УДК 519.6

О.В. Касіцький

МЕТОД ПОКООРДИНАТНОГО СПУСКУ З ЕВРИСТИКОЮ СЕРЕДНЬОЗВАЖЕНОГО НАПРЯМКУ

In this paper, we propose the heuristic for the coordinate descent method. It dramatically enhances the method convergence on "bad" functions for it. Also, we describe the theoretical foundation of the heuristic effectiveness. Furthermore, we propose the effective C++ implementation of the coordinate descent method with average weighted heuristic. Through experiments conducted, we assess the quality of the method under study as compared to gradient descent methods and Rosenbrock method. We show the strengths and weaknesses of this method. Finally, we draw the conclusions on appropriateness of using the method under various conditions.

Вступ

У процесі проектування ставиться зазвичай задача визначення найкращих, у певному сенсі, структури або значення параметрів об'єктів. Задачею оптимізації в математиці, інформатиці та дослідженні операцій називається задача знаходження екстремуму (мінімуму або максимуму) цільової функції в деякій області скінченовимірного векторного простору, обмеженого набором лінійних або нелінійних рівностей чи нерівностей [1, 2].

Методи оптимізації класифікують відповідно до задач оптимізації.

Локальні методи: зводяться до якого-небудь локального екстремуму цільової функції. У разі унімодальної цільової функції цей екстремум єдиний і буде глобальним максимумом/мінімумом [1, 2].

Глобальні методи: мають справу з багатоекстремальними цільовими функціями. При глобальному пошуку основною задачею є виявлення тенденцій глобальної поведінки цільової функції [3, 4].

За критерієм вимірності допустимої множини методи оптимізації поділяють на методи одновимірної та методи багатовимірної оптимізації [1, 2, 4].

За вимогами до гладкості й наявності в цільовій функції частинних похідних їх можна розділити на такі:

- прямі методи, що вимагають тільки обчислень цільової функції в точках наближень;
- методи першого порядку: вимагають обчислення перших частинних похідних функції;
- методи другого порядку: вимагають обчислення других частинних похідних, тобто гессіана цільової функції.

Докладніше з класифікацією цільових функцій можна ознайомитися в [2].

Постановка задачі

Мета роботи полягає у розробці ефективного алгоритму для вирішення задачі багатовимірної оптимізації унімодальної цільової функції. Вважатимемо, що цільова функція диференційована, але аналітичного вигляду похідної не вимагаємо (за необхідності будемо використовувати різницеві похідні). Запропонований метод необхідно протестувати на наборі функцій з різними властивостями, що впливають на збіжність методів оптимізації властивостями (опуклі функцій, яристі функцій та ін.). Пропонується набір відомих тестових функцій з відомими на них результатами інших методів [5, 6]. Необхідно зробити висновки про доцільність використання алгоритму в різних умовах.

Вихідні положення

Нехай цільова функція має вигляд

$$F(\mathbf{x}) : R^n \to R \qquad (1)$$

і задача оптимізації задана в такий спосіб:

$$F(\mathbf{x}) \to \min_{\mathbf{x} \in R^n}. \qquad (2)$$

Існує багато методів розв'язку задачі (2) [1–3, 7]. У даній статті запропоновано евристику для методу покоординатного спуску і проведено аналіз отриманого алгоритму, виконано порівняння методу Розенброка [4] і градієнтних методів [3].

Метод покоординатного спуску

Основна ідея методу полягає в тому [7], що напрямок наступного кроку вибирається уздовж однієї з осей координат:

$$\mathbf{x}_{j+1} = \mathbf{x}_j + \lambda_j e_k ,$$

де $\{e_k\}$ – стандартний базис у R^n, λ_j може бути як додатною, так і від'ємною, її знак зазвичай чергується, а модуль може вибиратися дробовим (змінним), тобто довжина кроку в процесі спуску ділиться або множиться на деяке число, або за принципом найшвидшого спуску: $\lambda_j = \arg\min_\lambda F(\mathbf{x}_j - \lambda e_k)$.

