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

Робочі матеріали
Випуск 19-03

П.І. Стецюк, В.О. Стовба

СУБГРАДІЄНТНІ МЕТОДИ З КРОКОМ ПОЛЯКА ТА ПРОГРАМА AMSG2P

Київ – 2019

УДК 519.8

Автори
П.І. Стецюк, В.О. Стовба

Субградієнтні методи з кроком Поляка та програма amsg2p[1]. – Київ: 2019. – 28 с. Робочі матеріали. Випуск 19-03 / Інститут кібернетики імені В.М. Глушкова НАН України, відділ методів негладкої оптимізації.

Розглядаються три субградієнтні методи (методи **A** та **B**, метод **amsg2p**) для знаходження точки мінімуму опуклої функції з відомим оптимальним значенням функції та їхні програмні реалізації мовою Octave: метод **A** – субградієнтний метод, що використовує крок Поляка у початковому просторі змінних (програма **PolyakA**); метод **B** – субградієнтний метод, що використовує крок Поляка у перетвореному просторі змінних (програма **PolyakB**); **amsg2p** – субградієнтний алгоритм з кроком Поляка та перетворенням простору за допомогою двох послідовних субградієнтів та агрегатного вектора (програма **amsg2p**).

Іл. 4. Табл. 5. Бібліогр.: с. 26–27 (13 найменувань).

UDC 519.8

Authors
P.I. Stetsyuk, V.O. Stovba

Subgradient methods with Polyak's step and program amsg2p. – Kyiv: 2019. – 28 p. Working papers. Issue 19-03 / V.M. Glushkov Institute of Cybernetics of NAS of Ukraine, Department of non-smooth optimization methods.

Three subgradient methods (methods **A** and **B**, method **amsg2p**) to find the minimum point of a convex function with known optimal values of a function and their program implementations in Octave are considered: method A is a subgradient method using Polyak's step in the original space of variables (program **PolyakA**); method **B** is a subgradient method using Polyak's step in the transformed space of variables (program **PolyakB**); **amsg2p** is a subgradient algorithm with Polyak's step and transformation of space using two successive subgradients and the aggregate vector (program **amsg2p**).

Fig. 4. Tab. 5. Ref.: p. 26–27 (13 references).

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

ВСТУП

Розглядаються три субградієнтних методи (методи **A** та **B** [1, 2], метод **amsg2p**) для знаходження точки мінімуму опуклої функції з відомим оптимальним значенням функції.

Метод **A** – це субградієнтний метод, що використовує крок Поляка у початковому просторі змінних. Метод **B** – це субградієнтний метод у перетвореному просторі змінних, що використовує крок Поляка у перетвореному просторі. Для обох методів розглядаються доведення збіжності до точки мінімуму з заданою точністю, що визначається оптимальним значенням функції. Наведені приклади гладких та негладких яружних опуклих функцій, для яких метод **A** збігається повільно. Показано, що правильно обравши матрицю перетворення, метод **B** можна суттєво прискорити у порівнянні з методом **A** для яружних опуклих функцій.

Крок Поляка використовується також у методі **amsg2p**, де перетворення простору направлене на зменшення яружності функції в черговому просторі змінних. Перетворення реалізується за допомогою однорангового еліпсоїдального оператора на тих ітераціях методу, коли тупим є хоча б один з кутів – або кут між двома послідовними субградієнтами, або кут між останнім субградієнтом та агрегатним вектором, який є опуклою комбінацією обчислених на попередніх ітераціях субградієнтів.

1. ПОЗНАЧЕННЯ ТА ВИЗНАЧЕННЯ

Нехай $f(x)$ – опукла функція, $x \in R^n$. Позначимо її мінімальне значення як $f^* = f(x^*)$ та припустимо, що точка мінімуму x^* − єдина. Субградієнт $g_f(x)$ задовольняє таку умову:

$$(x - x^*, g_f(x)) \geq f(x) - f^*, \ \forall x \in R^n. \tag{1}$$

Тут (x, y) – скалярний добуток векторів $x \in R^n$ та $y \in R^n$.

Якщо $f(x)$ є неперервно-диференційовною в точці $\bar{x}$, то субградієнт $g_f(\bar{x})$ однозначно визначається і збігається з $\nabla f(\bar{x})$ – градієнтом функції $f(x)$ у точці $\bar{x}$. У точках, де функція $f(x)$ негладка, субградієнт $g_f(\bar{x})$ визначається неоднозначно.

Нерівність (1) випливає з визначення субградієнта $g_f(x)$ для опуклої функції $f(x)$. Справді, субградієнт $g_f(x)$ в точці x задовольняє нерівність

$$f(y) - f(x) \geq (g_f(x), y - x), \ \forall y \in R^n, \tag{2}$$

яка виконується також для x^* − точки мінімуму. Для точки x^* нерівність (2) перетворюється на нерівність

$$f(x^*) - f(x) \geq (g_f(x), x^* - x),$$

яку, зважаючи на те, що $f(x^*) = f^*$, можна записати як нерівність (1).

Якщо f^* відоме, то для знаходження наближення до точки x^* можна використати субградієнтний метод Поляка [1]:

$$x_{k+1} = x_k - h_k \frac{g_f(x_k)}{\|g_f(x_k)\|}, \ h_k = \frac{f(x_k) - f^*}{\|g_f(x_k)\|}, \ k = 0,1,2... \tag{3}$$

Крок h_k називається кроком Поляка (або кроком Агмона-Моцкіна-Шонберга). Вперше крок Поляка використовувався для мінімізації кусково-лінійних опуклих функцій. Так, у 1954 році Аґмон [2] та Моцкін і Шонберг [3] використали цей крок у релаксаційному методі для знаходження хоча б одного розв’язку сумісної системи лінійних нерівностей. У 1965 р. І.І. Єрьомін [4] узагальнив цей релаксаційний метод для системи опуклих функцій.

Геометричний зміст методу (3) є таким. Функція $f(x)$ апроксимується лінійною функцією $\tilde{f}(x) = f(x_k) + (g_f(x_k), x - x_k)$, і крок вибирається таким чином, щоб апроксимуюча функція стала рівною f^* (тобто $\tilde{f}(x_{k+1}) = f^*$). Для опуклої функції $f(x)$ крок h_k визначає таку величину максимального зміщення з точки x_k у напрямку нормалізованого антисубградієнта, для якого умова (1) гарантує, що кут між антисубградієнтом у точці x_k та напрямком від точки x_{k+1} до точки мінімуму x^* не буде тупим.

Це означає такий факт. Субградієнт у точці x_k визначає гіперплощину, яка локалізує точку x^* у напівпросторі у напрямку антисубградієнта. Якщо її перемістити на величину максимального зміщення у напрямку нормованого антисубградієнта, то нова гіперплощина буде локалізувати x^* у такому напівпросторі по відношенню до точки x_{k+1}.

Далі розглянемо субградієнтний метод Поляка (метод **A**, розділ 2) та його прискорені варіанти (метод **B**, розділ 3, метод **amsg2p,** розділ 4) для знаходження наближення до точки мінімуму яружних опуклих функцій. Як критерій зупинки використаємо умову $f(x_k) - f^* \leq £$; для довільного малого $£ > 0$ це дозволяє нам знайти таку точку $x_£^* = x_k$, що $f(x_£^*) \leq f^* + £$.

Будемо розглядати методи **A** та **B** для більш загального випадку опуклої функції $f(x)$, коли її субградієнт $g_f(x)$ задовольняє таку умову:

$$(x - x^*, g_f(x)) \geq m(f(x) - f^*),\ \forall x \in R^n, \qquad (4)$$

де параметр $m \geq 1$. Параметр m вводиться для того, щоб урахувати спеціальні класи опуклих функцій; наприклад, для кусково-лінійних негладких функцій $m = 1$, для квадратичних гладких $m = 2$, для функцій виду $f(x) = \sum_{i=1}^{k} \left| \sum_{j=1}^{n} a_{ij} x_j - b_i \right|^p$, де $p > 1$, $m = p$.

Параметр m можна активно використовувати для диференційованих однорідних з показником ± опуклих функцій. Для них виконується рівність $±\left(f(x) - f^*\right) = \left(x - x^*, g_f(x)\right)$, отже параметр m можна вибрати $m = ± > 1$.

2. СУБГРАДІЄНТНИЙ МЕТОД ПОЛЯКА

Опис методу А. Якщо опукла функція $f(x)$ задовольняє умову (4) та f^* відоме, тоді для знаходження точки $x^*_{\pounds} \in R^n$ такої, що $f(x^*_{\pounds}) \le f^* + \pounds$, можна використати такий ітеративний метод.

Ініціалізація. Нехай f^* та $m \ge 1$ відомі. Вибираємо початкову точку $x_0 \in R^n$, величину $\pounds > 0$ та переходимо до наступної ітерації з величиною x_0.

Ітеративний процес. Нехай точка $x_k \in R^n$ була знайдена на *k*-й ітерації. Щоб перейти до наступної (*k*+1)-ї ітерації, виконаємо такі дії.

