Субградиентные методы с преобразованием пространства для минимизации овражных выпуклых функций

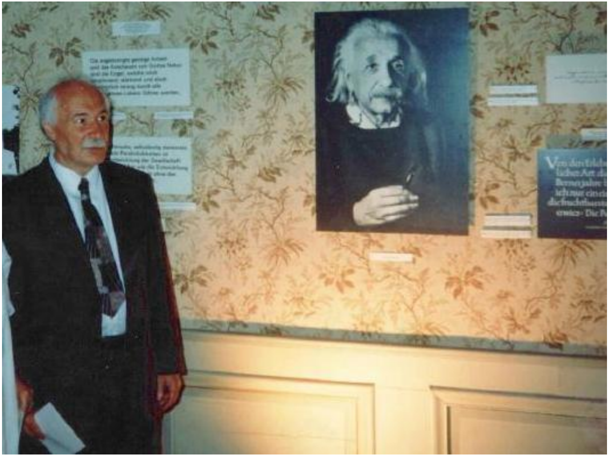
Стецюк П.И.
Институт кибернетики им. В.М. Глушкова НАН Украины
e-mail: stetsyukp@gmail.com

Международная конференция «Современные проблемы прикладной математики и механики: теория, эксперимент и практика»,
посвященная 90-летию со дня рождения академика Н.Н. Яненко
Новосибирск, Россия, 30 мая – 4 июня 2011 г.

План

1. Об одной истории с фотографией
   - О внешнем сходстве Шора и Ейнштейна
   - О сходстве идей ...
2. Субградиентные методы с преобразованием пространства
   - r-алгоритмы и octave-функция ralgb5
   - Метод amsg2p и его алгоритм-функция

Внешнее сходство Н.З. Шора с А. Эйнштейном

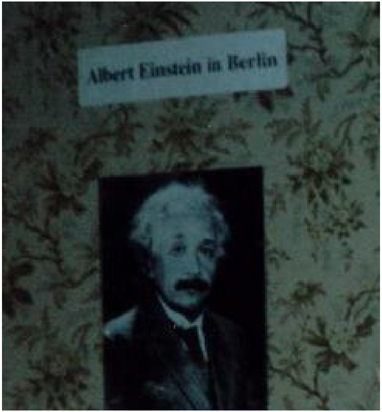


подметил Ж.-Ф. Эмменеггер (университет Фрибурга, Швейцария). Он был инициатором фотографии, фрагмент из которой приведен на слайде.

Фотография сделана в музей-квартире А. Ейнштейна, Берн, 1997 год.

Эту фотографию Н.З. положил на мой рабочий стол в 1997 году. При этом он улыбнулся и многозначительно промолчал. Я смог произнести только „А вот и еще одно доказательство ...“, вспоминая наш разговор 1995 года. Он еще раз улыбнулся и снова многозначительно промолчал.

И еще о внешней схожести ... но уже помоложе ...



Альберт Ейнштейн в Берлине.

Наум Шор в Киеве.

О сходстве идей ... или о разговоре 1995 года.

Что общего?

между

формулой Ейнштейна для энергии в релятивистской динамике

$$E = \frac{m_0}{\sqrt{1 - \frac{v^2}{c^2}}} c^2, \qquad \text{где} \qquad m = \frac{m_0}{\sqrt{1 - \frac{v^2}{c^2}}}$$

и

оператором Шора для растяжения пространства

$$R_\alpha(\xi) = I + (\alpha^2 - 1)\xi\xi^T, \qquad \text{где} \qquad \alpha > 1.$$

Чуть подробнее об операции растяжения пространства ...

Операция растяжения пространства переменных реализуется с помощью оператора растяжения пространства, который в матрично-векторной форме представим:

$$R_\alpha(\xi) = I_n + (\alpha - 1)\xi\xi^T, \quad \xi \in E^n, \quad \|\xi\| = 1, \quad \alpha > 1,$$

где $(\cdot)^T$ означает транспонирование, I_n – единичная матрица порядка n, α – коэффициент растяжения пространства, ξ – направление растяжения.

