Инструкция по использованию Фортран-программы ralgb4*

П. И. Стецюк

Аннотация

Инструкция включает описание, правила работы и Фортрановский код программы ralgb4 (реализует один из вариантов $r(\alpha)$-алгоритма в B-форме, т.е. корректируется матрица обратного преобразования B). В основу программы ralgb4 положена самая экономная по числу арифметических операций вычислительная схема $r(\alpha)$-алгоритма в B-форме, которая требует $4n^2$ арифметических операций на каждой итерации. В инструкции эта схема подробно описана согласно работы [7]. Единственное отличие состоит в том, что адаптивная регулировка шага в направлении нормированного антисубградиента в преобразованном пространстве переменных слегка нарушена по отношению к общепринятой. Если в общепринятой регулировке шаговый множитель увеличивается через каждые nh шагов одномерного спуска, то в приведенной алгоритмической схеме шаговый множитель увеличивается на каждом следующем (после nh шагов) шаге одномерного спуска по направлению. Дан протокол работы программы для квадратичной и кусочно-линейной выпуклых функций от 10 переменных и овражной структуры поверхностей уровня.

Содержание

*Работа выполнена в рамках совместного украинско-российского проекта ДФФД Украины – Ф28.1/005 и РФФИ – 09-01-90413-Укр.

1 Описание $r(\alpha)$-алгоритма

Так как на основе методов субградиентного типа с растяжением пространства в направлении разности двух последовательных субградиентов (так называемые r-алгоритмы [1, 2]) можно построить семейство алгоритмов, используя различные способы регулировки шагового множителя и коэффициента растяжения пространства, нужно уточнить, какой именно модификацией алгоритма мы пользуемся при выполнении численных экспериментов. Это так называемый $r(\alpha)$-алгоритм с постоянным коэффициентом растяжения пространства α и адаптивным способом регулировки шагового множителя (см. [3, 6]), разработанный в Институте кибернетики и хорошо зарекомендовавший себя в практических расчетах.

При описании вычислительной схемы $r(\alpha)$-алгоритма применительно к решению задачи минимизации выпуклой функции $f(x)$, определенной на n-мерном евклидовом пространстве E^n со скалярным произведением $(\cdot,\cdot)$ и обладающей свойством

$$\lim_{|x|\to+\infty} f(x) \to +\infty,$$

будем использовать оператор $R_\alpha(\xi)$ растяжения (сжатия) пространства с коэффициентом $\alpha \geq 1$ $(0 < \alpha < 1)$ в направлении $\xi \in E^n$, $\|\xi\| = 1$. В матричной форме он представим следующей формулой:

$$R_\alpha(\xi) = I_n + (\alpha - 1)\xi\xi^T,$$

где I_n – единичная матрица размерности $n \times n$. Субградиент функции $f(x)$ в точке x_k будем обозначать $g_f(x_k)$, а полученную с помощью $r(\alpha)$-алгоритма приближенную точку минимума $f(x)$ при выполнении условия останова обозначим x_r^*.

При комментировании основных операций в вычислительной схеме $r(\alpha)$-алгоритма (комментарии выделены скобками) будем использовать следующие обозначения: X – исходное пространство переменных (аргументов) функции $f(x)$; A – неособенная матрица размерности $n \times n$, задающая невырожденный линейный оператор A, который действует из E^n в E^n; B – матрица обратного преобразования пространства ($B{=}A^{-1}$); Y – преобразованное линейным оператором A пространство переменных (аргументов) функции $\varphi(y) = f(By)$; $g_\varphi(y_k)$ – субградиент функции $\varphi(y)$ в точке y_k, который вычисляется по формуле $g_\varphi(y) = B^T g_f(x)$.