A1. Обчислимо $f(x_k)$ та $g_f(x_k)$. Якщо $f(x_k) - f^* \le \pounds$, тоді STOP ($k^* = k,\ x^*_{\pounds} = x_k$).

A2. Обчислимо наступну точку

$$x_{k+1} = x_k - h_k \frac{g_f(x_k)}{\left\| g_f(x_k) \right\|}, \quad h_k = \frac{m(f(x_k) - f^*)}{\left\| g_f(x_k) \right\|}.$$

A3. Переходимо до наступної (*k*+1)-ї ітерації з x_{k+1}.

Теорема 1. Послідовність $\{x_k\}_{k=0}^{k^*-1}$, породжена методом **A**, задовольняє таку нерівність:

$$\left\| x_{k+1} - x^* \right\|^2 \le \left\| x_k - x^* \right\|^2 - \frac{m^2(f(x_k) - f^*)^2}{\left\| g_f(x_k) \right\|^2},\ k = 0,1,2,... \qquad (5)$$

Доведення. З **A2** для довільного k $(0 \le k \le k^* - 1)$ маємо

$$\left\| x_{k+1} - x^* \right\|^2 = \left\| x_k - x^* - h_k \frac{g_f(x_k)}{\left\| g_f(x_k) \right\|} \right\|^2 = \left\| x_k - x^* \right\|^2 - 2h_k \frac{(x_k - x^*, g_f(x_k))}{\left\| g_f(x_k) \right\|} + h_k^2.$$

Зважаючи на те, що з (4) випливає

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

маємо

$$\left\|x_{k+1}-x^*\right\|^2 \le \left\|x_k - x^*\right\|^2 - h_k^2 = \left\|x_k - x^*\right\|^2 - \left(\frac{m(f(x_k)-f)^*}{\left\|g_f(x_k)\right\|}\right)^2,$$

звідки випливають нерівності (5). Теорему доведено.

Теорема 1 гарантує, що в субградієнтному методі Поляка відстань до точки мінімуму монотонно зменшується. Крім того, задовольняються такі нерівності:

$$(x^* - x_{k+1}, -g_f(x_k)) \ge 0,\ k = 0,1,... \qquad (6)$$

які означають, що кут між антисубградієнтом в точці x_k і напрямком від точки x_{k-1} до точки мінімуму не тупий. Дійсно, нерівності (6) випливають з того факту, що, використовуючи (4), маємо

$$(x^* - x_{k+1}, -g_f(x_k)) = (x_{k+1} - x^*, g_f(x_k)) = (x_k - x^* - h_k \frac{g_f(x_k)}{\left\|g_f(x_k)\right\|}, g_f(x_k)) =$$

$$= (x_k - x^*, g_f(x_k) - h_k \left\|g_f(x_k)\right\|) = (x_k - x^*, g_f(x_k)) - m(f(x_k) - f^*) \ge 0.$$

Нерівності (6) означають, що для опуклої функції $f(x)$, яка задовольняє умову (4), h_k визначає величину максимального переміщення в напрямку нормалізованого антисубградієнта, при якому гарантується, що кут між антисубградієнтом і напрямком від точки x_{k-1} до точки мінімуму не буде тупим.

Octave-функція PolyakA. Octave-програма PolyakA знаходить наближення $x_{£}^{*}$ точки мінімуму опуклої функції $f(x)$ від n змінних, що для методу **A** визначається такими вхідними даними: початкова точка x_0; f^* – значення функції в точці мінімуму; $m \ge 1$ – параметр, що визначає величину переміщення в напрямку антисубградієнта, тобто виконання умови (4), параметри зупинки $£_f$ та maxitn.

Програма використовує octave-функцію **function [f, g] = calcfg (x)**, яка обчислює значення функції $f = f(x)$ та її субградієнт у точці x, $g_f(x) \in \partial f(x)$. Назва функції **calcfg** (x) може бути довільною, яку дозволяє синтаксис мови octave. Програма **PolyakA** використовує такі вхідні та вихідні параметри:

```
# Input parameters:
# calcfg - reference to function for calculation of f(x) and g(x)
# x0 - the starting point, x0(1:n)
# fstar - value of the function at the minimum point
# m - length of shift along anti-subgradient (m>=1)
# epsf, maxitn - stop parameters
# intp - print information every intp iteration
# Output parameters:
# x - the minimum point, which was found by the program, x(1:n)
# f - the value of the function f at the point x
# itn - the number of iterations used by the program
# info - exit code (0 = epsf, 4 = maxitn).
```

Статус точки x визначається значенням коду повернення **info** на ітерації itn = k**: info = 0** – зупинка, якщо знайдена точка x_k, для якої $f(x_k) - f^* \le £_f$; **info = 4** – зупинка, якщо itn > maxitn (ітераційний процес перевищив максимальну кількість ітерацій).

Програма **PolyakA** є octave-функцією, її код подано нижче.

```
function [x,f,itn,info] = PolyakA(calcfg,x0,fstar,m,              #row01
                                  epsf,maxitn,intp);              #.....
itn=0; x=x0; [f,g] = calcfg(x); dg=norm(g);                       #row02
if(intp>0)                                                        #row03
   printf("itn %4d f %14.6e \n", itn, f); # xprint = x’,          #.....
endif                                                             #.....
for(itn = 1:maxitn)                                               #row04
   if(f-fstar < epsf) info = 0; return; endif                     #row05
   g1=g/dg; hs=m*(f-fstar)/dg;                                    #row06
   x -= hs * g1;                                                  #row07
   [f,g] = calcfg(x); dg=norm(g);                                 #row08
   if(mod(itn,intp)==0)                                           #row09
      printf("itn %4d f %14.6e \n",itn,f); # xprint = x’,         #.....
   endif                                                          #.....
endfor                                                            #row10
info = 4;                                                         #row11
endfunction                                                       #row12
```

Код програми складається з **14** рядків і більшість із них містить декілька octave-операторів. Ітеративний процес виконується в циклі **for** (рядки 4−10), де для k-ої ітерації субградієнт $g_f(x_k)$ зберігається як вектор-стовпчик ***g***, а нормований субградієнт $g_f(x_k)$ – як вектор-стовпчик ***g*1**. У циклі **for** зупинка реалізується тоді, коли відхилення функції від оптимального значення стає меншим, ніж $£_f$ (рядок 5).

Використовуючи програму **PolyakA**, можна знайти прийнятно-точні наближення до точки мінімуму гладкої опуклої функції, поверхні рівня якої характеризуються малим ступенем яружності (витягнутості поверхонь рівня функції). Метод **A** збігається швидко до точки мінімуму опуклої функції, якщо кути між послідовними субградієнтами на кожній ітерації будуть гострими або тупими, близькими до прямого кута.

Недоліком методу **A** є його повільна збіжність для яружних функцій. Нижче буде детальніше розглянута збіжність методу **A** для функцій $f_1(x_1,x_2)=|x_1|+t|x_2|,\ t>1$ та $f_2(x_1,x_2)=x_1^2+tx_2^2,\ t>>1$.

Обчислювальні експерименти для яружних функцій. Нижче описуються результати обчислювальних експериментів з використанням програми **PolyakA**. Експерименти пов'язані з вивченням швидкості збіжності методу **A** для яружних функцій двох змінних – квадратичної функції $f_2(x_1,x_2)=x_1^2+tx_2^2,\ t>>1,\ f^*=0,\ x^*=(0,0)^T$ (задача **quad**) та кусково-лінійної функції $f_1(x_1,x_2)=|x_1|+t|x_2|,\ t>1,\ f^*=0,\ x^*=(0,0)^T$ (задача **sabs**).

Експерименти були проведені на комп'ютері Pentium 2.5 GHz з операційною системою Windows XP/32 з використанням GNU Octave версії 3.0.0. Для задач **quad** і **sabs** використовуються octave-функції для обчислення $f(x)$ та $g_f(x)$ у такому вигляді:

```
function [f,g] = squad(x)          |  function [f,g] = sabs(x)
global t;                          |  global t;
f=x(1,1)*x(1,1)+t*x(2,1)*x(2,1);   |  f=abs(x(1,1))+t*abs(x(2,1));
g(1,1) = 2*x(1,1);                 |  g(1,1) = sign(x(1,1));
g(2,1) = 2*t*x(2,1);               |  g(2,1) = t*sign(x(2,1));
endfunction                        |  endfunction
```

У випадку негладкої функції двох змінних $f_1(x_1,x_2)=|x_1|+t|x_2|,\ t>1$ швидкість збіжності методу **A** визначається геометричною прогресією зі знаменником

$$q = \sqrt{1 - 1/t^2} \tag{7}$$

і буде дуже повільною для великих значень t. Наприклад, зигзагоподібна траєкторія методу **A** для знаходження точки мінімуму слабко яружної функції $f_1(x_1, x_2) = |x_1| + 5|x_2|$ показана на рис. 1.

