17	1	1,3e–09	2,5e–36	38	253	39	2,1e–26	0

Джерело: розроблено автором.

Висновки

Представлено комбінований метод квазі-ньютоновського типу для розв'язання погано обмовлених або вироджених задач безумовної оптимізації, заснований на ортогональному розкладанні наближення матриці Гессе по формулі BFGS та поділі всього простору на два ортогональних підпростори. На одному підпросторі застосовується метод спряжених градієнтів, а на ортогональному доповненні до нього – квазі-ньютоновський метод. На кожному з цих підпросторів застосовується окремий одномірний пошук для визначення крокового множника у відповідному напрямку.

Відмітимо, що, в принципі, можна проводити поділ всього простору і на більшу, ніж два, кількість ортогональних підпросторів. Це залежить від співвідношення значень власних чисел матриці H_k на поточній ітерації. Потім на різних ортогональних підпросторах застосовується свій метод для визначення напрямку пошуку та окремий одномірний пошук. З одного боку, це потребує додаткових обчислень, а з іншого, це надає можливість просування методу на підпросторах, де цільова функція дуже повільно змінюється відносно інших підпросторів.

Ефективність представленого комбінованого методу підтверджується чисельними експериментами, які були проведені на загальноприйнятих тестових функціях для задач безумовної оптимізації. Потрібно відзначити, що запропонований метод дозволяє отримати досить точні розв'язки тестових завдань у разі погано обмовлених або вироджених задач безумовної оптимізації зі значно меншими затратами на обчислення градієнтів цільової функції, ніж це роблять процедури оптимізації відомих математичних пакетів.

У даній статті не проводилось теоретичне дослідження швидкості збіжності методу (2), (5)–(7) в разі розв'язання виродженої задачі (1) у зв'язку з обмеженням об'єму статті. Підкреслимо, що для цього можна застосувати достатні умови вищого порядку точки виродженого мінімуму, які представлені в роботі [36]. Однак, відмітимо, що, так як метод (2), (5)–(7) є комбінацією двох методів: методу Ньютона і методу спряжених градієнтів, то його швидкість збіжності буде не гірше, ніж швидкість збіжності методу градієнтного спуску, поведінка якого у випадку виродженого мінімуму досліджувалась в роботі [33].

Є сподівання, що по мірі збільшення продуктивності обчислювальної техніки у майбутньому при розв'язанні задач оптимізації в машинному навчанні будуть застосовуватись саме квазі-ньютонівські методи, так як вони мають значно більшу швидкість збіжності, ніж методи градієнтного спуску, особливо у разі погано обумовлених і вироджених задач.