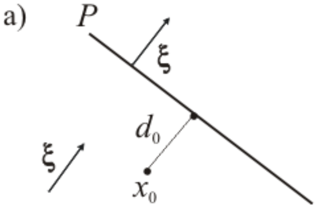
При описании алгоритмов с растяжением пространства используется оператор $R_\beta(\xi)$, обратный к оператору растяжения пространства $R_\alpha(\xi)$. Он имеет следующий вид:

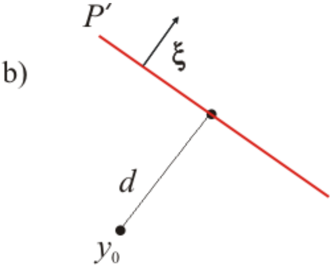
$$R_\beta(\xi) = R_\alpha^{-1}(\xi) = I_n + (\beta - 1)\xi\xi^T, \quad \beta = \frac{1}{\alpha} < 1$$

и в методах с растяжением пространства переменных используется для „сжатия“ пространства субградиентов.

О расстояниях в двух пространствах (простейший случай), $\alpha > 1$

Пусть $P = \{x \in E^n : (x - x_0, \xi) = d_0\}$ – гиперплоскость в исходном пространстве $X = E^n$ (рис. a). В преобразованном пространстве $Y = R_\alpha(\xi)X$ ей соответствует гиперплоскость $P' = \{y \in E^n : (y - y_0, \xi) = d\}$ (рис. b).



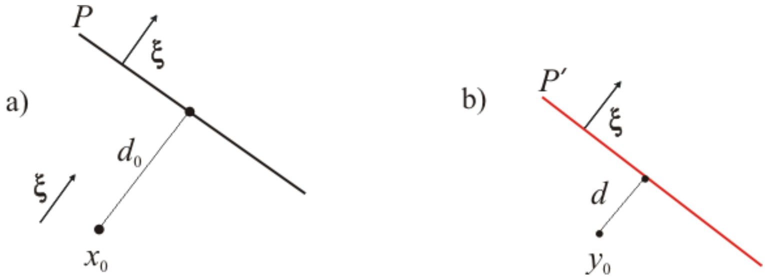


d_0 – расстояние до P и d – расстояние до P' связаны соотношением

$$d = d_0 \times \alpha = \frac{d_0}{\beta} = \frac{d_0}{\sqrt{1-(1-\beta^2)}} = \quad \text{почти} \quad = \frac{d_0}{\sqrt{1-v^2/c^2}}$$

Тот же простейший случай, но коэффициент растяжения $\alpha < 1$

Пусть $P = \{x \in E^n : (x - x_0, \xi) = d_0\}$ – гиперплоскость в исходном пространстве $X = E^n$ (рис. a). В преобразованном пространстве $Y = R_\alpha(\xi)X$ ей соответствует гиперплоскость $P' = \{y \in E^n : (y - y_0, \xi) = d\}$ (рис. b).



d_0 – расстояние до P и d – расстояние до P' связаны соотношением

$$d = d_0 \times \alpha = d_0\sqrt{1 - (1 - \alpha^2)} = \quad \text{почти} \quad = d_0\sqrt{1 - v^2/c^2}$$

О расстояниях до гиперплоскостей в общем случае

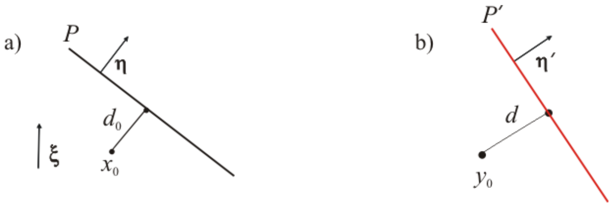
Пусть $P = \{x \in E^n : (x - x_0, \eta) = d_0\}$ – гиперплоскость в исходном пространстве $X = E^n$ (рис. a). В преобразованном пространстве $Y = R_\alpha(\xi)X$ ей соответствует гиперплоскость $P' = \{y \in E^n : (y - y_0, \eta\prime) = d\}$ (рис. b).

d_0 – расстояние до P и d – расстояние до P' связаны соотношением

$$d = \frac{d_0}{\|R_\beta(\xi)\eta\|} = \frac{d_0}{\sqrt{1 + (\beta^2 - 1)(\xi, \eta)^2}} \qquad \text{и} \qquad \eta\prime = \frac{R_\beta(\xi)\eta}{\|R_\beta(\xi)\eta\|}$$

Что это дает для субградиентных процессов?