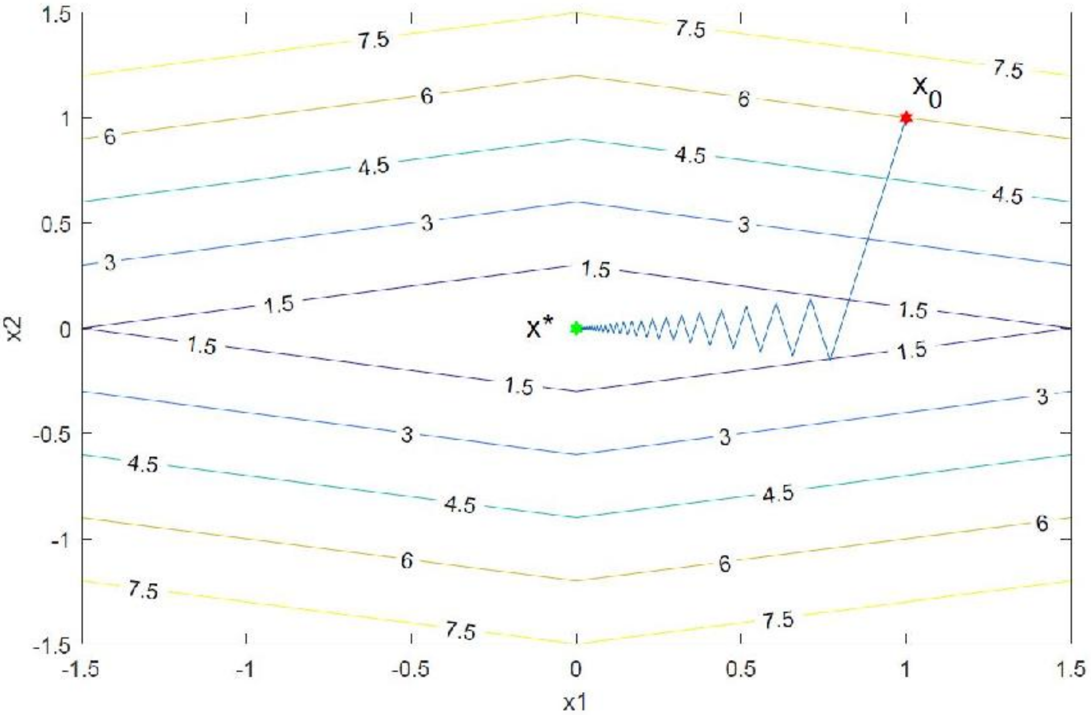

Рисунок 1 – Траєкторія методу **A** для пошуку наближення до точки мінімуму функції $f_1(x_1, x_2) = |x_1| + 5|x_2| : m = 1,\ f^* = 0,\ x^* = (1,1)^T$

На рис. 2, 3 зображені траєкторії методу **A** для мінімізації квадратичної яружної функції $f_2(x_1, x_2) = x_1^2 + t x_2^2$. Якщо $t = 6, m = 2$ та £ $= 0.01$, то траєкторія є зигзагоподібною (див. рис. 2), при чому для перших п'яти ітерацій метод породжує послідовність $x_0 = (1.00, 1.00)^T$, $x_1 = (0.811, -0.135)^T$, $x_2 = (0.338, 0.338)^T$, $x_3 = (0.274, -0.046)^T$, $x_4 = (0.114, 0.114)^T$, $x_5 = (0.093, -0.015)^T$ (див. рис. 2). Якщо ж $t = 6$ та $m = 1$, то траєкторія, породжена методом **A**, не є зигзагоподібною та прямує до точки мінімуму швидше (див. рис. 3).

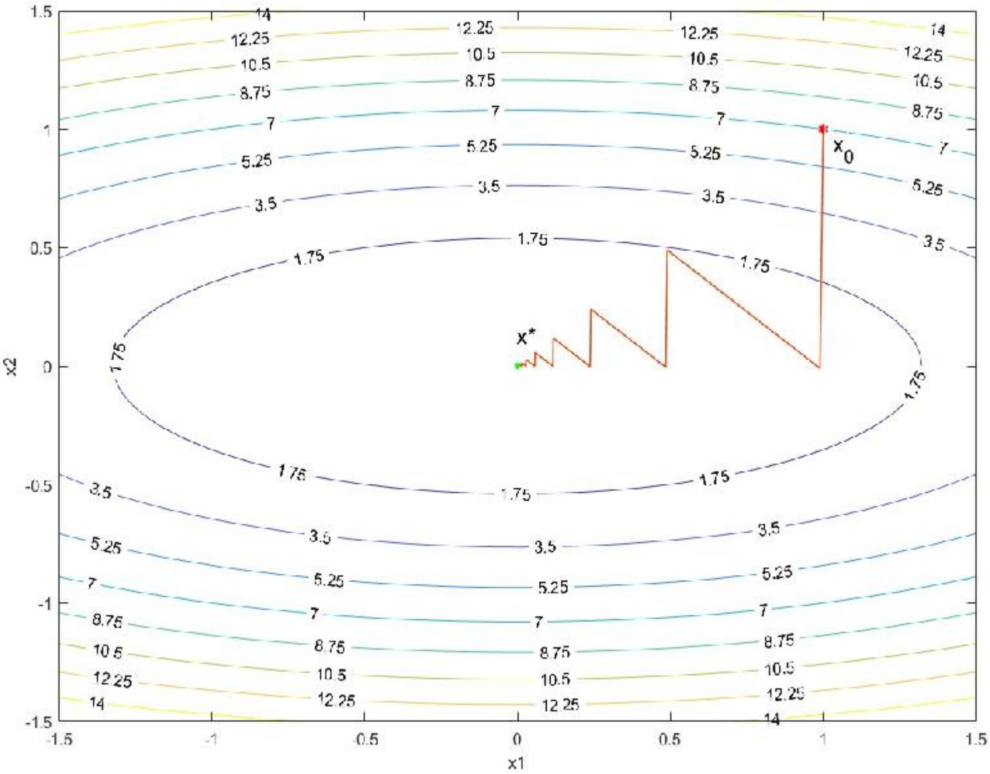

Рисунок 2 – Траєкторія методу **A** пошуку наближення до точки мінімуму квадратичної функції $f(x_1, x_2) = x_1^2 + 6x_2^2 : m = 2,\ f^* = 0,\ x_0 = (1,1)^T$

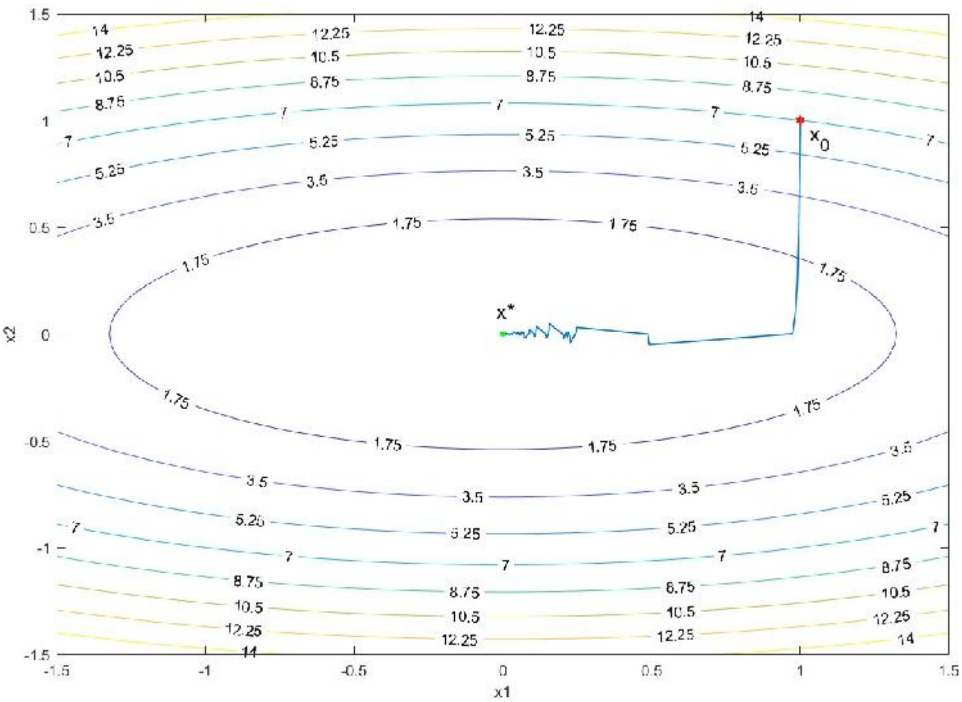

Рисунок 3 – Траєкторія методу **A** пошуку наближення до точки мінімуму квадратичної функції $f(x_1, x_2) = x_1^2 + 6x_2^2 : m = 1,\ f^* = 0,\ x_0 = (1,1)^T$

У табл. 1 наведені результати методу **А** для знаходження точки мінімуму функцій $f_1(x_1,x_2)=|x_1|+t|x_2|$ та $f_2(x_1,x_2)=x_1^2+tx_2^2$ для різних ступенів розтягу, які визначаються параметром t. Перший стовпчик містить точності від 10^{-1} до 10^{-10}, з якими обчислюються наближення до точки максимуму. Колонки 2, 3, 4 містять кількості ітерацій, які необхідні для мінімізації кусково-лінійної функції $f_1(x_1,x_2)=|x_1|+t|x_2|$ з $t=100, 50, 25$ відповідно. Колонки 5, 6, 7 показують кількості ітерацій для квадратичної функції $f_2(x_1,x_2)=x_1^2+tx_2^2$ з $t=10000, 1000, 100$, де кількість ітерацій подається в дужках, якщо параметр $m=1$ використовується для методу **А** замість $m=2$. З колонок 5 – 7 табл. 1 видно, що кількість ітерацій при $m=1$ є значно більшою, ніж кількість ітерацій при $m=2$.

