OCTAVE-ПРОГРАМА **ralgb5a** – *r(α)*-АЛГОРИТМ З АДАПТИВНИМ КРОКОМ

ПОСІБНИК КОРИСТУВАЧА

АНОТАЦІЯ

“**Octave-програма ralgb5a – *r(α)*-алгоритм з адаптивним кроком**”

Документ містить першу редакцію посібника користувача, необхідну для роботи з дослідницькою програмою **ralgb5a**, яка призначена для мінімізації гладких та негладких опуклих функцій.

Програма **ralgb5a** реалізує подану нижче модифікацію *r*-алгоритмів Шора – субградієнтних методів з розтягом простору в напрямку різниці двох послідовних субградієнтів. Ця модифікація використовує адаптивний спосіб регулювання кроку в напрямі антисубградієнта у перетвореному просторі змінних.

Програма розроблена на некомерційній мові GNU Octave.

ЗМІСТ

1. Про *r(α)*-алгоритми та Octave-функцію ralgb5a

Для мінімізації гладких та негладких опуклих функцій можна використовувати подану нижче модифікацію *r*-алгоритмів Шора – субградієнтних методів з розтягом простору в напрямку різниці двох послідовних субградієнтів [1 – 3]. Ця модифікація відома як $r(\alpha)$-алгоритми, де α – коефіцієнт розтягу простору ($\alpha > 1$), який є одним і тим самим на кожній ітерації *r*-алгоритмів. Опис $r(\alpha)$-алгоритмів наведемо згідно [3].

1.1 Що таке *r(α)*-алгоритми? Розглянемо задачу пошуку точки мінімуму опуклої функції $f(x)$, $x \in R^n$. Мінімальне значення функції позначимо $f^* = f(x^*)$, де $x^* \in X^*$. Будемо вважати, що множина мінімумів X^* є обмеженою, тобто виконується умова $\lim\limits_{\|x\| \to \infty} f(x) = +\infty$, яка забезпечує коректність адаптивного регулювання кроку в r-алгоритмах.

Означення. $r(\alpha)$-Алгоритмом мінімізації функції $f(x)$ називається ітеративна процедура знаходження послідовності n-вимірних векторів $\{x_k\}_{k=0}^{\infty}$ та послідовності $n \times n$-матриць $\{B_k\}_{k=0}^{\infty}$ за таким правилом:

$$x_{k+1} = x_k - h_k B_k \xi_k, \quad B_{k+1} = B_k R_\beta(\eta_k), \quad k = 0,1,2,\ldots, \tag{1}$$

де

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

$$\eta_k = \frac{B_k^T r_k}{\| B_k^T r_k \|}, \quad r_k = g_f(x_{k+1}) - g_f(x_k). \tag{3}$$

Тут x_0 – стартова точка; $B_0 = I_n$ – одинична $n \times n$-матриця[1]; h_k^* – величина кроку до точки мінімуму функції $f(x)$ у напрямку нормованого антисубградієнта в перетвореному просторі змінних; $R_\beta(\eta) = I_n + (\beta - 1)\eta\eta^T$ – оператор стиснення простору субградієнтів у нормованому напрямку η з

[1]В якості матриці B_0 часто вибирають діагональну матрицю D_n з додатними коефіцієнтами на діагоналі, за допомогою якої здійснюється масштабування змінних.

коефіцієнтом $\beta = 1/\alpha < 1$; $g_f(x_k)$ і $g_f(x_{k+1})$ – субградієнти функції $f(x)$ в точках x_k та x_{k+1}. Якщо на ітерації k процесу (1) – (3) виконані деякі критерії (умови) зупинки, то вважаємо $k^* = k$, $x_k^* = x_k$ і закінчуємо роботу алгоритму.

Коментар. На кожній ітерації r-алгоритмів реалізується субградієнтний спуск для опуклої функції $\varphi(y) = f(B_k y)$ в перетвореному просторі змінних $y = A_k x$, де $A_k = B_k^{-1}$. Дійсно, якщо обидві частини формули $x_{k+1} = x_k - h_k B_k \xi_k$ домножити зліва на матрицю A_k, то отримаємо

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

де вектор $g_\varphi(y_k) = B_k^T g_f(x_k)$ є субградієнтом функції $\varphi(y) = f(B_k y)$ в точці $y_k = A_k x_k$ простору змінних $y = A_k x$. Дійсно, субградієнт $g_f(x_k)$ для опуклої функції $f(x)$ задовільняє нерівності

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

звідки, здійснюючи заміну змінних $x = B_k y$, отримаємо нерівність

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

для субградієнта $g_\varphi(y_k)$ опуклої функції $\varphi(y) = f(B_k y)$.