Недоліки методу покоординатного спуску. Неважко зрозуміти, що існують функції, на яких метод покоординатного спуску може не привести навіть у локальний оптимум. Нехай лінії рівня утворюють "гострий" яр (рис. 1), якщо на черговій ітерації поточна точка виявиться на "дні" яру, то будь-який наступний рух уздовж осей координат приведе на підйом. Ніякий подальший спуск по координатах неможливий, хоча мінімуму не досягнуто.

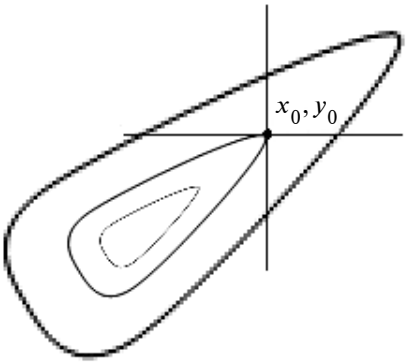

Рис. 1. "Гострий" яр, утворений лініями рівня функції

У випадку гладкої цільової функції "гострого" яру бути не може, але функція може бути як завгодно близькою до нього і рух уздовж осей координат буде можливий лише з дуже маленьким кроком. Як наслідок, таких "шажків" потрібно буде виконати дуже багато.

Евристика середньозваженого напрямку

По суті метод покоординатного спуску виконує оптимізацію в $2N$ напрямках:

$$e_1, e_2, .., e_N, -e_1, -e_2, .., -e_N, \qquad (3)$$

де N – розмірність простору. Позначимо ці вектори $\mathbf{a}_1, ..., \mathbf{a}_{2N}$.

Нехай уже була зроблена певна кількість ітерацій і спуск був зроблений $c_i > 0$ раз у напрямку $\mathbf{a}_i$, $c_j > 0$ раз у напрямку $\mathbf{a}_j$ і не зроблений жодного разу в інших напрямках. Логічно припустити, що в такому випадку напрямок "дійсного спуску" десь між векторами $\mathbf{a}_i$ й $\mathbf{a}_j$, причому "ближчий" до того з них, у якого c більше. Використовуючи ці навідні міркування, запропонуємо евристику.

На кожній ітерації додамо до базового набору з $2N$ векторів (3), їх середньозважене значення, де вагою вектора будемо вважати кількість спусків у його напрямку, тобто вектор $\tilde{v} = \sum a_i c_i, v = \tilde{v} / \|\tilde{v}\|$.

Неважко зрозуміти, що запропонована евристика сильно прискорює процес оптимізації у випадку, показаному на рис. 1.

Реалізація евристики середньозваженого напрямку

У даній статті запропонована реалізація алгоритму, в якій вибір кроку відбувається "діленням і множенням" (докладніше в [5]). Увесь час у наборі напрямних векторів перебувають $2N$ базових векторів (3) і ще один вектор, який виходить на кожній ітерації заново по описаному вище принципу.

Конкретна реалізація алгоритму (мова програмування — C++):

```
void Coordplusaverage(vect &x){
 vect a[2*DIM + 1];
//all vectorz are zeroed in constructor
 cfloat c[2*DIM+1];

//init base vectors [1..2*DIM]
for(int i = 1; i <= DIM; i++){
 c[i] = c[i+DIM] = 1;
// one is kind of a start bonus
 a[i][i-1] = 1;
 a[DIM+i][i-1] = -1;
 }

//the average(0th) vector remains zero

longlong itc0 = itc;
 cfloat cur_val = F(x), next_val;
//we save current and next values
//to use less function calls
for(cfloat d = INF ; d > EPS ; ){
//start with a big step
//finish when it's realy small
bool changed = false;
for(int i = 0; i < 2*DIM+1; i++){
//for each base vector
 x = x + a[i] * d;
```

```
//try to move
 next_val = F(x);
if( next_val < cur_val ){
//if it's better
 c[i]++;
 changed = true;
 cur_val = next_val;
 }else
 x = x - a[i] * d;
//move back
 }
 vect v;// = zero
//create new vector as a weighted average
for(int i = 0; i < 2*DIM+1; i++)
 v = v + (c[i]*a[i]);

if( v.Norm() > 1E-6 ){
//if a vector is non-zero
 v.Normalize();
 a[0] = v;
 c[0] = 1;
//place it
 }

if( !changed ){
//if we're not able to move with current step
 d /= 2;
//make it smaller
}else
 d *= 2;
//make it bigger
 }
 cout<<"function F was executed "<<itc-itc0<<"
times"<<endl;
}
```