Таблиця 1 – Кількість ітерацій методу **А** для знаходження $x^*_{£}$ функцій $f_1(x_1,x_2)=|x_1|+t|x_2|$ та $f_2(x_1,x_2)=x_1^2+tx_2^2$: $f^*=0,\ x^*=(1,1)^T$

$£_f$	$f_1(x_1,x_2)=\|x_1\|+t\|x_2\|$			$f_2(x_1,x_2)=x_1^2+tx_2^2$		
	$t=100$	$t=50$	$t=25$	$t=10000$	$t=1000$	$t=100$
1.0e–01	14930	3721	925	6(139)	6(97)	6(9)
1.0e–02	26443	6600	1645	10(295)	10(127)	10(29)
1.0e–03	37956	9478	2365	12(551)	12(194)	12(51)
1.0e–04	49469	12356	3084	16(646)	16(230)	16(68)
1.0e–05	60982	15234	3804	20(885)	20(324)	20(86)
1.0e–06	72495	18113	4523	22(1003)	22(390)	22(98)
1.0e–07	84008	20991	5243	26(1135)	26(418)	26(121)
1.0e–08	95521	2386	5962	30(1332)	30(485)	28(135)
1.0e–09	107034	26747	6682	32(1518)	32(513)	32(150)
1.0e–10	118547	29625	7401	36(1947)	36(543)	36(169)

З табл. 1 бачимо повільну збіжність методу **А**, якщо функції $f_1(x_1,x_2)$ та $f_2(x_1,x_2)$ є яружними. Чим більша витягнутість поверхонь рівня, якій відповідає більше значення параметра t ¸ тим повільніша збіжність методу **А**. Зі збільшенням точності знаходження наближення до точки мінімуму значно зростає й кількість ітерацій. До того, ситуація буде гіршою для кусково-лінійної функції $f_1(x_1, x_2)$. Так, наприклад, для досягнення точності 10^{-10} при $t = 100$ необхідно виконати 118547 ітерацій.

Однак скоротивши ступінь розтягу поверхонь рівня в 4 рази, для досягнення тієї ж точності необхідно суттєво менше ітерацій – 7401. У випадку мінімізації квадратичної функції ситуація дещо інша: для $m = 2$ кількість ітерацій практично не залежить від ступеню розтягу. Наприклад, для досягнення тієї ж точності 10^{-10} необхідно лише 36 ітерацій. Якщо використовується параметр $m = 1$ для квадратичної функції, то кількість ітерацій зростає при збільшенні параметра t.

Схожою буде ситуація для опуклих функцій $f(x_1, x_2) = x_1^4 + 10000x_2^4$ та $f(x_1, x_2) = \left(x_1 + 1.001x_2\right)^4 + \left(1.001x_1 + x_2\right)^4$, для яких $m = 4$. Так, при точності $\varepsilon_f = 10^{-20}$ для знаходження точки мінімуму першої функції знадобилось 4 ітерації при $m = 4$ та 50 ітерацій при $m = 1$, а для другої функції – 2 ітерації при $m = 4$ та 45 ітерацій при $m = 1$.

Нижче розглядається модифікація субградієнтного методу з кроком Поляка, де прискорення збіжності порівняно з методом **А** може бути забезпечене вибором матриці перетворення простору.

3. СУБГРАДІЄНТНИЙ МЕТОД З КРОКОМ ПОЛЯКА У ПЕРЕТВОРЕНОМУ ПРОСТОРІ

Крок Поляка у перетвореному просторі змінних. Розглянемо субградієнтний крок Поляка у перетвореному просторі змінних згідно ідеї Шора [5] для прискорення методів, використовуючи лінійні перетворення простору. Ця ідея дала метод еліпсоїдів і *r*-алгоритми (субградієнтні методи з розтягом простору в напрямку послідовних субградієнтів) для опуклої оптимізації [6, 7, 8].

Зробимо заміну змінних $x = By$, де B є невиродженою $n \times n$-матрицею (тобто для неї існує обернена матриця $A = B^{-1}$). Субградієнт $g_{\varphi}(x)$ опуклої функції $\varphi(y) = f(By)$ в точці $y = Ax$ задовольняє таку умову:

$$(y - y^*, g_{\varphi}(y)) \geq m(\varphi(x) - \varphi^*), \ \forall y \in R^n, \qquad (8)$$

де $g_{\varphi}(y) = B^T g_f(x)$, $\varphi^* = \varphi(y^*) = y^* = Ax^*$. Дійсно, оскільки $A = B^{-1}$ і $x = By$, нерівність (4) можна переписати у вигляді

$$(A(x - x^*), B^T g_f(x)) \geq m(f(By) - f(By^*)), \ \forall By \in R^n,$$

звідки отримуємо нерівність (8).

Субградієнтний метод з кроком Поляка у перетвореному просторі (визначений невиродженою матрицею B) має такий вигляд:

$$x_{k+1} = x_k - h_k B \frac{B^T g_f(x_k)}{\|B^T g_f(x_k)\|}, \quad h_k = \frac{m(f(x_k) - f^*)}{\|B^T g_f(x_k)\|}, \quad k = 0,1,2... \qquad (9)$$

Тут h_k є кроком Поляка (Аґмона–Моцкіна–Шонберга), але в перетвореному просторі змінних $y = Ax$. Це випливає з того, що в перетвореному просторі змінних метод (9) записується як субградієнтний процес

$$y_{k+1} = y_k - h_k \frac{g_{\varphi}(y_k)}{\|g_{\varphi}(y_k)\|}, \quad h_k = \frac{m(\varphi(y_k) - \varphi^*)}{\|g_{\varphi}(y_k)\|}, \quad k = 0,1,2... \qquad (10)$$

Крок Поляка у перетвореному просторі змінних має ті ж самі властивості, що й крок Поляка в початковому просторі. Вони визначаються мінімальним значенням функції φ^* та нерівністю (8).

Опис методу B. Щоб знайти точку $x_{\varepsilon}^* \in R^n$, для якої $f(x_{\varepsilon}^*) \leq f^* + \varepsilon$, субградієнтний метод з кроком Поляка у перетвореному просторі описується такою ітеративною процедурою.

Ініціалізація. Маємо f^* та $m \geq 1$. Виберемо початкову точку $x_0 \in R^n$, невироджену $n \times n$-матрицю B і величину $\varepsilon > 0$. Переходимо до наступної ітерації з величиною x_0.

Ітеративний процес. Нехай точка $x_k \in R^n$ була знайдена на *k*-й ітерації. Щоб перейти до наступної (*k*+1)-ї ітерації, виконаємо такі дії.

B1. Обчислимо $f(x_k)$ та $g_f(x_k)$. Якщо $f(x_k) - f^* \le \pounds$, тоді STOP ($k^* = k,\ x_{\pounds}^* = x_k$).

B2. Обчислимо наступну точку

$$x_{k+1} = x_k - h_k B \frac{B^T g_f(x_k)}{\left\| B^T g_f(x_k) \right\|}, \qquad h_k = \frac{m(f(x_k) - f^*)}{\left\| B^T g_f(x_k) \right\|}.$$

B3. Переходимо до наступної (*k*+1)-ї ітерації з x_{k+1}.

Теорема 2. Послідовність $\{x_k\}_{k=0}^{k^*-1}$, породжена методом **B**, задовольняє нерівності

$$\left\| A(x_{k+1} - x^*) \right\|^2 \le \left\| A(x_k - x^*) \right\|^2 - \frac{m^2 (f(x_k) - f^*)^2}{\left\| B^T g_f(x_k) \right\|^2}, \quad k = 0,1,2,... \tag{11}$$

Доведення. З **B2** для довільного $k\ (0 \le k \le k^* - 1)$ маємо

$$\left\| A(x_{k+1} - x^*) \right\|^2 = \left\| A(x_k - x^*) - h_k \frac{B^T g_f(x_k)}{\left\| B^T g_f(x_k) \right\|} \right\|^2 =$$

$$= \left\| A(x_k - x^*) \right\|^2 - 2h_k \frac{(x_k - x^*, g_f(x_k))}{\left\| B^T g_f(x_k) \right\|} + h_k^2 .$$

Беручи до уваги, що з (4) випливає нерівність

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

маємо

$$\left\| A(x_{k+1} - x^*) \right\|^2 \le \left\| A(x_k - x^*) \right\|^2 - h_k^2 = \left\| A(x_k - x^*) \right\|^2 - \left(\frac{m(f(x_k) - f)^*}{\left\| B^T g_f(x_k) \right\|} \right)^2 ,$$

що доводить нерівність (11). Теорема доведена.