Сімейство $r(\alpha)$-алгоритмів визначається коефіцієнтом розтягу простору $\alpha > 1$ (допускається $\alpha = \infty$, що відповідає $\beta = 0$) і послідовністю величин кроків $\{h_k\}_{k=0}^{\infty}$., які визначають ті два послідовні субградієнти $g_f(x_k)$ і $g_f(x_{k+1})$, розтягування за різницею яких (див. формулу (3)) зменшує степінь витягнутості функції в перетвореному просторі змінних. Вибір коефіцієнта $\alpha > 1$ та величин $\{h_k\}_{k=0}^{\infty}$ у поєднанні з вибором критеріїв зупинки визначають той чи інший варіант $r(\alpha)$-алгоритмів.

1.2 Регулювання кроку. Регулювання величини h_k задає певний спосіб реалізації одновимірного спуску в напрямку нормованого антисубградієнта функції $\varphi(y) = f(B_k y)$, який здійснюється в перетвореному просторі змінних

$y = A_k x$ (див. формулу (4)). В r-алгоритмах використовуються два способи регулювання кроку [2, 4].

При першому способі величина h_k вибирається з умови $h_k = h_k^*$ ($h_k \approx h_k^*$), що означає точний (чи наближений) одновимірний пошук мінімуму функції, який гарантує, що r-алгоритми є монотонними (майже монотонними) по функції, яка мінімізується. Таке регулювання кроку використовуться в теоретично обгрунтованих модифікаціях $r(\alpha)$-алгоритмів, коли коефіцієнт α можна вибирати досить великим. Це дозволило показати, що граничний варіант r-алгоритмів (при $\beta = 0$) є проективним методом спряжених градієнтів, та обгрунтувати для граничного варіанту r-алгоритмів з відновленням квадратичну швидкість збіжності до точки мінімуму опуклої двічі неперервної диференційовної функції при деяких умовах гладкості і регулярності. Знайдено також такі умови для кусково-гладких опуклих функцій, при яких $r_\mu(\alpha)$-алгоритм збігається до точки мінімуму.

Другий спосіб регулювання кроку називається адаптивним і використовується в практичних реалізаціях r-алгоритмів. Він полягає у тому, що величина h_k налаштовується (адаптується) в процесі виконання одновимірного спуску, який завершується, як тільки знайдено субградієнт, що утворює негострий кут з субградієнтом, який визначає напрямок одновимірного спуску (умова завершення спуску за напрямом). Налаштування величини кроку h_k здійснюється за допомогою чотирьох параметрів: $h_0 > 0$ — величина початкового кроку (використовується на першій ітерації, а на кожній наступній – уточнюється); q_1 ($q_1 \le 1$) — коефіцієнт зменшення кроку (якщо умова завершення спуску за напрямком виконується за перший крок); q_2 ($q_2 \ge 1$) – коефіцієнт збільшення кроку. Через кожні n_h кроків одновимірного спуску ($h_h > 1$) величина кроку збільшується в q_2 раз. Оскільки припускається, що $\lim\limits_{\|x\| \to \infty} f(x) = +\infty$, то після скінченної кількості кроків адаптивного спуску в напрямку нормованого

антисубградієнта обов'язково виконується умова завершення спуску за напрямом.

Для адаптивного регулювання кроку коефіцієнт α не рекомендується вибирати великим. Він повинен бути узгодженим з коефіцієнтом q_1, тому що вони обидва, хоча і по різному, впливають на зменшення величини кроку h_k. В результаті визначаються ті два послідовні субградієнти, розтягування за різницею яких зменшує степінь витягнутості функції в перетвореному просторі змінних. Цим пояснюється пришвидшена збіжність $r(\alpha)$-алгоритмів для яружних (з витягнутими поверхнями рівня) функцій. Кількість ітерацій $r(\alpha)$-алгоритму, необхідна для знаходження точки x_{k^*}, для якої $f(x_{k^*}) - f^* \le \varepsilon$, емпірично оцінюється як $k^* = O(n\log(1/\varepsilon))$, де n – кількість змінних.