Список літератури

1. Shastri A. S., Shaw K., Singh M. Machine Learning and Optimization for Engineering Design. New York : Springer, 2023. 178 p.
2. Avriel M. Nonlinear Programming: Analysis and Methods. Dover Publishing, 2003. 544 p.
3. Nocedal J., Wright S. J. Numerical Optimization. New York : Springer, 2006. 686 p.
4. Tirer T., Bruna J. Extended Unconstrained Features Model for Exploring Deep Neural Collapse. *arXiv* : web site. URL: https://arxiv.org/abs/2202.08087 (accessed 15.11.2024).
5. Ring W., Wirth B. Optimization methods on Riemannian manifolds and their application to shape space. *SIAM Journal on Optimization*. 2012. Vol. 22. No. 2. P. 596–627. https://doi.org/10.1137/11082885X.
6. Maratos N. G.., Moraitis M. A. Some results on the Sign recurrent neural network for unconstrained minimization. *Neurocomputing*. 2018. Vol. 287. P. 1–21. https://doi.org/10.1016/j.neucom.2017.09.036.
7. Anil R., Gupta V., Koren T., Regan K., Singer Y. Scalable Second Order Optimization for Deep Learning. *arXiv* : https://doi.org/10.48550/arXiv.2002.09018.
8. Sun R. Optimization for deep learning: theory and algorithms. *arXiv* : https://doi.org/10.48550/arXiv.1912.08957.
9. Berahas A. S., Jahani M., Richtárik P., Takáč M. Quasi-Newton Methods for Machine Learning: Forget the Past, Just Sample. *arXiv* . https://doi.org/10.48550/arXiv.1901.09997.
10. Wills A., Schön T. B., Jidling C. A fast quasi-Newton-type method for large-scale stochastic optimization. *IFAC-PapersOnLine*. 2020. Vol. 53. No. 2. P. 1249–1254. https://doi.org/10.1016/j.ifacol.2020.12.1849.
11. Krutikov V., Tovbis E., Stanimirović P., Kazakovtsev L., Karabašević D. Machine Learning in Quasi-Newton Methods. *Axioms.* 2024. Vol. 13. No. 4. Art. 240. https://doi.org/10.3390/axioms13040240.
12. Birgin E. G.., Gardenghi J. L., Martínez J. M., Santos S. A. On the use of third-order models with fourth-order regularization for unconstrained optimization. *Optimization Letters*. 2020. Vol. 14. P. 815–838. https://doi.org/10.1007/s11590-019-01395-z.
13. Li D.-H., Fukushima M., Qi L., Yamashita N. Regularized newton methods for convex minimization problems with singular solutions. *Computational Optimization Applications*. 2004. Vol. 28. P. 131–147. https://doi.org/10.1023/B:COAP.0000026881.96694.32.
14. Shen C., Chen X., Liang Y. A regularized Newton method for degenerate unconstrained optimization problems. *Optimization Letters.* 2011. Vol. 6. P. 1913–1933. https://doi.org/10.1007/s11590-011-0386-z.
15. Graspa T. N. A modified Newton direction for unconstrained optimization. *Optimization*. 2012. Vol. 63. No. 7. P. 983–1004. https://doi.org/10.1080/02331934.2012.696115.
16. Taheri S., Mammadov M., Seifollahi S. Globally convergent algorithms for solving unconstrained optimization problems. *Optimization*. 2012. Vol. 64. No. 2. P. 249–263. https://doi.org/10.1080/02331934.2012.745529.
17. Li X., Wang B., Hu W. A modified nonmonotone BFGS algorithm for unconstrained optimization. *Journal of Inequalities and Applications*. 2017. Art. 183. https://doi.org/10.1186/s13660-017-1453-5.
18. Ghazali K., Sulaiman J., Dasril Y., Gabda D. Newton-SOR Iteration for Solving Large-Scale Unconstrained Optimization Problems with an Arrowhead Hessian Matrices. *Journal of Physics: Conference Series*. 2019. Vol. 1358. Art. 012054. https://doi.org/10.1088/1742-6596/1358/1/012054.
19. Wang H., Qin M. A Regularized Newton Method with Correction for Unconstrained Nonconvex Optimization. *Journal of Mathematics Research*. 2015. Vol. 7. No 2. P. 7–17. https://doi.org/10.5539/jmr.v7n2p7.

20. Li Q. A Modified Fletcher-Reeves-Type Method for Nonsmooth Convex Minimization. *Statistics*. 2014. Vol. 2. No. 3. P. 200–210. https://doi.org/10.19139/soic.v2i3.64.

21. Andrei N. A new accelerated diagonal quasi-Newton updating method with scaled forward finite differences directional derivative for unconstrained optimization. *Optimization*. 2020. Vol. 70. No. 2. P. 345–360. https://doi.org/10.1080/02331934.2020.1712391.

22. Cartis C., Gould N. I. M., Toint Ph. L. A concise second-order complexity analysis for unconstrained optimization using high-order regularized models. *Optimization Methods and Software*. 2019. Vol. 35. No. 2. P. 243–256. https://doi.org/10.1080/10556788.2019.1678033.

23. Yang Y. A robust BFGS algorithm for unconstrained nonlinear optimization problems. *Optimization*. 2022. Vol. 73. No. 3. P. 851–873. https://doi.org/10.1080/02331934.2022.2124869.

24. Andrei N. Nonlinear Conjugate Gradient Methods for Unconstrained Optimization. New York : Springer, 2020. 526 p.

25. Wasi H. A., Shiker M. A. K. A new hybrid CGM for unconstrained optimization problems. *Journal of Physics: Conference Series.* 2020. Vol. 1664. Art. 012077. https://doi.org/10.1088/1742-6596/1664/1/012077.

26. Wasi H. A., Shiker M. A. K. Proposed CG method to solve unconstrained optimization problems. *Journal of Physics: Conference Series*. 2021. Vol. 1804. Art. 012024. https://doi.org/10.1088/1742-6596/1804/1/012024.

27. Ishak M. A. I. B., Marjugi S. M. B. A New Hybrid Three-Term HS-DY Conjugate Gradient In Solving Unconstrained Optimization Problems. *Applied Mathematics and Computational Intelligence (AMCI)*. 2024. Vol. 13. No. 1. P. 52–68. https://doi.org/10.58915/amci.v13iNo.1.493.