Теорема 2 гарантує, що в субградієнтному методі з кроком Поляка у перетвореному просторі змінних відстань до точки мінімуму зменшується монотонно в перетвореному просторі змінних. Крім того, задовольняються такі нерівності:

$$(A(x^* - x_{k+1}), -B^T g_f(x_k)) \geq 0,\ k = 0,1,... \tag{12}$$

які можуть бути переписані як нерівності

$$(y^* - y_{k+1}, -g_{\cdot\cdot}(y_k)) \geq 0,\ k = 0,1,... \tag{13}$$

Дійсно, нерівності (12) випливають з того факту, що, використовуючи (8) і (10), маємо:

$$(x^* - x_{k+1}, -g_f(x_k)) = (x_{k+1} - x^*, g_f(x_k)) = (x_k - x^* - h_k B \frac{B^T g_f(x_k)}{\|B^T g_f(x_k)\|}, g_f(x_k)) =$$

$$= (x_k - x^*, g_f(x_k) - h_k \|B^T g_f(x_k)\|) = (x_k - x^*, g_f(x_k)) - m(f(x_k) - f^*) \geq 0 .$$

Нерівності (13) означають, що для опуклої функції ¨ (x), яка задовольняє умову (8), крок h_k визначає величину максимального переміщення в напрямку нормалізованого антисубградієнта, при якому гарантується, що кут між антисубградієнтом у точці y_k і напрямком від точки y_{k+1} до точки мінімуму y^* не буде тупим у перетвореному просторі змінних.

Octave-функція PolyakB і обчислювальні експерименти. Octave-функція **PolyakB** знаходить наближення $x^*_{£}$ опуклої функції $f(x)$, яка для методу **B** визначається такими вхідними даними: B − $n \times n$-матриця перетворення простору; x_0 − початкова точка; f^* − значення функції в точці мінімуму; $m \geq 1$ – параметр, що задовольняє умову (4); параметри зупинки $£_f$ та maxitn. Її код наведено нижче.

```
# Input parameters:
# calcfg - name of the function for calculation of f(x) and g(x)
# B - n*n-matrix for transformation of space
# x0 - the starting point, x0(1:n)
# fstar - value of the function at the minimum point
# m - length of shift along anti-subgradient (m>=1)
# epsf, maxitn - stop parameters
# intp - print information every intp iteration
# Output parameters:
```

```
# x - the minimum point, which was found by the program, x(1:n)
# f - the value of the function f at the point x
# itn - the number of iterations used by the program
# info - exit code (0 = epsf, 4 = maxitn)

function [x,f,itn,info] = PolyakB(calcfg,B,x0,fstar,m,     #row01
epsf,maxitn,intp);                                         #.....
itn=0; x=x0; [f,g] = calcfg(x);                            #row02
if(intp>0)                                                 #row03
printf("itn %4d f %14.6e \n", itn, f); # xprint = x',      #.....
endif                                                      #.....
for((itn = 1:maxitn)                                       #row04
   if(f-fstar < epsf) info = 0; return; endif              #row05
g1=B'*g; dg1=norm(g1);                                     #row06
g2=g1/dg1; hs=m*(f-fstar)/dg1;                             #row07
x -= hs * B * g2;                                          #row08
[f,g] = calcfg(x); dg=norm(g);                             #row09
if(mod(itn,intp)==0)                                       #row10
printf("itn %4d f %14.6e \n",itn,f); # xprint = x',        #.....
endif                                                      #.....
endfor                                                     #row11
info = 4;                                                  #row12
endfunction                                                #row13
```

Використовуючи програму **PolyakB**, можна знайти досить точні наближення до точки мінімуму опуклої функції. Якщо матрицю B вибрати так, що в перетвореному просторі змінних поверхні рівня яружних функцій витягнуті менше, ніж у вихідному просторі змінних, то метод **B** збігатиметься швидше, ніж метод **A**.

Наприклад, якщо матриця $B = diag\left(1;0.5\right)$, то для функції $f_1(x_1,x_2) = |x_1| + t|x_2|$, $t > 1$ швидкість збіжності методу **B** визначається геометричною прогресією із знаменником $q = \sqrt{1-4/t^2}$, який менший, ніж $q = \sqrt{1-1/t^2}$ для методу **A** (див. (7)). Якщо $m = 2$, $B = diag\left(1;0.7\right)$ і £ $= 0.01$, тоді при мінімізації функції $f_2(x_1,x_2) = x_1^2 + 6x_2^2$ за перших п'ять ітерацій метод **B** генерує послідовність $x_0 = (1.00,1.00)^T$, $x_1 = (0.624,-0.104)^T$, $x_2 = (0.136,0.136)^T$, $x_3 = (0.085,-0.014)^T$.

Для яружних функцій субградієнтний метод Поляка з перетворенням простору (метод **B**) буде ефективнішим, ніж субградієнтний метод Поляка без перетворення простору (метод **A**). Це узгоджується з кількістю ітерацій методу **B** для знаходження десяти послідовно уточнених наближень точки мінімуму функції $f_1(x_1, x_2) = |x_1| + 10|x_2|$ для шести різних матриць B, кожній з яких відповідає стовпчик у табл. 2.

Матриці B одержано як результат розтягу простору змінних у напрямку x_2 з коефіцієнтами розтягу $\sim\ = 1; 1.5; 2; 3; 4; 5$. Матриця B має вигляд $B = \begin{pmatrix} 1 & 0 \\ 0 & \frac{1}{\sim} \end{pmatrix}$,

і, якщо $\sim\ = 1$, вона збігається з одиничною матрицею. Випадок $\sim\ = 1$ відповідає методу **А**.

Таблиця 2 – Кількість ітерацій методу **B** для мінімізації функції $f_1(x_1, x_2) = |x_1| + 10|x_2|$, $x_0 = (1,1)^T$

$£_f$	$\sim\ = 1$	$\sim\ = 1.5$	$\sim\ = 2$	$\sim\ = 3$	$\sim\ = 4$	$\sim\ = 5$
1.0e–01	147	63	33	6	10	9
1.0e–02	262	114	62	19	17	13
1.0e–03	377	165	91	31	24	18
1.0e–04	492	216	119	44	31	22
1. 0e–05	607	268	148	57	38	27
1.0e–06	722	319	177	70	45	31
1.0e–07	837	370	206	82	53	36
1.0e–08	952	421	234	95	60	40
1.0e–09	1068	472	263	108	67	45
1.0e–10	1183	523	292	121	74	49

З табл. 2 бачимо, що кількість ітерацій методу Поляка з перетворенням простору зменшується монотонно, коли ступінь яружної функції $\ddot{}_1(y_1, y_2) = |y_1| + \frac{10}{\sim}|y_2|$ зменшується в перетвореному просторі змінних (що відповідає збільшенню коефіцієнта розтягу $\sim$).

Ситуація зі швидкістю збіжності буде складнішою для суттєво-яружної функції $f_3(x_1, x_2) = \max\left\{x_1^2 + (2x_2 - 2)^2 - 3,\ x_1^2 + (x_2 + 1)^2\right\}$. З табл. 3 бачимо, що монотонного зменшення кількості ітерацій не спостерігається, якщо величина $\sim$, коефіцієнт розтягу простору в напрямку x_2, збільшується. Але є певний розрив, який має місце при значенні коефіцієнта $\sim\ = 2$ (відповідає колонці з $\sim\ = 2$).

Таблиця 3 – Кількість ітерацій методу **B** для мінімізації функції $f_3(x_1,x_2)=\max\left\{x_1^2+(2x_2-2)^2-3,\ x_1^2+(x_2+1)^2\right\},\ x_0=(1,1)^T$, maxitn = 100 000

$\pounds_f$	$\sim=1$	$\sim=1.5$	$\sim=2$	$\sim=3$	$\sim=4$	$\sim=5$
1.0e–01	16	4	4	5	7	8
1.0e–02	162	37	4	6	7	9
1.0e–03	1604	679	5	6	8	9
1.0e–04	16004	7079	5	6	9	46
1. 0e–05	−	71079	6	8	8061	6206
1.0e–06	−	−	6	−	98061	63806
1.0e–07	−	−	6	−	−	−

На рис. 4 зображені траєкторії методів **A** ($\sim=1$) та **B** ($\sim=2$) суцільною та штриховою лініями відповідно. На траєкторії методу **B** зображені всі точки, отримані алгоритмом, на траєкторії методу **A** зображені лише перші чотири точки.