У роботі були експериментально досліджені "особливості поведінки" реалізованого алгоритму на прикладі функцій, описаних у [5]. Це "гарна" функція, функція Розенброка й "погана" функція. Усі тести повністю ідентичні відповідним тестам з [5, 6]. Для порівняння зазначено результати деяких реалізацій (усіх, крім найшвидшого спуску), досліджених у [5, 6].

"Гарна" функція

$F(x, y) = x^2 + y^2 + 3x - 4y + 2$ — гладка, опукла функція. Глобальний мінімум існує, єдиний і досягається в точці $x^* = -1,5, y^* = 2$.

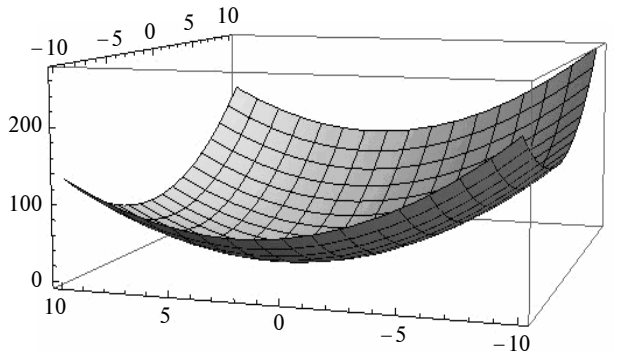

Рис. 2. "Гарна" функція

Тест. Початкове наближення $x_0 = 123$, $y_0 = -321$, нескінченність: INF = 1E9, точність: EPS = 1E−9 (табл. 1).

У першому стовпці таблиці назва алгоритму, у другому/третьому — абсолютна/відносна погрішність знаходження x^*, у четвертому/ п'ятому стовпцях — абсолютна/відносна погрішність знаходження $F(x^*)$, в останньому стовпці — кількість викликів цільової функції.

***Таблиця* 1.** "Гарна" функція. Тест

Algorithm	$\|x - x^*\|$	$\|x - x^*\|/\|x^*\|$	$\|F(x) - F(x^*)\|$	$\|F(x) - F(x^*)\|/\|F(x^*)\|$	$F()$calls #
Coordinatediv	3,8E−10	1,5E−10	0,0E+0	0,0E+0	334
Coordinatedivmul	3,8E−10	1,5E−10	0,0E+0	0,0E+0	450
Coordinatesteepest	3,8E−10	1,5E−10	0,0E+0	0,0E+0	702
Gradientdiv	1,8E−8	7,4E−9	3,5E−16	8,2E−17	985
Gradientdivnorm	1,6E−9	6,5E−10	2,6E−18	6,1E−19	391
Gradientdivmulnorm	1,6E−9	6,5E−10	2,6E−18	6,1E−19	481
Gradientsteepest	1,0E−10	4,0E−11	0,0E+0	0,0E+0	356
Rosenbrokdiv	1,6E−9	6,5E−10	2,6E−18	6,1E−19	381
Rosenbrokdivmul	1,6E−9	6,5E−10	2,6E−18	6,1E−19	454
Coordplusaverage	9,6E−10	3,8E−10	8,6E−19	2,0E−19	571

Як видно з табл. 1, метод покоординатного спуску з евристикою середньозваженого напрямку впорався так само добре, як і методи, досліджені в [5, 6].