$r(\alpha)$-алгоритм использует следующие параметры: коэффициент растяжения пространства α $(\alpha > 1)$; параметры адаптивной регулировки шага в направлении нормированного субградиента в преобразованном пространстве переменных h_0, q_1, n_h, q_2 (h_0 – величина начального шага, q_1 – коэффициент уменьшения шага $(q_1 < 1)$, если условие спуска по направлению выполняется за один шаг, q_2 – коэффициент увеличения шага $(q_2 > 1)$, натуральное число n_h задает число шагов одномерного спуска, после которых шаг будет увеличиваться в q_2 раз $(n_h > 1)$); параметры ε_x и ε_g, задающие условия останова алгоритма (ε_x задает останов по отклонению аргумента, т.е. останов метода в точке x_{k+1} происходит, если выполнено условие

$\|x_{k+1} - x_k\| \le \varepsilon_x$; ε_g задает останов по норме субградиента, т.е. останов метода в точке x_{k+1} происходит, если выполнено условие $\|g_f(x_{k+1})\| \le \varepsilon_g$).

Вычислительная схема $r(\alpha)$-алгоритма принимает следующий вид.

В начале процесса имеем начальную точку $x_0 \in E^n$, параметры α, h_0, q_1, n_h, q_2, ε_x, ε_g. Вычислим $f(x_0)$ и $g_f(x_0)$. Если $\|g_f(x_0)\| \le \varepsilon_g$, то $x_r^* = x_0$ и останов. В противном случае полагаем $B_0 = I_n$ (матрица размерности $n \times n$), $\tilde{g}_0 = g_f(x_0)$, $g_0 = g_f(x_0)$.

Пусть в результате вычислений на k-й итерации процесса получены величины x_k, h_k, $\tilde{g}_k$, $g_k = g_f(x_k)$ и матрица B_k размерности $n \times n$. (Здесь h_k – текущее значение шагового множителя в направлении нормированного субградиента в преобразованном пространстве переменных $Y_k = A_k X = B_k^{-1} X$. Вектор $\tilde{g}_k$ используется для обозначения субградиента на предыдущей итерации метода, т.е. субградиента функции $\varphi_k(y) = f(B_k y)$ в точке $y_{k-1} = B_k^{-1} x_{k-1}$. Исключением является итерация при $k = 0$, где $\tilde{g}_0$ совпадает с субградиентом $f(x)$ в точке x_0. Это сделано для удобства при программировании метода (см. п. 1.3.)). Тогда переход к (k+1)-й итерации процесса состоит из следующих операций.

1. Подготовка к операции растяжения пространства

Вычисляем следующие величины

1.1. $g_k^* = B_k^T g_k$ (соответствует $g_{\varphi_k}(y_k)$, где $y_k = A_k x_k$, т.е. субградиенту функции $\varphi_k(y) = f(B_k y)$ в точке $y_k = A_k x_k = B_k^{-1} x_k$ преобразованного пространства аргументов $Y_k = A_k X = B_k^{-1} X$).

1.2. $r_k = g_k^* - \tilde{g}_k$ (соответствует вектору разности двух последовательных субградиентов в преобразованном пространстве аргументов $Y_k = A_k X$, т.е. $r_k = g_{\varphi_k}(y_k) - g_{\varphi_k}(y_{k-1}) = B_k^T (g_f(x_k) - g_f(x_{k-1}))$.

1.3. $\xi_k = \begin{cases} 0, & \text{если } \|r_k\| \le \varepsilon_0, \\ \frac{r_k}{\|r_k\|}, & \text{если } \|r_k\| > \varepsilon_0. \end{cases}$ (нормировка вектора r_k для выполнения очередной операции растяжения пространства в направлении разности двух последовательных субградиентов).