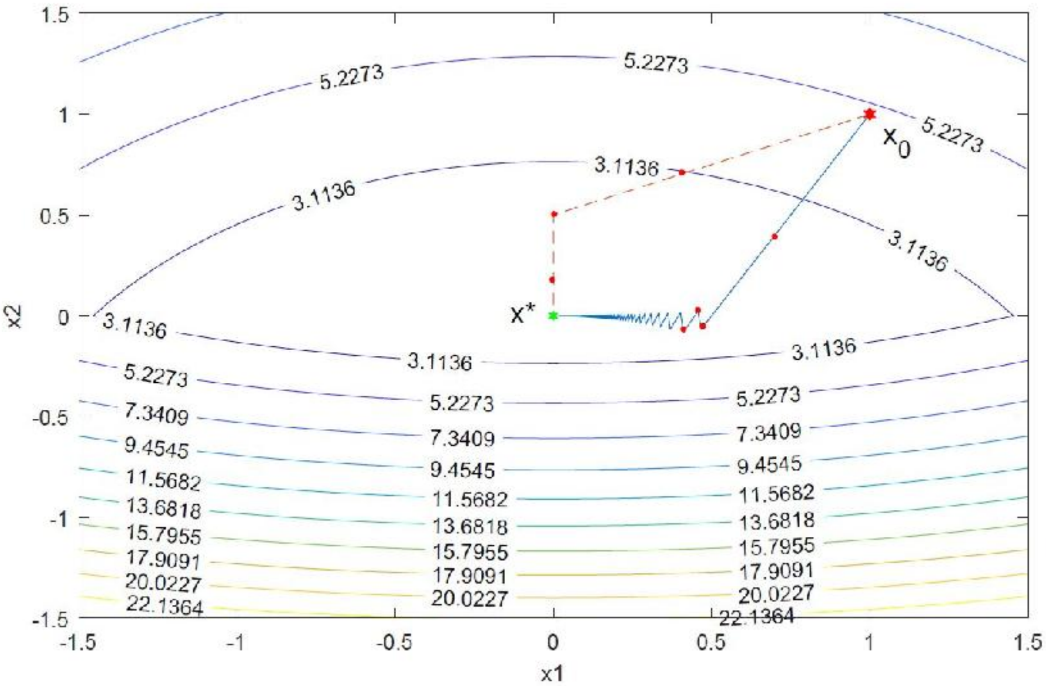

Рисунок 4 – Траєкторія методу **A** (суцільна лінія) і траєкторія методу **B** (штрихова лінія) для знаходження наближення до точки мінімуму кусочно-квадратичної функції $f_3(x_1,x_2)$: $m=1,\ f^*=1,\ x_0=(1,1)^T$, B = diag(1;0.5), $x^*=(0,0)$

Розглянемо використання програм **PolyakA** і **PolyakB** для мінімізації опуклої квадратичної функції $f_4(x) = \|Ax - b\|^2$ від змінних $x \in R^n$, де $A = \|a_{ij}\|_{i,j=1}^{l,n}$ – довільна $l \times n$-матриця і l-вимірний вектор b такий, що його компоненти $b_i = \sum_{j=1}^{n} a_{ij}$, $i = 1,..,l$. Для функції $f^* = 0$ і, якщо матриця A має повний ранг, то функція має єдину точку мінімуму $x^* = (1,1,..,1)^T$.

Octave-функція для обчислення значення функції $f_4(x)$ та її градієнта має такий вигляд:

```
function [f,g] = fgfun4(x)
global A b;
temp=A*x- b;
f = temp’*temp;
g=2*A’*temp;
endfunction
```

Розглянемо тестовий приклад з 500×100-матрицею, що має вигляд

$$A = \begin{pmatrix} 100 & 0 & 0...0 \\ 0 & 100 & 0...0 \\ & A_1 & \end{pmatrix},$$

де A_1 – 498×100-матриця, яка отримана за допомогою генератора випадкових чисел з відрізка [0, 3].

Код octave-програми для порівняння методів **A** і **B**, де матриця $B = diag\left(0.1;\, 0.1;\, 1;...;\, 1\right)$, є таким:

```
global A b;
n=100; x0 = zeros(n,1); B=diag([0.1 0.1 ones(1,n-2)]);
rand("seed", 2018); A1=3*rand(498,n);
A=[diag([100 100]) zeros(2,n-2); A1]; b=sum(A’)’;
m=2; fstar=0.d0; epsf = 0.0001; maxitn = 50000; intp=1000;
for(itn = 1:6)
 [xA,fA,itnA,infoA]=PolyakA(@fgfun4,x0,fstar,m,epsf,maxitn,intp);
 [xB,fB,itnB,infoB]=PolyakB(@fgfun4,B,x0,fstar,m,epsf,maxitn,intp);
 epsf, itnA, dnA=norm(xA-ones(n,1)), itnB, dnB=norm(xB-ones(n,1)),
 epsf=epsf/100;
endfor
```

Результати роботи цієї програми є такими:

```
epsf=1.0e-04  itnA= 695  dnA=8.0612e-04  itnB= 86  dnB=1.2175e-04
epsf=1.0e-06  itnA=1085  dnA=8.2861e-05  itnB=108  dnB=1.1773e-05
epsf=1.0e-08  itnA=1491  dnA=8.3470e-06  itnB=130  dnB=1.1353e-06
epsf=1.0e-10  itnA=1901  dnA=8.3881e-07  itnB=150  dnB=1.3526e-07
epsf=1.0e-12  itnA=2313  dnA=8.3994e-08  itnB=172  dnB=1.3018e-08
epsf=1.0e-14  itnA=2725  dnA=8.4440e-09  itnB=194  dnB=1.2523e-09
```

З даної таблиці видно, що метод **B** потребує значно меншу кількість ітерацій, ніж метод **A**. Наприклад, щоб досягнути точності 10^{-14} методу **A** необхідно виконати 2725 ітерацій, тоді як метод **B** потребує лише 194 ітерації.

4. МЕТОД ТА ПРОГРАМА AMSG2P

Субградієнтний метод **amsg2p** дозволяє знайти точку мінімуму опуклої функції $f(x)$ при відомому значенні f^* або його верхній оцінці [10, с. 386–387]. Його детальний опис наведено у звіті [11]. У назві методу «ams» означає, що в ітераційному процесі використовується крок Агмона–Моцкіна–Шонберга (крок Поляка) в напрямку нормованого антисубградієнта, а «g2p» вказує, що ams-крок використовується в просторі змінних, перетвореному за допомогою двох послідовних субградієнтів (g2) та агрегатного вектора (p). Перетворення простору в методі **amsg2p** проводиться за допомогою однорангового еліпсоїдального оператора [12, 13] і лише на тих ітераціях методу, коли тупим є хоча б один з кутів – або кут між двома послідовними субградієнтами, або кут між останнім субградієнтом та агрегатним вектором, який є опуклою комбінацією обчислених на попередніх ітераціях субградієнтів.

Нехай $f(x)$ – опукла функція векторного аргументу $x \in R^n$, а її субградієнт $g_f(x)$ задовольняє умову

$$(x - x^*, g_f(x)) \geq \gamma(f(x) - f^*), \quad \forall x \in R^n, \forall x^* \in X^*, \quad \gamma \geq 1. \qquad (14)$$

Тут X^* – множина точок мінімуму функції $f(x)$; f^* – мінімальне значення функції $f(x)$: $f^* = f(x^*)$, $x^* \in X^*$. Параметр γ вважається відомим та введений для врахування спеціальних класів опуклих функцій.

Він позначається інакше, ніж аналогічний йому параметр m у співвідношенні (4), щоб відрізнити випадок $X^* = x^*$ (точка мінімуму єдина) від загального випадку. Значення $\gamma = 1$ забезпечує виконання умови (14) для довільної опуклої функції.

Метод **amsg2p** дає можливість або знайти таку точку, де значення опуклої функції $f(x)$ менше або дорівнює $f_{min} + \varepsilon_f$, або гарантує достатню умову того, що точки, в якій значення $f(x)$ дорівнює f_{min}, у кулі заданого радіуса не існує. У першому випадку метод **amsg2p** знаходить точку $x^*_\varepsilon \in \left\{ x : f(x) - f_{min} \le \varepsilon_f \right\}$ і відповідний їй номер ітерації позначається k^*_ε, а в другому – зупиняється з повідомленням «точки не існує».

Метод **amsg2p** полягає в наступному.

На ітерації $k = 0$ задані: початкове наближення $x_0 \in R^n$; початковий радіус r_0 такий, що $\left\| x_0 - x^* \right\| \le r_0$; досить мале $\varepsilon_f > 0$. Обчислимо $f(x_0)$ та $g_f(x_0)$. Якщо $f(x_0) - f_{min} \le \varepsilon$, тоді $x^*_\varepsilon = x_0$, $k^*_\varepsilon = 0$, і алгоритм закінчує роботу. Інакше, визначаємо $h_0 = \dfrac{\gamma(f(x_0) - f_{min})}{\left\| g_f(x_0) \right\|}$, $\xi_0 = \dfrac{g_f(x_0)}{\left\| g_f(x_0) \right\|} \in R^n$, $p_0 = 0 \in R^n$, $B_0 = I$ – одинична $n \times n$–матриця. Перейдемо до наступної ітерації.

Нехай на k-й ітерації отримано $x_k \in R^n$, h_k, r_k, $\xi_k \in R^n$, $p_k \in R^n$, B_k – матриця $n \times n$. Для переходу на $(k+1)$-у ітерацію виконаємо такі кроки.