1.3 Octave-функція ralgb5a [4]. Програма **ralgb5a** є спрощеною (для зручночності використання) версією програми **ralgb5** [2, с. 383–386], в якій використовується метод (1)–(3). Тут абревіатура "**b5**" означає, що в основу програми покладено r-алгоритм у B-формі, де коректується $n \times n$-матриця B, а кожна ітерація методу (1)–(3) вимагає $5n^2$ арифметичних операцій множення, які визначають обчислювальну трудомісткість ітерації (операції додавання враховувати не будемо через їх малий вклад у трудомісткість ітерації). З них $3n^2$ операцій множення потрібно для обчислення векторів $B_k\xi_k$, $B_k^T g_f(x_k)$ і $B_k^T r_k$ (множення матриці на вектор), а $2n^2$ операцій множення вимагає однорангова корекція матриці $B_{k+1} = B_k R_\beta(\eta_k)$. Дійсно, корекція матриці B_{k+1} виконується за формулою

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

звідки легко бачити, що обчислення вектора $\eta = B_k\eta_k$ вимагає n^2 операцій множення, і стільки ж операцій множення вимагає побудова тимчасової однорангової матриці $\eta\eta_k^T$.

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

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

2. Опис програми ralgb5a

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

$$f(x)=\sum_{i=1}^{100}(1.2)^{i-1}|x_i-1|,\quad f^*=f(x^*)=0,\quad x^*=(1,1,\ldots,1)^T, \tag{5}$$

де $|a|$ – абсолютна величина числа a. Функція (4) є яружною, так як коефіцієнти при $|x_i-1|$, $i=1,\ldots,100$ утворюють геометричну прогресію з показником $q=1.2$, де мінімальний коефіцієнт дорівнює $(1.2)^0=1$, а максимальний – $(1.2)^{99}\approx 6.9015\text{e+}07$.

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

```
# Input parameters:
#    calcfg - name of the function calcfg(x) for calculation of f and g
#    x - the starting point, x0(1:n) (it is modified in the program)
#    alpha - the value of coefficient of space dilation
#    h0, q1 - parameters of the adaptive step adjustment
#    epsx, epsg, maxitn - stop parameters
#    intp - print information every intp iteration
# Output parameters:
#    xr - a minimum point, which was found by the program, xr(1:n)
#    fr - the value of the function f at the point xr
#    itn - the number of iterations used by the program
#    nfg - the number of function calcfg calls
#    istop - exit code (2 = epsg, 3 = epsx, 4 = maxitn, 5 = error)
function [xr,fr,itn,nfg,istop] = ralgb5a(calcfg,x,alpha,h0,q1,  #row001
                                         epsg,epsx,maxitn,intp);
itn = 0; B = eye(length(x)); hs = h0; lsa = 0; lsm = 0;         #row002
xr = x; [fr,g0] = calcfg(xr); nfg = 1;                          #row003
printf("itn %4d  f%15.6e  fr%15.6e  nfg %4d\n",itn,fr,fr,nfg);  #row004
if(norm(g0) < epsg) istop = 2;  return; endif                   #row005
for (itn = 1:maxitn)                                            #row006
```

```
   dx = B * (g1 = B' * g0)/norm(g1);                                  #row007
   d = 1; ls = 0; ddx = 0;                                            #row008
   while (d > 0)                                                      #row009
      x -= hs * dx; ddx += hs * norm(dx);                             #row010
      [f, g1] = calcfg(x); nfg ++;                                    #row011
      if (f < fr)  fr = f; xr = x;  endif                             #row012
      if(norm(g1) < epsg) istop = 2;  return; endif                   #row013
      ls ++; (mod(ls,3) == 0) && (hs *= 1.1);                         #row014
      if(ls > 500) istop = 5;  return; endif                          #row015
      d = dx' * g1;                                                   #row016
   endwhile                                                           #row017
   (ls == 1) && (hs *= q1);  lsa=lsa+ls; lsm=max(lsm,ls);             #row018
   if(mod(itn,intp)==0)                                               #row019
     printf("itn %4d  f %14.6e  fr %14.6e", itn, f, fr);              #row020
     printf("  nfg %4d  lsa %3d  lsm %3d\n", nfg, lsa, lsm);          #row021
     lsa=0; lsm=0;                                                    #row022
   endif                                                              #row023
   if(ddx < epsx) istop = 3;  return; endif                           #row024
   xi = (dg = B' * (g1 - g0) )/norm(dg);                              #row025
   B += (1 / alpha - 1) * B * xi * xi';                               #row026
   g0 = g1;                                                           #row027
endfor                                                                #row028
istop = 4;                                                            #row029
endfunction                                                           #row030
```