Функція Розенброка

$F(x, y) = (1 - x)^2 + 100(y - x^2)^2$ — гладка, неопукла функція. Глобальний мінімум існує, єдиний і досягається в точці $x^* = 1, y^* = 1$.

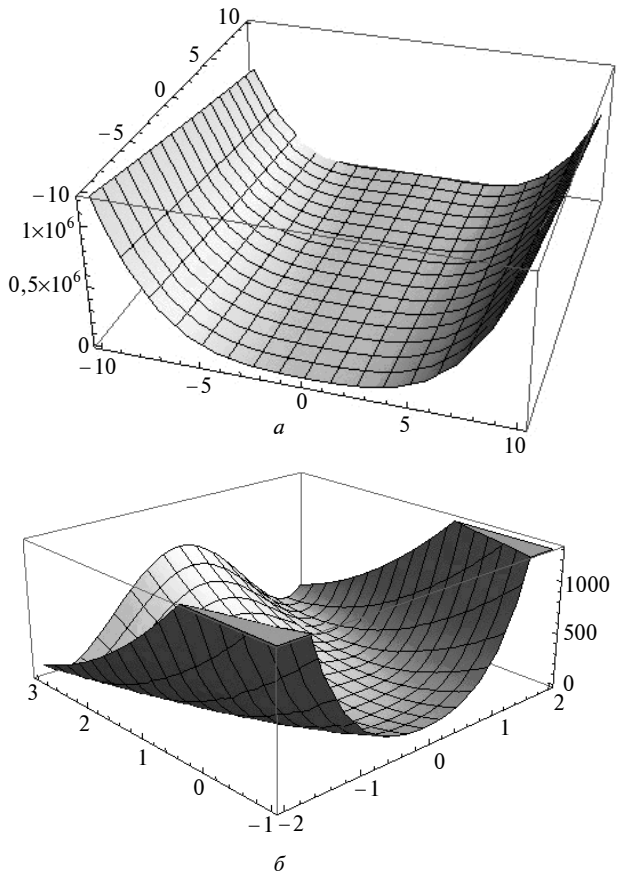

Рис. 3. Функція Розенброка: *а* – глобальний вигляд; *б* – локальний вигляд

***Тест* 1.** Початкове наближення $x_0 = 123$, $y_0 = -321$, нескінченність – INF = 1E9, точність – EPS = 1E–9 (табл. 2).

Як видно, запропонований метод обігнав усі методи, досліджені в [5, 6], з досить більшою перевагою.

***Тест* 2.** Початкове наближення $x_0 = 123$, $y_0 = -321$, нескінченність – INF = 1E9, точність – EPS = 1E–12 (табл. 3).

Як варто було сподіватися, кількість ітерацій досліджуваного методу збільшилася незначно, як і в координатних методах без евристики. У цьому тесті метод покоординатного спуску з евристикою середньозваженого напрямку також обігнав усі інші методи.

***Тест* 3.** Початкове наближення $x_0 = 12$, $y_0 = -32$, нескінченність – INF = 1E9, точність – EPS = 1E–12 (табл. 4).

Взявши початкову точку ближче до точки мінімуму, метод покоординатного спуску з евристикою середньозваженого напрямку обігнав усі інші методи із ще більш істотною перевагою (на порядок менше ітерацій, ніж у найближчого конкурента – методу Розенброка). Спробуємо взяти точку ще ближче.

***Тест* 4.** Початкове наближення $x_0 = 3$, $y_0 = -3$, нескінченність – INF = 1E9, точність – EPS = 1E–12 (табл. 5).

Як видно, час роботи досліджуваного методу знизився не так помітно, як при попередньому наближенні точки, і не так помітно, як для координатних методів або методу Розенброка. Це пояснюється тим, що й на попередньому тесті кількості ітерацій було досить мало (хоч метод Розенброка й розв'язав цей тест у 5 разів швидше, ніж попередній, це однаково у два рази повільніше за час, показаний методом покоординатного спуску з евристикою середньозваженого напрямку).