Крок 1. Обчислимо $t_k = h_k / r_k$. Якщо $t_k > 1$, тоді "точки не існує" і алгоритм завершує роботу. Інакше – визначимо $r_{k+1} = r_k \sqrt{1 - t_k^2}$ та обчислимо чергове наближення

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

Крок 2. Обчислимо $f(x_{k+1})$ і $g_f(x_{k+1})$. Якщо $f(x_{k+1}) - f_{min} \le \varepsilon$, то $x^*_\varepsilon = x_{k+1}$, $k^*_\varepsilon = k + 1$ і алгоритм завершує роботу. Інакше – визначимо

$$\xi_{k+1} = \frac{B_k^T g_f(x_{k+1})}{\left\| B_k^T g_f(x_{k+1}) \right\|}, \quad h_{k+1} = \frac{\gamma(f(x_{k+1}) - f_{min})}{\left\| B_k^T g_f(x_{k+1}) \right\|}.$$

Крок 3. Обчислимо $\lambda_1 = -p_k^T \xi_{k+1}$ і $\lambda_2 = -\xi_k^T \xi_{k+1}$. Визначимо

$$p_{k+1} = \begin{cases} \frac{\lambda_1}{\sqrt{\lambda_1^2+\lambda_2^2}} p_k + \frac{\lambda_2}{\sqrt{\lambda_1^2+\lambda_2^2}} \xi_k, & \text{якщо } \lambda_1 > 0 \ i \ \lambda_2 > 0, \\ p_k, & \text{якщо } \lambda_1 > 0 \ i \ \lambda_2 \le 0, \\ \xi_k, & \text{якщо } \lambda_1 \le 0 \ i \ \lambda_2 > 0, \\ 0, & \text{якщо } \lambda_1 \le 0 \ i \ \lambda_2 \le 0. \end{cases}$$

Крок 4. Обчислимо $\mu_k = p_{k+1}^T \xi_{k+1}$. Якщо $-1 < \mu_k < 0$, то обчислимо

$$B_{k+1} = B_k + (B_k \eta) \xi_{k+1}^T, \text{ де } \eta = \left(\frac{1}{\sqrt{1-\mu_k^2}} - 1 \right) \xi_{k+1} - \frac{\mu_k}{\sqrt{1-\mu_k^2}} p_{k+1}$$

і перерахуємо

$$h_{k+1} = \frac{h_{k+1}}{\sqrt{1-\mu_k^2}}, \qquad p_{k+1} = \frac{1}{\sqrt{1-\mu_k^2}} (p_{k+1} - \mu_k \xi_{k+1}).$$

Інакше – визначимо $B_{k+1} = B_k$ і $p_{k+1} = 0$.

Крок 5. Перейдемо до наступної ітерації з x_{k+1}, h_{k+1}, r_{k+1}, ξ_{k+1}, p_{k+1}, B_{k+1}.

Теорема 3 [11]. Нехай $A_k = B_k^{-1}$, $A_{k+1} = B_{k+1}^{-1}$. Якщо $f_{min} \ge f^*$ та $X^* = x^*$, то для всіх точок, згенерованих методом **amsg2p**, справедливі нерівності

$$\left\| A_{k+1}(x_{k+1} - x^*) \right\|^2 \le \left\| A_k (x_k - x^*) \right\|^2 - \left(\frac{\gamma(f(x_k) - f_{min})}{\left\| B_k^T g_f(x_k) \right\|} \right)^2, \ k = 0,1,\ldots, \qquad (15)$$

Теорема 3 означає, що в кожному черговому перетвореному просторі змінних відстань до точки мінімуму монотонно зменшується. Завдяки цьому на кожній ітерації $k > 1$ виконується нерівність

$$\left\| A_k (x_k - x^*) \right\|^2 \le \left\| x_0 - x^* \right\|^2 - \sum_{i=0}^{k-1} \left(\frac{\gamma(f(x_i) - f_{min})}{\left\| B_i^T g_f(x_i) \right\|} \right)^2 = r_0^2 - \sum_{i=0}^{k-1} h_i^2 = r_k^2,$$

за допомогою якої забезпечується достатня умова відсутності точки x_ε^* при $f_{min} < f^* - \varepsilon_f$ (реалізовано на кроці 1 методу **amsg2p**).

Подібно до того, як це зроблене в r-алгоритмах Шора, перетворення простору в методі **amsg2p** направлене на зменшення ступеня яружності поверхонь рівня опуклих функцій. Для яружних функцій це забезпечує прискорену збіжність методу при довільній початковій стартовій точці x_0 та досить малих значенннях параметра ε_f ($\varepsilon_f \approx 10^{-14} \div 10^{-10}$).

А якщо функція не яружна, то метод **amsg2p** переходить у відомий субградієнтний метод Поляка:

$$x_{k+1} = x_k - h_k \frac{g_f(x_k)}{\|g_f(x_k)\|}, \quad h_k = \frac{\gamma\left(f(x_k) - f^*\right)}{\|g_f(x_k)\|}, \quad k = 0,1,2,\ldots. \qquad (16)$$

який описано в розділі 2. Тут величину h_k називають кроком Поляка або кроком Агмона–Моцкіна–Шонберга, які вперше використали таке регулювання кроку в релаксаційному субградієнтному методі для знаходження хоча б одного розв'язку сумісної системи лінійних нерівностей.

Теорема 4 [11]. Для всіх точок, згенерованих методом (16), справедливі нерівності

$$\|x_{k+1} - x^*\|^2 \le \|x_k - x^*\|^2 - \left(\frac{\gamma(f(x_k) - f^*)}{\|g_f(x_k)\|}\right)^2, \quad \forall x^* \in X^*, k = 0,1,\ldots, \quad (17)$$

Програмна реалізація методу **amsg2p** виконана мовою Octave. Однойменна програма **amsg2p** знаходить точку x_ε^*, де значення опуклої функції $f(x_\varepsilon^*) \le f_{\min} + \varepsilon_f$ і визначається заданими: стартовою точкою x_0; радіусом r_0, параметром γ (зміщення по опуклості), параметрами зупинки ε_g, ε_x та **maxitn**.

Програма використовує octave-функцію **function [f,g] = calcfg(x)**, яка в точці x обчислює значення функції $f = f(x)$ та її субградієнта $g = g_f(x)$. Ім'я функції **calcfg(x)** може бути довільним, яке допускає синтаксис мови Octave.

Програма **amsg2p** використовує такі вхідні та вихідні параметри:

```
%  Вхідні параметри:
%    calcfg – ім’я функції calcfg(x) для обчислення f і g
%    x0  -- стартова точка
%    fmin – мінімальне значення функції (або його оцінка)
%    gamma – параметр зміщення по опуклості
%    rad – радіус початкової кулі з центром у точці x0
%    epsf, epsg, maxitn – параметри зупинки
%  Вихідні параметри:
%    x -- знайдена точка
%    f -- значення функції f в точці x
%    itn -- кількість проведених ітерацій
%    istop -- код зупинки (0=epsf, 1=epsg, 2=maxitn, 3=error).
```

Точка x_{itn} – це остання точка ітераційного процесу та її статус характеризується значенням параметра зупинки **istop** на ітерації **itn**:

1. зупинка відбулася за умовою $f(x_{itn}) \le f_{\min} + \varepsilon_f$ (**istop=0**);
2. зупинка відбулася за умовою $\| g_f(x_{itn}) \| \le \varepsilon_g$ (**istop=1**);
3. зупинка відбулася за умовою **itn>maxitn** (перевищена максимальна кількість ітерацій) (**istop=2**);
4. зупинка відбулася тому, що отримана достатня умова того, що точки, де значення $f(x)$ дорівнює f_{min}, не існує в кулі заданого радіусу (**istop=3**) (ця зупинка вважається аварійною і пов’язана або з тим, що вказане значення **fmin** менше від мінімального значення функції, або початковий радіус **rad** занижений і його варто збільшити).

Програма **amsg2p** оформлена як octave-функція:

```
# octave-function of amsg2p-method
function [x,f,itn,istop]=amsg2p(calcfg,x0,fmin,gamma,rad,      # row001
                              epsf,epsg,maxitn);
itn=0; rad1=rad; B=eye(length(x0)); x=x0;                      # row002
[f,g]=calcfg(x); if(f-fmin<epsf) istop=0; return; endif        # row003
dg=norm(g); if(dg<epsg) istop=1; return; endif                 # row004
hs=gamma*(f-fmin)/dg; g1=g/dg; p=zero=zeros(length(x),1);      # row005
printf("itn %4d f %21.13e rad %10.2e \n",itn, f, rad1);        # row006
for (itn = 1:maxitn)                                           # row007
   tmp=hs/rad1; if(tmp>1.d0) istop=3; return; endif            # row008
   rad1=rad1*sqrt(1.d0-tmp)*sqrt(1.d0+tmp);                    # row009
   x -= hs * B * g1; [f,g] = calcfg(x);                        # row010
   if(f-fmin < epsf) istop=0; return; endif                    # row011
   if(norm(g) < epsg) istop=1; return; endif                   # row012
   g2=B’*g; dg2=norm(g2); g2=g2/dg2; t1=-p’*g2; t2=-g1’*g2;    # row013
```

```
    if(t1>0.d0) if(t2>0.d0) tmp=sqrt(t1*t1+t2*t2);              # row014
      t1=t1/tmp; t2=t2/tmp; p=t1*p+t2*g1; endif                 # row015
    else if(t2>0.d0) p=g1; else p=zero; endif endif             # row016
    hs=gamma*(f-fmin)/dg2; tcos=p'*g2; tmp=1.d0;                # row017
    if(-1.d0<tcos)&&(tcos<0.d0) tmp=sqrt((1-tcos)*(1+tcos));    # row018
      g1=(1.d0/tmp-1.d0)*g2-(tcos/tmp)*p; B+=B*g1*g2';          # row019
      hs=hs/tmp; p=(p-tcos*g2)/tmp; else p=zero; endif          # row020
    g1=g2;                                                      # row021
    printf("itn %4d f %21.13e rad %9.1e cos %9.1e dB %9.1e\n",  # row022
           itn, f, rad1, tcos,tmp);
endfor                                                          # row023
istop=2;                                                        # row024
endfunction                                                     # row025
```