В п. 1.3 ε_0 – точность представления нуля для ЭВМ. При нахождении x_r^* с точностью $\varepsilon_x = 10^{-6}$ либо $\varepsilon_g = 10^{-10}$ достаточно положить $\varepsilon_0 = 10^{-20}$. Отметим, что $R_\alpha(\xi_k) = I_n$ при $\xi_k = 0$, т.е. растяжение пространства не будет выполняться. Это всегда имеет место на итерации метода при $k = 0$, поскольку $\tilde{g}_0 = g_f(x_0)$. Кроме того $\xi_k = 0$ может иметь место на "предельных"шагах метода при выборе либо большого значения α, либо слишком малых значений ε_x ($\varepsilon_x \sim 10^{-10}$) и ε_g ($\varepsilon_g \sim 10^{-20}$). Однако такая ситуация имеет "вырожденный"характер, так как получение x_r^* с достаточно высокой точностью ($\sim 10^{-6}$) обеспечивает выбор параметров $\varepsilon_x = 10^{-6}$, $\varepsilon_g = 10^{-10}$ и $\alpha \in [2, 3]$. Тем не менее, если значение x_r^* должно быть более точным, то в этом случае можно проводить процедуру "восстановления"процесса, положив $B_{k+1} = I_n$ и установив величину шага h_{k+1} порядка $\|x_k - x_{k-1}\|$.

2. Операция растяжения пространства

Вычисляем следующие величины

2.1. $B_{k+1} = B_k + (\beta - 1)(B_k \xi_k)\xi_k^T$, где $\beta = \frac{1}{\alpha}$ (пересчет матрицы об-

ратного преобразования пространства при переходе в очередное преобразованное пространство аргументов, который следует из соотношения $B_{k+1} = (R_\alpha(\xi_k)A_k)^{-1} = B_k R_\beta(\xi_k) = B_k(I_n + (\beta-1)\xi_k\xi_k^T))$.

2.2. $\tilde{g}_{k+1} = g_k^* + (\beta-1)(g_k^*, \xi_k)\xi_k^T$ (пересчет текущего субградиента при переходе в очередное преобразованное пространство аргументов, который следует из соотношения $\tilde{g}_{k+1} = g_{\varphi_{k+1}}(y_k) = B_{k+1}^T g_f(x_k) = R_\beta(\xi_k)B_k^T g_f(x_k) = R_\beta(\xi_k)g_k^* = (I_n + (\beta-1)\xi_k\xi_k^T)g_k^*)$.

3. Процедура одномерного спуска по направлению

3.1. Вычисляем $p_{k+1} = B_{k+1}\frac{\tilde{g}_{k+1}}{\|\tilde{g}_{k+1}\|}$ (направление спуска, соответствующее направлению нормированного субградиента в преобразованном пространстве аргументов $Y_{k+1} = A_{k+1}X = B_{k+1}^{-1}X$).

3.2. Осуществляем спуск по направлению -p_{k+1} с адаптивной регулировкой шага.

Положим $z_0 = x_k$, $l = 0$, $\tilde{h}_0 = h_k$. Вычисляем

$$z_{l+1} = z_l - \tilde{h}_l p_{k+1},$$

(здесь $\tilde{h}_l = h_k q_2^{\max\{0, l+1-n_h\}}$) до тех пор, пока не выполнится одно из следующих условий:

3.2.1. $(p_{k+1}, g_f(z_{l+1})) \le 0$ (задает условие спуска по направлению, его выполнение обеспечивается в силу предположения $\lim\limits_{|x|\to+\infty} f(x) \to +\infty$).

3.2.2. $\|g_f(z_{l+1})\| \le \varepsilon_g$, тогда $x_r^* = z_{l+1}$ и останов (задает останов по норме субградиента и в основном имеет место для гладких функций).

Если условие спуска (п.3.2.1) выполнилось на первом шаге, т.е. при $l = 0$, то полагаем $\tilde{h}_0 = q_1 h_k$ (уменьшение шага, которое используется, как правило, при минимизации гладких функций).

4. Переход к очередному шагу.

4.1. Полагаем $x_{k+1} = z_{l+1}$, $g_{k+1} = g_f(z_{l+1})$ и $h_{k+1} = \tilde{h}_l$.

4.2. Если $\|x_{k+1} - x_k\| \le \varepsilon_x$, то $x_r^* = x_{k+1}$ и останов (по отклонению аргумента). В противном случае переходим $(k+1)$-й итерации процесса с новыми значениями величин x_{k+1}, h_{k+1}, $\tilde{g}_{k+1}$, $g_{k+1} = g_f(x_{k+1})$ и матрицы B_{k+1}.