***Таблиця* 2.** Функція Розенброка. Тест 1

Algorithm	$\lvert x - x^* \rvert$	$\lvert x - x^* \rvert / \lvert x^* \rvert$	$\lvert F(x) - F(x^*) \rvert$	$\lvert F(x) - F(x^*) \rvert / \lvert F(x^*) \rvert$	$F()$calls #
Coordinatediv	9,7E–7	6,9E–7	1,9E–13	–	1,592,528,468
Coordinatedivmul	9,7E–7	6,8E–7	1,8E–13	–	163,948,785
Coordinatesteepest	7,6E–7	5,4E–7	1,1E–13	–	1,318,302
Gradientdiv	1,1E–2	7,9E–3	2,5E–5	–	232,200,030
Gradientdivnorm	2,4E–7	1,7E–7	1,1E–14	–	84,481
Gradientdivmulnorm	5,2E–8	3,6E–8	5,4E–16	–	182,591
Rosenbrokdiv	1,2E–7	8,9E–8	3,4E–15	–	49,728,621
Rosenbrokdivmul	2,9E–7	2,0E–7	1,7E–14	–	95,423
Coordplusaverage	2,1E–7	1,5E–7	9,2E–15	–	22,651

***Таблиця* 3.** Функція Розенброка. Тест 2

Algorithm	$\|x - x^*\|$	$\|x - x^*\|/\|x^*\|$	$\|F(x) - F(x^*)\|$	$\|F(x) - F(x^*)\|/\|F(x^*)\|$	$F()$calls #
Coordinatediv	4,5E–10	3,2E–10	4,1E–20	–	1,592,539,501
Coordinatedivmul	4,5E–10	3,2E–10	4,1E–20	–	163,968,322
Coordinatesteepest	1,1E–9	8,0E–10	2,5E–19	–	2,323,658
Gradientdiv	1,1E–5	7,9E–6	2,5E–11	–	621,017,650
Gradientdivnorm	1,5E–9	1,0E–9	4,6E–19	–	118,041
Gradientdivmulnorm	1,4E–9	1,0E–9	4,1E–19	–	228,111
Rosenbrokdiv	7,1E–11	5,0E–11	1,0E–21	–	49,731,039
Rosenbrokdivmul	1,0E–10	7,5E–11	2,2E–21	–	99,695
Coordplusaverage	3,8E–10	2,6E–10	2,9E–20	–	23,031

***Таблиця* 4.** Функція Розенброка. Тест 3

Algorithm	$\|x - x^*\|$	$\|x - x^*\|/\|x^*\|$	$\|F(x) - F(x^*)\|$	$\|F(x) - F(x^*)\|/\|F(x^*)\|$	$F()$calls #
Coordinatediv	8,9E–10	6,3E–10	1,6E–19	–	1,867,763
Coordinatedivmul	8,9E–10	6,3E–10	1,6E–19	–	884,977
Coordinatesteepest	1,0E–9	7,4E–10	2,2E–19	–	2,324,240
Gradientdiv	8,8E–8	6,2E–8	1,5E–15	–	6,987,095
Gradientdivnorm	1,5E–9	1,0E–9	4,6E–19	–	120,186
Gradientdivmulnorm	1,3E–9	9,8E–10	3,8E–19	–	230,841
Rosenbrokdiv	8,4E–10	5,9E–10	1,4E–19	–	44,898
Rosenbrokdivmul	5,1E–10	3,6E–10	5,3E–20	–	27,077
Coordplusaverage	6,8E–11	4,8E–11	9,7E–22	–	3,651