Нормали к гиперплоскостям P' и P связаны формулой $\eta\prime = \dfrac{R_\beta(\xi)\eta}{\|R_\beta(\xi)\eta\|}$



а значит, выбрав подходящие направления растяжения и соответствующие им коэффициенты, можно так преобразовать текущее пространство переменных, чтобы в преобразованном пространстве переменных улучшались те или иные свойства субградиентного процесса.
!!! очень похоже на использование Ейнштейном преобразований Лоренца !!!

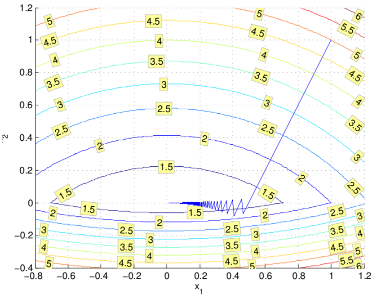
На этом принципе основаны r-алгоритмы и метод amsg2p, которые имеют ускоренную сходимость для овражных и существенно овражных выпуклых функций. (see http://conf.nsc.ru/niknik-90/reportview/37828).

Пример ускоренной сходимости для существенно овражной функции.

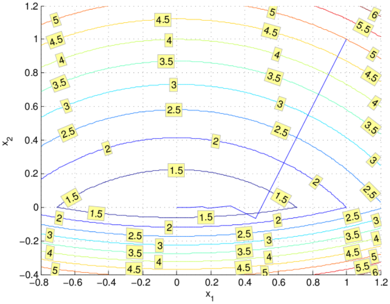
Существенно овражная кусочно-квадратичная функция

$$f_2(x_1, x_2) = \max\left\{x_1^2 + (2x_2 - 2)^2 - 3, x_1^2 + (x_2 + 1)^2\right\}$$

вырождена в точке минимума $x^\star = (0, 0)$, $f^\star = 1$.



Метод Поляка (10000 итераций)



Метод amsg2p (30 итераций)

Об r-алгоритмах Н.З. Шора

r-Алгоритмы – семейство субградиентных методов с растяжением пространства, предназначенных для нахождения f_r^* и x_r^* в задаче

$$\text{найти}\quad f_r^* \approx f^* = \min_{x\in R^n} f(x) \quad \text{и} \quad x_r^* \approx x^* = \operatorname*{argmin}_{x\in R^n} f(x), \tag{1}$$

где $f(x)$ – выпуклая функция (как негладкая, так и гладкая).

Вычислительная схема r-алгоритмов такова

$$x_{k+1} = x_k - h_k B_k \frac{B_k^T \partial f(x_k)}{\|B_k^T \partial f(x_k)\|}, \quad h_k \geq h_k^*, \tag{2}$$

$$h_k^* = \operatorname*{argmin}_{h\geq 0} f\left(x_k - h_k B_k \frac{B_k^T \partial f(x_k)}{\|B_k^T \partial f(x_k)\|}\right)$$

$$r_k = \partial f(x_{k+1}) - \partial f(x_k), \quad \xi_k = \frac{B_k^T r_k}{\|B_k^T r_k\|}, \tag{3}$$

$$B_{k+1} = B_k R_{\beta_k}(\xi_k), \qquad k = 0, 1, 2, \dots, \tag{4}$$

Здесь $\partial f(x)$ – субградиент (градиент) функции $f(x)$

Об r-алгоритмах Н.З. Шора (продолжение)

r-Алгоритмы базируются на процедуре точного (приближенного) поиска минимума функции по направлению антисубградиента в преобразованном пространстве переменных и обеспечивают монотонность (или почти монотонность) по значениям минимизируемой функции.

r-Алгоритмы используют операцию растяжения пространства в направлении разности двух последовательных субградиентов (второй субградиент вычислен в точке минимума функции в направлении первого антисубградиента).