У табл. 4 та 5 наведені порівняльні результати обчислювальних експериментів для знаходження за допомогою методів **A**, **B** та **amsg2p** точки мінімуму квадратичної функції $f_4(x)=\|Ax-b\|^2$, для якої обчислення значення функції та її градієнта реалізує octave-функція **`fgfun4(x)`** (див. розділ 3). Розглядався той самий тестовий приклад з 500×100-матрицею $A=\begin{pmatrix}100 & 0 & 0...0\\ 0 & 100 & 0...0\\ & A_1 & \end{pmatrix}$, де A_1 – 498×100-матриця, яка була отримана за допомогою генератора випадкових чисел з інтервалів [0, 3] та [0, 5], відповідно.

Таблиця 4 – Збіжність методів **A**, **B, amsg2p** при мінімізації функції $f_4(x)=\|Ax-b\|^2,\ A_1\in[0,3]^{498\times100}$

Epsf	**Метод A**		**Метод B**		**Метод amsg2p**	
	itn	$\|x_{itn}-x^*\|$	**itn**	**time**	**itn**	**time**
1.00E–04	695	8.0612E–04	86	1.2175E–04	12	2.4107E–04
1.00E–06	1085	8.2861E–05	108	1.1773E–05	14	3.6769E–05
1.00E–08	1491	8.3470E–06	130	1.1353E–06	19	2.1710E–06
1.00E–10	1901	8.3881E–07	150	1.3526E–07	22	3.1324E–07
1.00E–12	2313	8.3994E–08	172	1.3018E–08	24	4.2297E–08
1.00E–14	2725	8.4439E–09	194	1.2523E–09	27	3.8820E–09
1.00E–16	3139	8.4141E–10	216	1.2040E–10	30	3.1344E–10
1.00E–18	3553	8.3820E–11	238	1.1637E–11	33	2.6745E–11
1.00E–20	3967	8.4642E–12	260	1.1283E–12	35	5.0263E–12

Таблиця 5 – Збіжність методів **A**, **B, amsg2p** при мінімізації функції $f_4(x) = \|Ax - b\|^2$, $A_1 \in [0,5]^{498\times100}$

epsf	Метод A		Метод B		Метод amsg2p	
	itn	$\|x_{itn} - x^*\|$	**itn**	**time**	**itn**	**time**
1.00E–04	265	4.9693E–04	352	1.0379E–04	12	2.1617E–04
1.00E–06	399	5.2130E–05	438	1.0776E–05	15	2.5641E–05
1.00E–08	539	5.2116E–06	526	1.0616E–06	20	1.3339E–06
1.00E–10	681	5.1205E–07	614	1.0459E–07	23	1.9892E–07
1.00E–12	821	5.2309E–08	700	1.0862E–08	26	1.8676 E–08
1.00E–14	963	5.1901E–09	788	1.0704E–09	28	2.4180E–09
1.00E–16	1105	5.1594E–10	876	1.0541E–10	31	1.9506E–10
1.00E–18	1247	5.1462E–11	964	1.0372E–11	33	2.1189E–11
1.00E–20	1389	5.1388E–12	1052	1.0594E–12	36	2.1323E–12

У розрахунках використовувався параметр $\gamma = 2$, $f_{min} = f^* = 0$, $x_0 = (0,\ldots,0)^T$. З табл. 4 та 5 видно, що робота методу **A** є більш, ніж удвічі кращою, коли випадкові числа для заповнення матриці A_1 беруться з інтервалу [0, 5], а не з інтервалу [0, 3]. Натомість, метод **B** демонструє прямо протилежні результати і працює більш, ніж учетверо швидше, якщо випадкові числа беруться з інтервалу [0, 3]. А от робота методу **amsg2p** не залежить від вибору інтервалу – результати майже повністю співпадають.

В обох прикладах кількість ітерацій методу **amsg2p** є суттєво меншою, ніж кількості ітерацій методів **A** та **B**. Тому метод **amsg2p** можна успішно використовувати для мінімізації яружних опуклих функцій, а матрично-векторні операції роблять його перспективним у системах паралельних та розподілених обчислень.

Наявні бібліотеки стандартних програм для паралельних матрично-векторних операцій дозволяють за короткий термін адаптувати алгоритм для ефективної роботи з використанням векторних процесорів на основі графічних прискорювачів (GPU).

ВИСНОВКИ

Розглянуті у розділі субградієнтні методи з кроком Поляка можуть бути використані для знаходження допустимої точки сумісної системи опуклих нерівностей $f_i(x) \le 0,\ i = 1,..,l,\ x \in R^n$. Ця задача еквівалентна мінімізації негладкої опуклої функції $\cdot\ (x) = \max\left\{0, \max\limits_{1 \le i \le l} f_i(x)\right\}$, для якої $\cdot\ ^{*} = 0$. Частковим випадком системи опуклих нерівностей є система лінійних нерівностей.

Задачі лінійного програмування можна звести до цього випадку, використовуючи обмеження для прямих і двоїстих задач. При мінімізації $\cdot\ (x)$ рекомендується використовувати значення параметра $m = 1$. Параметр $m = 2$ може бути використаний для знаходження розв'язку сумісної системи лінійних рівнянь (повного рангу, недовизначеної або перевизначеної).

Список літератури

1. Стецюк П.И. Ускорение субградиентного метода Поляка. *Теорія оптимальних рішень*. 2012. № 11. С. 151 – 60.
2. Stetsyuk P., Stovba V., Chernousova Z. Subgradient Method with Polyak's Step in Transformed Space. *Optimization and Applications, OPTIMA 2018, Communications in Computer and Information Science 974 (2019)*. P. 49–63.
3. Поляк, Б.Т.: Минимизация негладких функционалов. *Вычислительная математика и математическая физика*. 1969. Том 9. № 3. С. 509–521.
4. Agmon S. The relaxation method for linear inequalities. *Canad. J. Math.* 6 (1954). P. 382−392.
5. Motzkin T., Schoenberg I.J. The relaxation method for linear inequalities. *Canad. J. Math.* 6 (1954). P. 393−404.
6. Еремин И.И. Обобщение релаксационного метода Агмона-Моцкина. *УМН*. 1965.Том XX. Вып. 2 (122). С. 183–187.
7. Сергиенко И.В., Стецюк П.И. О трех научных идеях Н.З. Шора. *Кибернетика и системный анализ*. 2012. № 1. С. 4–22.

8. 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.).
9. Shor N.Z. Nondifferentiable optimization and polynomial problems. Boston; Dordrecht; London: Kluwer Academic Publishers. 1998. 412 p.
10. Стецюк П.И. Методы эллипсоидов и r-алгоритмы. Кишинэу: Эврика. 2014. 488 с.
11. Стецюк П.І., Журбенко М.Г., Березовський О.А. та ін. Розробити субградієнтні алгоритми розв'язання задач оптимізації з гарантованою точністю. *Заключний звіт про науково-дослідну роботу № держ. реєстрації 0114U001055.* – К.: Ін-т кібернетики імені В.М. Глушкова НАН України, 2016. 235 с.
12. Stetsyuk P.I. 2d-Ellipsoid of optimal volume and its applications. *In: L.N. Polyakova (ed.) Constructive Nonsmooth Analysis and Related Topics (Dedicated to the Memory of V.F. Demyanov) (CNSA).* IEEE, 2017. P. 303–306.
13. Стецюк П.И. Оптимальный по объему 2d-эллипсоид и его приложения. *Тезисы докладов Международной конференции "Конструктивный негладкий анализ и смежные вопросы", посвященной памяти профессора В.Ф. Демьянова. Часть II.* СПБ.: Издательство ВВМ, 2017. С. 173–176.

ЗМІСТ

СУБГРАДІЄНТНІ МЕТОДИ З КРОКОМ ПОЛЯКА ТА ПРОГРАМА AMSG2P

РОБОЧІ МАТЕРІАЛИ.

Випуск 19-03

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

СТОВБА Віктор Олександрович,
аспірант Інституту кібернетики імені В.М. Глушкова НАН України.
vik.stovba@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 19 березня 2019 року

Попередні випуски «Робочих матеріалів»

19-01. Стецюк П.І., Бєлих Т.В., Криворучко О.І. Теорія та програмні реалізації *r*-алгоритмів Шора, 28 с.

19-02. Чикрій Г.Ц. Принцип розтягування часу при прийнятті рішень в умовах конфлікту та невизначеності, 32 с.