***Таблиця* 5.** Функція Розенброка. Тест 4

Algorithm	$\|x - x^*\|$	$\|x - x^*\|/\|x^*\|$	$\|F(x) - F(x^*)\|$	$\|F(x) - F(x^*)\|/\|F(x^*)\|$	$F()$calls #
Coordinatediv	1,0E–9	7,2E–10	2,1E–19	–	59,634
Coordinatedivmul	1,0E–9	7,2E–10	2,0E–19	–	94,006
Coordinatesteepest	1,1E–9	7,9E–10	2,5E–19	–	2,323,968
Gradientdiv	1,1E–8	8,1E–9	2,6E–17	–	987,780
Gradientdivnorm	1,5E–9	1,0E–9	4,6E–19	–	117,206
Gradientdivmulnorm	1,3E–9	9,8E–10	3,8E–19	–	225,601
Rosenbrokdiv	6,5E –10	4,6E –10	8,4E–20	–	38,792
Rosenbrokdivmul	4,6E –10	3,2E –10	4,3E –20	–	5,434
Coordplusaverage	2,3E –11	1,6E –11	1,4E –22	–	2,751

У всіх тестах для функції Розенброка досліджуваний метод упорався краще, маючи перевагу в кілька разів порівняно з іншими методами.

“Погана” функція

$F(x, y) = x^2 + 1000\dfrac{y^2}{x^2 + 0,01}$ – гладка, неопукла функція Глобальний мінімум існує, єдиний і досягається в точці $x^* = 0, y^* = 0$.

Тест. Початкове наближення $x_0 = 123$, $y_0 = -321$, нескінченність – INF = 1E9, точність – EPS = 1E–9 (табл. 6).

Як видно з табл. 6, координатні методи впоралися із задачею краще інших, градієнтні – гірше. Метод покоординатного спуску з евристикою середньозваженого напрямку працював майже так само швидко, як і чисто координатні методи (у півтора разу повільніше найшвидшого з них), обігнавши градієнтні на кілька порядків і метод Розенброка на порядок. Це