$r(\alpha)$-Алгоритм – вариант r-алгоритмов с постоянным на каждой итерации коэффициентом растяжения пространства α ($\alpha>1$) и адаптивной регулировкой шага в направлении нормированного антисубградиента в преобразованном пространстве.

О параметрах $r(\alpha)$-алгоритма

Адаптивная регулировка шага реализует спуск в направлении нормированного антисубградиента в преобразованном пространстве переменных и делает это с помощью параметров h_0, q_1, n_h, q_2. Здесь h_0 — величина начального шага (используется на 1-й итерации, на каждой последующей итерации эта величина уточняется); q_1 — коэффициент уменьшения шага ($q_1 \leq 1$), если условие завершения спуска по направлению выполняется за один шаг; q_2 — коэффициент увеличения шага ($q_2 \geq 1$); натуральное число n_h задает число шагов одномерного спуска ($n_h > 1$), через каждые из которых шаг будет увеличиваться в q_2 раз.

Параметры ε_x и ε_g определяют условия завершения $r(\alpha)$-алгоритма: метод останавливается в точке x_{k+1}, если выполнено $\|x_{k+1} - x_k\| \leq \varepsilon_x$ (останов по аргументу); метод останавливается в точке x_{k+1}, если выполнено условие $\|g_f(x_{k+1})\| \leq \varepsilon_g$ (останов по норме субградиента, используется для гладких функций). Аварийное завершение метода связано либо с тем, что функция $f(x)$ неограничена снизу, либо h_0 слишком мал и его требуется увеличить.

Что делает octave-функция ralgb5?

Octave-функция ralgb5 реализует $r(\alpha)$-алгоритм. Она использует подготовленную пользователем octave-функцию

function [f,g] = calcfg(x),

которая вычисляет значение функции $f = f(x)$ и её субградиента $g = \partial f(x)$ в точке x.

Вход и выход octave-функции ralgb5

Octave-функция ralgb5 использует следующие параметры

```
%  Входные параметры:
%    calcfg -- имя функции вида calcfg(x) для вычисления f и g
%    x  -- начальная точка x(n) (на выходе портится)
%    alpha -- коэффициент растяжения пространства
%    h0, nh, q1, q2 -- параметры адаптивной регулировки шага
%    epsx, epsg, maxitn -- параметры останова
%  Выходные параметры:
%    xr -- найденная точка минимума функции xr(n)
%    fr -- значение функции в точке минимума
%    itn -- число затраченных итераций
%    ncalls -- число вызовов функции calcfg
%    istop -- код останова (2 = epsg, 3 = epsx, 4 = maxitn, 5 = error)
```

Код octave-функции ralgb5

```
# ralgb5 -- Octave-function for Shor's r-algorithm
function [xr,fr,itn,ncalls,istop]=ralgb5(calcfg,x,alpha,h0,q1,
                                          q2,nh,epsg,epsx,maxitn);
itn=0; hs=h0; B=eye(length(x)); xr=x;                         # row001
ncalls = 1; [fr,g0] = calcfg(xr);                             # row002
printf("itn %4d f %14.6e fr %14.6e ls %2d ncalls %4d\n",      # row003
       itn, fr, fr, 0, ncalls);
if(norm(g0) < epsg) istop = 2;  return; endif                 # row004
for (itn = 1:maxitn)                                          # row005
   dx = B * (g1 = B' * g0)/norm(g1);                          # row006
   d = 1; ls = 0; ddx = 0;                                    # row007
   while (d > 0)                                              # row008
      x -= hs * dx; ddx += hs * norm(dx);                     # row009
      ncalls ++; [f, g1] = calcfg(x);                         # row010
      if (f < fr)  fr = f; xr = x; endif                      # row011
      if(norm(g1) < epsg) istop = 2;  return; endif           # row012
      ls ++; (mod(ls,nh)==0) && (hs *= q2);                   # row013
      if(ls > 500) istop = 5;  return; endif                  # row014
      d = dx' * g1;                                           # row015
   endwhile                                                   # row016
   (ls == 1) && (hs *= q1);                                   # row017
   printf("itn %4d f %14.6e fr %14.6e ls %2d ncalls %4d\n",   # row018
          itn, f, fr, ls, ncalls);
   if(ddx < epsx) istop = 3;  return; endif                   # row019
   xi = (dg = B' * (g1 - g0) )/norm(dg);                      # row020
   B += (1 / alpha - 1) * B * xi * xi';                       # row021
   g0 = g1;                                                   # row022
endfor                                                        # row023
istop = 4;                                                    # row024
endfunction
```