При мінімізації негладких функцій рекомендується вибирати: alpha=$2 \div 3$, h0=1.0, q1=1.0. При мінімізації гладких функцій рекомендується використовувати q1=$0.8 \div 0.95$. При правильному підборі цих параметрів можна значно скоротити кількість ітерацій для виконання одних і тих же критеріїв зупинки. Це залежить від конкретного виду функції, що мінімізується, ступеня її яружності і масштабу змінних.

2.2 Тестовий приклад. Для тестового прикладу використовується кусково-лінійна функція (5). Обчислення значення функції (5) та її субградієнта реалізовано Octave-функцією

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

для якої значення коефіцієнтів w встановлюються за допомогою операторів

```
global w
n=100; temp=[0:(n-1)]'; w=1.2.**temp;
```

Ітераційний процес запускається зі стартової точки $x_0 = (0,0,\ldots,0)^T$, для якої значення функції $f(x_0) = 4.140899\text{e+}08$. Параметри $r(\alpha)$-алгоритма

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

Для вказаних параметрів головна Octave-програма має такий вигляд:

```
global w
n=100; temp=[0:(n-1)]'; w=1.2.**temp; # w(1,1), w(100,1),
x = zeros(n,1); alpha = 4.0, h0 = 10.0, q1 = 1.0,
epsx = 1.e-8, epsg = 1.e-12, maxitn = 5000, intp=500;
[xr,fr,itn,nfg,istop] = ralgb5a(@sabs,x,alpha,h0,q1,epsg,epsx,maxitn,intp);
printf("itn %4d  fr %23.15e  istop %d  nfg %4d\n", itn, fr, istop,nfg);
dx = norm(xr-ones(n,1)),
```

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

```
alpha =  4  h0 =  10  q1 =  1
epsx = 1.0000e-008  epsg = 1.0000e-012  maxitn =  5000
itn    0  f  4.140899e+008  fr  4.140899e+008  nfg    1
itn  500  f  1.718525e+003  fr  1.273433e+003  nfg  532  lsa 531  lsm   4
itn 1000  f  1.409472e+000  fr  1.192802e+000  nfg 1032  lsa 500  lsm   1
itn 1500  f  1.258921e-003  fr  1.258921e-003  nfg 1532  lsa 500  lsm   1
itn 2000  f  1.422859e-006  fr  1.224438e-006  nfg 2032  lsa 500  lsm   1
itn 2046  fr  6.340398755873688e-007  istop 3  nfg 2078
dx = 1.9497e-008
```

Тут вхідні параметри $r(\alpha)$-алгоритма зібрано в два перших рядки. Із протоколу видно, що кількість ітерацій відповідає емпіричній оцінці, тобто на одну ітерацію в середньому затрачено $2078/2046 < 3$ обчислень значення функції (5) та її субградієнта. Якщо вибрати параметр $q_1 = 0.95$, то кількість ітерацій зменшується до **920** при **1539** викликах функції **sabs**.

3. Вимоги до програми

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

ЛІТЕРАТУРА

1. *Шор Н.З.* Методы минимизации недифференцируемых функций и их приложения. – Киев: Наукова думка, 1979. – 200 с.

2. *Стецюк П.И.* Методы эллипсоидов и *r*-алгоритмы. – Эврика: Кишинэу, 2014. – 488 с.

3. *Стецюк П.И.* Теория и программные реализации *r*-алгоритмов Шора // Кибернетика и системный анализ. – 2017. – № 5. – С. 43–57.

4. *Стецюк П.И., Фишер А. r*–Алгоритмы Шора и octave-функция ralgb5a // Тези міжнародної наукової конференції «Сучасна інформатика: проблеми, досягнення та перспективи розвитку», що присвячена 60-річчю заснування Інституту кібернетики імені В.М.Глушкова НАН України (м. Київ, 13-15 грудня 2017 р.). – К.: Ін-т кібернетики ім. В.М.Глушкова НАН України, 2017. – С. 143–146.

5. Octave [Электронный ресурс]: http://www.octave.org – Режим доступа: свободный.