Для B-формы $r(\alpha)$-алгоритма (т.е. использующего коррекцию матрицы обратного преобразования B) приведенная вычислительная схема является наиболее экономной по числу арифметических операций. Именно она лежит в основе практически используемой программной реализации $r(\alpha)$-алгоритма [4] и требует на одной итерации всего три операции умножения матрицы на вектор ($3n^2$ арифметических операций умножения и столько же операций сложения) и одну операцию одноранговой коррекции матрицы B_{k+1} (n^2 арифметических операций умножения и столько же операций сложения). Прямолинейная реализация B-формы $r(\alpha)$-алгоритма в рамках общей схемы r-алгоритмов, описанной в работе [5], потребовала бы одну дополнительную операцию умножения матрицы на вектор, а трудоемкость одной итерации составила бы $5n^2$ арифметических операций умножения вместо $4n^2$ для приведенной вычислительной схемы $r(\alpha)$-алгоритма.

Вычислительная эффективность $r(\alpha)$-алгоритма зависит от коэффициента растяжения пространства и параметров адаптивной регулировки шага. Суть выбора параметров состоит в том, чтобы адаптивный способ регулировки шагового множителя позволял увеличивать точность поиска минимума функции по направлению в процессе счета и при этом число шагов по направлению не должно быть большим. При минимизации негладких функций это обеспечивает 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 ($q_1 = 0.8 \div 0.95$). Это обусловлено тем, что дополнительное измельчение шага способствует увеличению точности поиска минимума функции по направлению, что при минимизации гладких функций обеспечивает более быструю скорость сходимости.

При таком выборе параметров $r(\alpha)$-алгоритм, как правило, дает следующие результаты: число спусков по направлению редко превосходит два, за n шагов точность по функционалу улучшается в три-пять раз. Выбирая в качестве критериев останова $\varepsilon_x, \varepsilon_g \sim 10^{-6} \div 10^{-5}$ при минимизации выпуклой функции, даже существенно овражной структуры, можно обеспечить нахождение x_r^* со значением целевой функции, достаточно близким к оптимальному ($\frac{f(x_r^*)-f(x^*)}{|f(x^*)|+1} \sim 10^{-6} \div 10^{-5}$ – для негладких и $\frac{f(x_r^*)-f(x^*)}{|f(x^*)|+1} \sim 10^{-12} \div 10^{-10}$ – для гладких функций). Это подтверждается результатами многочисленных тестовых и реальных расчетов.

Приведенная вычислительная схема $r(\alpha)$-алгоритма реализована в виде ФОРТРАНовской программы ralgb4 (r-**алг**оритм в **B**-форме, требующий $\mathbf{4}n^2$ арифметических операций на итерации).

2 Фортрановская программа ralgb4

Здесь дадим краткую инструкцию для пользования ФОРТРАНовской программой ralgb4.

2.1 Подпрограмма ralgb4(...)

```
      subroutine ralgb4(n,x,calcfg,
     a                  alp,h0,nh,q1,q2,maxitn,epsx,epsg,intp,
     b                  fr,xr,itn,istop,
     c                  b,g,g1,g2)
c  Входные параметры:
c    n --  размерность пространства переменных
c    alp -- коэффициент растяжения пространства
c    maxitn, epsx, epsg  -- критерии останова
c    h0,nh,q1,q2 -- параметры адаптивной регулировки шагового множителя
c    intp -- параметры для печати
c    x(n) -- начальная точка (на выходе портится)
c    calcfg -- имя внешней (external) подпрограммы
c              calcfg(n,f,g,x) -- подпрограмма для вычисления f и g
c    Выходные параметры:
c      itn -- число затраченных итераций
c      istop -- код останова (2-по epsg, 3-по epsx, 4-по числу итераций,
c               5--не найден минимум по направлению)
c      xr(n) -- найденная точка минимума функции
c      fr  -- значение функции в точке минимума
c    Рабочие массивы:
c      b(n,n) -- матрица обратного преобразования пространства
c      g1(n),g2(n) -- используются для хранения промежуточных векторов
c
c    Используется функция dsqrt().
c
c    Последняя модификация 11.01.2002 /Стецюк П.И./
c
      implicit real*8(a-h,o-z),integer*4(i-n)
      external calcfg
      dimension g(n),x(n),b(n,n),xr(n),g1(n),g2(n)
      . . . . . .
      return
      end
```