Octave-функция ralgb5 ... rows(001-004,024)

```
# ralgb5 -- Octave-function for Shor's r-algorithm
function [xr,fr,itn,ncalls,istop]=ralgb5(calcfg,x,alpha,h0,q1,
                                          q2,nh,epsg,epsx,maxitn);
itn=0; hs=h0; B=eye(length(x)); xr=x;                      # row001
ncalls = 1; [fr,g0] = calcfg(xr);                           # row002
printf("itn %4d f %14.6e fr %14.6e ls %2d ncalls %4d\n",   # row003
       itn, fr, fr, 0, ncalls);
if(norm(g0) < epsg) istop = 2;  return; endif               # row004
for (itn = 1:maxitn)                                        # row005

   ... ... row006 -- row022 ... ...                          ........

endfor                                                      # row023
istop = 4;                                                  # row024
endfunction
```

Octave-функция ralgb5 ... rows(005-023)

```
for (itn = 1:maxitn)                                        # row005
   dx = B * (g1 = B' * g0)/norm(g1);                        # row006
   d = 1; ls = 0; ddx = 0;                                  # row007
   while (d > 0)                                            # row008
      x -= hs * dx; ddx += hs * norm(dx);                   # row009
      ncalls ++; [f, g1] = calcfg(x);                       # row010
      if (f < fr)  fr = f; xr = x; endif                    # row011
      if(norm(g1) < epsg) istop = 2;  return; endif         # row012
      ls ++; (mod(ls,nh)==0) && (hs *= q2);                 # row013
      if(ls > 500) istop = 5;  return; endif                # row014
      d = dx' * g1;                                         # row015
   endwhile                                                 # row016
   (ls == 1) && (hs *= q1);                                 # row017
   printf("itn %4d f %14.6e fr %14.6e ls %2d ncalls %4d\n", # row018
           itn, f, fr, ls, ncalls);
   if(ddx < epsx) istop = 3;  return; endif                 # row019
   xi = (dg = B' * (g1 - g0) )/norm(dg);                    # row020
   B += (1 / alpha - 1) * B * xi * xi';                     # row021
   g0 = g1;                                                 # row022
endfor                                                      # row023
```

Выбор параметров для octave-функции ralgb5

При минимизации негладких функций рекомендуется следующий выбор параметров: $\alpha = 2 \div 3$, $h_0 = 1.0$, $q_1 = 1.0$, $q_2 = 1.1 \div 1.2$, $n_h = 2 \div 3$. Если известна априорная оценка расстояния от начальной точки x_0 до точки минимума x^*, то начальный шаг h_0 целесообразно выбирать порядка $\|x_0 - x^*\|$. При минимизации гладких функций рекомендуемые параметры такие же, за исключением q_1 $(q_1 = 0.8 \div 0.95)$.

При таком выборе параметров, как правило, число спусков по направлению редко превосходит два, а за n шагов точность по функции улучшается в три-пять раз. Параметры останова $\varepsilon_x, \varepsilon_g \sim 10^{-6} \div 10^{-5}$ при минимизации выпуклой функции даже существенно овражной структуры обеспечивает нахождение x_r^* со значением функции, достаточно близким к оптимальному. При этом обычно

$$\frac{f(x_r^*) - f(x^*)}{|f(x^*)| + 1} \sim 10^{-6} \div 10^{-5} \text{ — для негладких}$$

и

$$\frac{f(x_r^*) - f(x^*)}{|f(x^*)| + 1} \sim 10^{-12} \div 10^{-10} \text{ — для гладких функций,}$$

что подтверждается результатами многочисленных тестовых и реальных расчетов.