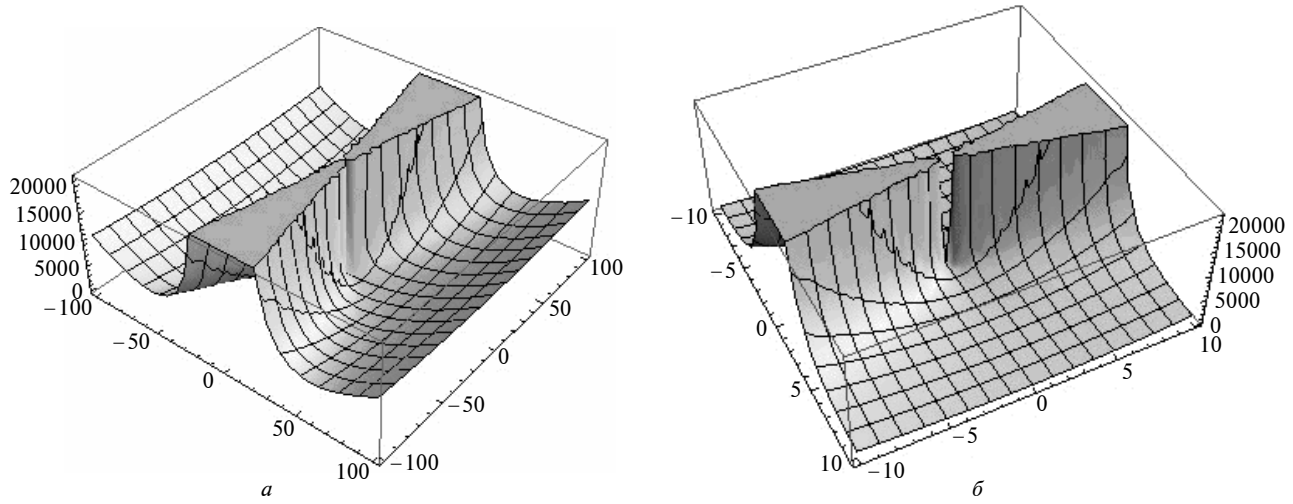

Рис. 4. "Погана" функція: *а* – глобальний вигляд; *б* – локальний вигляд

***Таблиця* 6.** "Погана" функція. Тест

Algorithm	$\|x - x^*\|$	$\|x - x^*\|/\|x^*\|$	$\|F(x) - F(x^*)\|$	$\|F(x) - F(x^*)\|/\|F(x^*)\|$	$F()$calls #
Coordinatediv	8,8E–10	–	4,7E–14	–	891
Coordinatedivmul	8,8E–10	–	4,7E–14	–	700
Coordinatesteepest	4,0E–10	–	6,6E–15	–	940
Gradientdiv	7,0E–5	–	4,9E–9	–	2,589,625
Gradientdivnorm	3,4E–6	–	1,2E–11	–	4,492,921
Gradientdivmulnorm	1,8E–8	–	3,5E–16	–	9,867,881
Rosenbrokdiv	5,9E–5	–	3,5E–9	–	735,927
Rosenbrokdivmul	1,2E–6	–	1,6E–12	–	9,560
Coordplusaverage	8,6E–10	–	2,1E–14	–	1,051

ще раз доводить, що запропонована евристика "не псує" метод покоординатного спуску.

Висновки

Запропоновано евристику для методу покоординатного спуску. Розглянуто ефективну реалізацію методу із запропонованою евристикою. Збіжність отриманого алгоритму була протестована на трьох функціях: "гарній" (опуклій), функції Розенброка з різними початковими наближеннями і "поганій" функції. Результати тестування були порівняні з результатами, отриманими в [5, 6]. Отже, можна зробити такі висновки:

- для "гарних" функцій усі методи працюють досить добре;
- на всіх запропонованих функціях метод покоординатного спуску з евристикою середньозваженого напрямку продемонстрував бажану швидкодію;
- запропонований алгоритм програвав максимум у півтора разу градієнтним і координатним методам на "гарних" для цих методів тестах і вигравав на кілька порядків на "поганих";
- евристика середньозваженого напрямку істотно поліпшує роботу методу покоординатного спуску на "поганих" для нього функціях;
- запропонована евристика істотно ускладнює реалізацію методу й збільшує довжину коду. Це важливо, оскільки простота реалізації вважається однією з головних переваг методу покоординатного спуску;
- метод покоординатного спуску з евристикою середньозваженого напрямку має перевагу над методом Розенброка (який вважається досить універсальним методом [4]) за швидкодією на всіх "складних" тестах. На "гарній" функції він програв на 25 %. Це говорить про те, що запропонований алгоритм теж є ефективним в обчислювальному відношенні і досить універсальним.

У подальших дослідженнях можна буде перевірити ефективність запропонованого методу для мінімізації функцій у просторах з більшою вимірністю.

1. *Химмельблау Д.* Прикладное нелинейное программирование. – М.: Мир, 1974. – 536 с.
2. *Полак Э.* Численные методы оптимизации. Единый подход. – М.: Мир, 1974. – 376 с.
3. *Метод* градієнтного спуску. – http://en.wikipedia.org/wiki/Gradient_descent
4. *Rosenbrock H.H.* An automatic Method for finding the greatest or least Value of a Function // Computer Journal. – 1960. – N 3. – P. 175–184.
5. *Касицкий А.В.* Сравнение градиентных методов // Системные науки и кибернетика. – 2011. – № 1. – С. 42–55.
6. *Касицкий А.В.* Анализ сходимости метода Розенброка // Системные технологии. – 2011. – № 6. – С. 35–43.
7. *Метод* покоординатного спуска. – http://www.machinelearning.ru/wiki/index.php?title=Метод покоординатного спуска
8. *Функция* Розенброка. – http://en.wikipedia.org/wiki/Rosenbrock_function

Рекомендована Радою
Навчально-наукового комплексу
“Інститут прикладного системного
аналізу” НТУУ “КПІ”

Надійшла до редакції
28 лютого 2012 року