Для своей работы подпрограмма ralgb4(...) требует подготовки подпрограммы, которая вычисляет значение минимизируемой функции и ее субградиента в некоторой заданной точке.

2.2 Требования к подпрограмме calcfg

Подпрограмма вычисления функции $f(x)$ и субградиента $g_f(x)$ в некоторой заданной точке $x_k \in R^n$ должна быть оформлена в виде следующей фортрановской подпрограммы:

```
      subroutine calcfg(n,f,g,x)
      real*8 g(n),x(n),f
      integer*4 n
      f= ...
      g= ...
      return
      end
```

где в качестве имени подпрограммы вместо "calcfg"можно использовать произвольное имя, построенное согласно правилам ФОРТРАНа. Эта подпрограмма должна быть обьявлена как внешняя (с помощью оператора "external") в том блоке фортрановской программы, где вызывается подпрограмма ralgb4. Имя этой программы должно быть передано на вход подпрограммы ralgb4 по месту формального параметра "calcfg".

Ниже даны примеры подпрограмм для вычисления значений функции и субградиента для следующих функций: $f_1(x_1,\ldots,x_n) = \sum_{i=1}^{n} 10^{i-1}x_i^2$ (подпрограмма fg1) и $f_2(x_1,\ldots,x_n) = \sum_{i=1}^{n} 10^{i-1}|x_i|$ (подпрограмма fg2).

```
subroutine fg1(n,f,g,x)
real*8 g(n),x(n),f
integer*4 n
real*8 a1
f=0.d0
a1=1.d0
do i=1,n
   f=f+a1*x(i)*x(i)
   g(i)=2.d0*a1*x(i)
   a1=a1*10.d0
enddo
return
end
```

```
subroutine fg2(n,f,g,x)
real*8 g(n),x(n),f
integer*4 n
real*8 a1,one
f=0.d0
a1=1.d0
do i=1,n
   f=f+a1*dabs(x(i))
   one=1.d0
   if(x(i).lt.0.d0) one=-1.d0
   g(i)=one*a1
   a1=a1*10.d0
enddo
return
end
```

При использовании эти программы должны быть обьявлены как внешние, т.е.

```
      external fg1
      external fg2
```

в том блоке FORTRAN-текста, откуда будет реализован вызов подпрограммы ralgb4.

2.3 Тестовый пример и результаты расчета

Тестовый пример и результаты расчета дадим для указанных выше функций fg1 и fg2 при размерности пространства $n = 10$ и начальной стартовой точке $x_0 = (1, \ldots, 1)^T$.

Тестовый пример задается следующей фортрановской программой:

```
      implicit real*8 (a-h,o-z),integer*4(i-n)
      real*8 x(10),g(10),b(100),g1(10),g2(10),xr(10)
      external fg1,fg2
c     Расчет для функции fg1
      n=10
      do j=1,n
         x(j)=1.d0
      enddo
c     коэффициент растяжения пространства
      alp=2.d0
c     параметры адаптивной регулировки шагового множителя
      h0=1.d0
      nh=3
      q1=0.9d0
      q2=1.1d0
c     параметры останова
      maxitn=2000
      epsx=1.d-6
      epsg=1.d-6
c     параметр управления печатью
      intp= 20
      call ralgb4(n,x,fg1,alp,h0,nh,q1,q2,maxitn,epsx,epsg,intp,
     *            fr,xr,itn,istop,b,g,g1,g2)
      write (6,'(3x,a,3x,g12.5,5x,i3)')' Решение: fr istop',fr,istop
      do j=1,n
         write (6,'(3x,a,i4,5x,g12.5)')'j xr(j) ',j,xr(j)
      enddo
c     Расчет для функции fg2
      do j=1,n
         x(j)=1.d0
      enddo
      q1=1.d0
      call ralgb4(n,x,fg2,alp,h0,nh,q1,q2,maxitn,epsx,epsg,intp,
     *            fr,xr,itn,istop,b,g,g1,g2)
      write (6,'(3x,a,3x,g12.5,5x,i3)')' Решение: fr istop',fr,istop
      do j=1,n
         write (6,'(3x,a,i4,5x,g12.5)')'j xr(j) ',j,xr(j)
      enddo
      stop
      end
```