Постановка задачи

Метод amsg2p предназначен для решения задачи

$$\text{найти} \quad x^\star = \operatorname*{argmin}_{x \in R^n} f(x), \quad \text{где } f(x) \text{ – выпуклая функция} \tag{5}$$

при условиях:

1) $f^\star$ – известно, $f^\star = f(x^\star)$, $x^\star \in X^\star$;
2) $\forall x \in R^n$ и $\forall x^\star \in X^\star$ выполняется неравенство

$$(x - x^\star, \partial f(x)) \geq \gamma(f(x) - f^\star), \quad \text{где } \gamma \geq 1. \tag{6}$$

Здесь $\partial f(x)$ – субградиент (градиент) функции $f(x)$

Метод amsg2p

есть субградиентным методом с преобразованием пространства

$$x_{k+1} = x_k - h_k B_k \frac{B_k^T \partial f(x_k)}{\|B_k^T \partial f(x_k)\|}, \quad h_k = \frac{\gamma(f(x_k) - f^*)}{\|B_k^T \partial f(x_k)\|}, \tag{7}$$

$$B_{k+1} = B_k T^{-1}(\xi, \eta) \quad \text{или} \quad B_{k+1} = B_k, \quad k = 0, 1, 2, \dots, \tag{8}$$

„ams“ указывает на способ регулировки шага. Впервые AMS-шаг использовали S. Agmon и T. S. Motzkin, I. J. Schoenberg в 1954 году в релаксационном методе для нахождения хотя бы одного из решений совместной системы линейных неравенств.

„g2p“ указывает на использование AMS-шага в пространстве переменных, преобразованном с помощью двух последних субградиентов (g2) и агрегатного вектора (p)

Одноранговый эллипсоидальный оператор

Линейный оператор из R^n в R^n

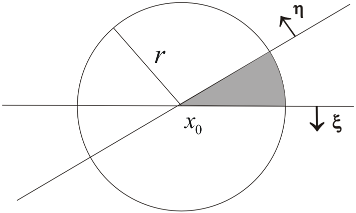
$$T_1(\xi,\eta) = I - \frac{1}{1-(\xi,\eta)^2}\left(\left(1-\sqrt{1-(\xi,\eta)^2}\right)\eta-(\xi,\eta)\xi\right)\eta^T. \qquad (9)$$

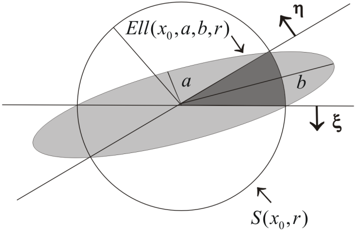
Здесь $\xi,\eta \in R^n$ – векторы, такие что $\|\xi\| = 1, \|\eta\| = 1$ и $(\xi,\eta)^2 \neq 1$, I – единичная матрица размера $n \times n$.

Стецюк П.И. Ортогонализующие линейные операторы в выпуклом программировании (Часть I) // Кибернетика и системный анализ. – 1997. – N3. – С.97–119.

Преобразует в шар специальный эллипсоид, описанный вокруг тела W, которое получено в результате пересечения шара и двух полупространств, проходящих через центр шара.

Тело W и Специальный эллипсоид

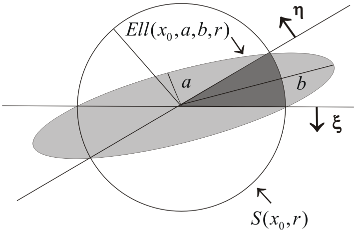




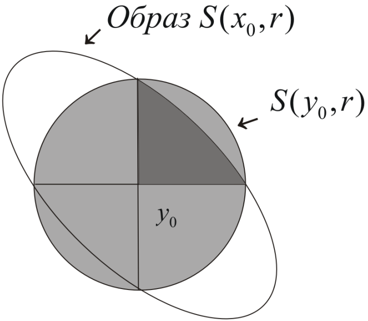
Тело W получено как пересечение шара и двух полупространств.