28. Andrei N. Hybrid Conjugate Gradient Algorithm for Unconstrained Optimization. *Journal of Optimization Theory and Applications*. 2009. Vol. 141. P. 249–264. https://doi.org/10.1007/s10957-008-9505-0.

29. Wang Z., Li P., Li X.**,** Pham H. A Modified Three-Term Type CD Conjugate Gradient Algorithm for Unconstrained Optimization Problems. *Mathematical Problems in Engineering*. 2020. Vol. 2020. No. 1. Art. 4381515. https://doi.org/10.1155/2020/4381515.

30. Dai Y.-H., Kou C.-X. A Nonlinear Conjugate Gradient Algorithm with an Optimal Property and an Improved Wolfe Line Search. *SIAM Journal on Optimization*. 2013. Vol. 23. No. 1. P. 296–320. https://doi.org/10.1137/100813026.

31. Yang Z. Adaptive stochastic conjugate gradient for machine learning. *Expert Systems with Applications*. 2022. Vol. 206. Art. 117719. https://doi.org/10.1016/j.eswa.2022.117719.

32. Задачин В. М. Комбінований метод для розв'язання вироджених задач безумовної оптимізації. *Системи обробки інформації.* 2020. № 1(160). С. 52–58. https://doi.org/10.30748/soi.2020.160.06.

33. Zadachyn V. M., Bebiya M. O. Combined Methods for Solving Degenerate Unconstrained Optimization Problems. *Ukrainian Mathematical Journal*. 2024. Vol. 76. No. 5. P. 695–718. https://doi.org/10.3842/umzh.v76i5.7395.

34. Zadachyn V. M. A Combined Quasi-Newton-type Method using 4th-order Directional Derivatives for Solving Degenerate Problems of Unconstrained Optimization. *Cybernetics and Computer Technologies*. 2024. No. 3. P. 12–24. https://doi.org/10.34229/2707-451X.24.3.2.

35. Andrei N. A Collection of 75 Unconstrained Optimization Test Functions. Technical Report No. 6/2018. *Center for Advanced Modeling and Optimization (CAMO)* : web site. URL: https://camo.ici.ro/neculai/TRR6.pdf (accessed 15.11.2024

36. Zadachyn V. M. Higher-order Optimality Conditions for Degenerate Unconstrained Optimization Problems. *Journal of Optimization, Differential Equations and Their Application.* 2022. Vol. 30. No. 1. P. 88–97. https://doi.org/10.15421/142204.

Надійшла до редколегії 27.12.2024
Схвалена до друку 28.02.2025

Відомості про автора:

Задачин Віктор Михайлович
кандидат фізико-математичних наук доцент
доцент
Харківського національного економічного
університету ім. С. Кузнеця,
Харків, Україна
https://orcid.org/0000-0002-8107-4639

Information about the author:

Viktor Zadachyn
PhD in Physical and Mathematical Sciences Associate Professor
Associate Professor
of Simon Kuznets Kharkiv
National University of Economics,
Kharkiv, Ukraine
https://orcid.org/0000-0002-8107-4639

QUASI-NEWTONIAN COMBINED METHOD FOR SOLVING UNCONSTRAINED OPTIMIZATION PROBLEMS IN MACHINE LEARNING

V. Zadachyn

We present a quasi-Newtonian method for solving ill-conditioned or degenerate unconditional optimization problems, which is a combination of two methods: the quasi-Newtonian method and the conjugate gradient method. At each iteration, the entire space is represented as an orthogonal sum of two subspaces based on the spectral decomposition of the Hessian approximation using the BFGS formula and the "regularization" parameter of the numerical method. The quasi-Newtonian method is used on one orthogonal subspace and the conjugate gradient method on the other. On each orthogonal subspace, a separate one-dimensional search in the corresponding direction is applied. To reduce the complexity of the process of spectral decomposition of the current matrix at each iteration of the method, an algorithm for its recalculation through the previous decomposition is presented. The effectiveness of the presented method is confirmed by numerical experiments carried out on common test functions for unconditional optimization problems in comparison with the optimization procedures of well-known mathematical packages R, Python, Scilab, MATLAB. In the numerical implementation of the method in Python, the calculation of the gradient of the objective function was used using the TensorFlow package.

Keywords: *conjugate gradient method; ill-conditioned and degenerate problems; Machine Learning; quasi-Newtonian methods; spectral decomposition of a matrix; unconstrained optimization.*