Этому примеру соответствуют следующие результаты расчета:

```
        Протокол процесса негладкой оптимизации
 Itn.       ..f(x)..             ...f(x_r)...     LS    LSa
   0    1.1111111110D+09    1.1111111110D+09      0      0
  20    3.2936898956D+03    3.2936898956D+03     31     31
  40    1.1072478907D+01    8.8702472609D+00     31     62
  60    1.7034737651D-01    1.5423634781D-01     33     95
  80    1.5407414137D-05    1.5407414137D-05     33    128
 100    1.0964940707D-06    1.6615560203D-07     27    155
 120    3.2985652825D-09    2.3974702356D-10     25    180
 139    2.7406826476D-12    1.0813064108D-12     25    205
 Решение: fr istop     .10813E-11       3
j xr(j)    1       .19423E-06
j xr(j)    2       .11058E-06
j xr(j)    3       .15131E-08
j xr(j)    4       .16494E-07
j xr(j)    5       .23382E-08
j xr(j)    6       .11666E-08
j xr(j)    7       .16637E-09
j xr(j)    8       .79775E-10
j xr(j)    9       .59879E-10
j xr(j)   10      -.28940E-11

        Протокол процесса негладкой оптимизации
 Itn.       ..f(x)..             ...f(x_r)...     LS    LSa
   0    1.1111111110D+09    1.1111111110D+09      0      0
  20    5.6848458353D+05    5.6848458353D+05     31     31
  40    9.9523435151D+04    8.0674142382D+04     33     64
  60    4.3334269656D+03    4.3334269656D+03     24     88
  80    8.5706821918D+02    6.6756597115D+02     26    114
 100    1.0033509956D+02    1.0033509956D+02     24    138
 120    1.5133728589D+01    1.5133728589D+01     25    163
 140    5.1388661037D+00    3.4394265228D+00     22    185
 160    1.1142878606D+00    9.7020365177D-01     24    209
 180    3.0371750034D-01    2.2529921444D-01     27    236
 200    6.7702607158D-02    6.7702607158D-02     25    261
 220    1.7433792064D-02    1.2200926870D-02     25    286
 240    4.9979186805D-03    4.2374389648D-03     21    307
 260    1.0119630820D-03    7.4750818555D-04     25    332
 280    2.2629578005D-04    1.7167611945D-04     21    353
 300    6.6100694176D-05    6.6100694176D-05     22    375
 320    2.2578584764D-05    1.0478054041D-05     23    398
 327    7.0848791999D-06    7.0848791999D-06      7    405
 Решение: fr istop     .70849E-05       3
j xr(j)    1      -.23812E-06
```

```
 j xr(j)    2      -.20408E-07
 j xr(j)    3       .72061E-08
 j xr(j)    4      -.20514E-08
 j xr(j)    5       .56065E-10
 j xr(j)    6      -.32902E-11
 j xr(j)    7       .23689E-11
 j xr(j)    8       .19097E-13
 j xr(j)    9       .65513E-15
 j xr(j)   10       .35559E-15
```

Здесь LS и LSA – число осуществленных одномерных спусков по направлению (LS – между печатаемыми итерациями, LSA – на момент печатаемой итерации). Они соответствуют числу вычислений значений функции и субградиента (числу обращений к подпрограмме "calcfg").