Специальный эллипсоид содержит W и имеет минимальный обьем.

Специальный эллипсоид до и после преобразования



Специальный эллипсоид



в преобразованном пространстве становится шаром

Алгоритм-функция для метода amsg2p

Алгоритм-функция amsg2p: $(x_\varepsilon^*, k_\varepsilon^*)$ =amsg2p$(x_0, \varepsilon, f^*, \gamma)$

На итерации $k{=}0$ имеем начальное приближение $x_0 \in R^n$ и достаточно малое $\varepsilon{>}0$. Вычислим $f(x_0)$ и $\partial f(x_0)$. Если $f(x_0){-}f^*{\le}\varepsilon$, то $x_\varepsilon^*{=}x_0$, $k_\varepsilon^*{=}0$ и окончание работы алгоритма.

Иначе положим $h_0{=}\dfrac{\gamma(f(x_0){-}f^*)}{\|\partial f(x_0)\|}$, $\xi_0{=}\dfrac{\partial f(x_0)}{\|\partial f(x_0)\|}{\in}R^n$, $p_0{=}0{\in}R^n$, $B_0{=}I_n$ – единичная матрица размера $n{\times}n$.

Перейдем к следующей итерации.

Пусть на k-й итерации получены $x_k \in R^n$, h_k, $\xi_k \in R^n$, $p_k \in R^n$, B_k – матрица $n{\times}n$. Для $(k+1)$-й итерации выполним пп. 1–5.

Алгоритм-функция amsg2p: п. 1-3

1. Вычислим очередное приближение

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

2. Вычислим $f(x_{k+1})$ и $\partial f(x_{k+1})$. Если $f(x_{k+1}) - f^* \le \varepsilon$, то $x_\varepsilon^* = x_{k+1}$, $k_\varepsilon^* = k+1$ и окончание алгоритма. Иначе положим

$$\xi_{k+1} = \frac{B_k^T \partial f(x_{k+1})}{\|B_k^T \partial f(x_{k+1})\|}, \quad h_{k+1} = \frac{\gamma(f(x_{k+1}) - f^*)}{\|B_k^T \partial f(x_{k+1})\|}.$$

3. Вычислим $\lambda_1 = -p_k^T \xi_{k+1}$ и $\lambda_2 = -\xi_k^T \xi_{k+1}$. Положим

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

Алгоритм-функция amsg2p: п. 4-5

4. Вычислим $\mu_k = p_{k+1}^T \xi_{k+1}$. Если $-1 < \mu_k < 0$, то вычислим

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

и пересчитаем

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

Иначе положим $B_{k+1} = B_k$ и $p_{k+1} = 0$.

5. Перейдем к новой итерации с x_{k+1}, h_{k+1}, ξ_{k+1}, p_{k+1}, B_{k+1}.

Выводы

Приведенные алгоритмы можно использовать при решении негладких задач из различных областей приложений. Так как гладкая функция с очень быстро изменяющимся градиентом близка по своим свойствам к негладкой функции, то наши алгоритмы обладают ускоренной сходимостью при оптимизации овражных гладких функций. Матрично-векторные вычисления для обоих семейств алгоритмов легко поддаются параллельной обработке, что может быть полезным при их реализации на параллельных ЭВМ.

Библиография

1. Шор Н.З. Методы недифференцируемой оптимизации и сложные экстремальные задачи: Сб. избр. тр. – Кишинэу: Эврика, 2008. – 270 с.

2. Шор Н.З. Методы минимизации негладких функций и матричные задачи оптимизации: Сб. избр. тр. – Кишинэу: Эврика, 2009. – 240 с.

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

Сборники избранных трудов Шора доступны по ссылке
http://elis.dvo.ru/?q=node/114

Вопросы?

СПАСИБО ЗА ВНИМАНИЕ!