3 Текст программы ralgb4

```
      subroutine ralgb4(n,x,calcfg,
     a                  alp,h0,nh,q1,q2,maxitn,epsx,epsg,intp,
     b                  fr,xr,itn,istop,
     c                  b,g,g1,g2)
c
c  Входные параметры:
c    n --  размерность пространства переменных
c    alp -- коэффициент растяжения пространства
c    h0,nh,q1,q2 -- параметры адаптивной регулировки шагового множителя
c    maxitn, epsx, epsg  -- критерии останова
c    intp -- печать о ходе процесса через каждые intp-итераций
c            (если intp<0 -- печати не будет)
c    x(n) -- начальная точка (на выходе портится)
c    calcfg -- имя внешней (external) подпрограммы
c              calcfg(n,f,g,x) -- подпрограмма для вычисления f и g
c    Выходные параметры:
c      itn -- число затраченных итераций
c      istop -- код останова (2-по epsg, 3-по epsx, 4-по числу итераций,
c                5--не найден минимум по направлению)
c      xr(n) -- найденная точка минимума функции
c      fr  -- значение функции в точке минимума
c    Рабочие массивы:
c      b(n,n) -- матрица обратного преобразования пространства
c      g1(n),g2(n) -- используются для хранения промежуточных векторов
c
c    Используется функция dsqrt().
c
c    Последняя модификация 11.01.2002 /Стецюк П.И./
c
      implicit real*8(a-h,o-z),integer*4(i-n)
      external calcfg
      dimension g(n),x(n),b(n,n),xr(n),g1(n),g2(n)
c     Установка нуля и начальных параметров
      dzero=1.d-20
      iprint=6
      w=1./alp-1.
      hs=h0
      itn=0
      lp=itn+intp
      do i=1,n
         do j=1,n
            b(j,i)=0.
         enddo
         b(i,i)=1.
```

```
      enddo
      call calcfg(n,f,g,x)
      fr=f
      do i=1,n
         g1(i)=g(i)
         xr(i)=x(i)
      enddo
      nls=0
      nlsa=0
      if(intp.ge.0) write(iprint,3000)
      if(intp.ge.0) write(iprint,3100) itn,f,fr,nlsa,nls
c     Основное тело программы, реализующей алгоритм
      do itn=1,maxitn
c     Критерий останова по норме градиента
         dg=0.d0
         do i=1,n
            dg=dg+g(i)*g(i)
         enddo
         istop=2
         if(dsqrt(dg).le.epsg) goto 100
c     Вычислить субградиент в преобразованном пространстве
         do i=1,n
            d=0.
            do j=1,n
               d=d+b(j,i)*g(j)
            enddo
            g2(i)=d
         enddo
c     Вычислить и нормировать разность субградиентов
         dg=0.d0
         do i=1,n
            g(i)=g2(i)-g1(i)
            dg=dg+g(i)*g(i)
         enddo
         dg=dsqrt(dg)
         if(dg.gt.dzero) then
           dg=1.d0/dg
           do i=1,n
              g(i)=dg*g(i)
           enddo
         endif
c     Вычислить нормированный субградиент в преобразованном пространстве
         d=0.d0
         do i=1,n
            d=d+g(i)*g2(i)
         enddo
```

```
         d=w*d
         d1=0.
         do i=1,n
            g1(i)=g2(i)+d*g(i)
            d1=d1+g1(i)**2
         enddo
         d1=1./dsqrt(d1)
         do i=1,n
            g2(i)=d1*g1(i)
         enddo
c     Пересчитать матрицу B
         do i=1,n
            d=0.
            do j=1,n
               d=d+b(i,j)*g(j)
            enddo
            d=w*d
            do j=1,n
               b(i,j)=b(i,j)+d*g(j)
            enddo
         enddo
c     Вычислить направление движения
         dg=0.d0
         do i=1,n
            d=0.
            do j=1,n
               d=d+b(i,j)*g2(j)
            enddo
            g(i)=d
            dg=dg+d*d
         enddo
         dg=dsqrt(dg)
c      Одномерный спуск по направлению
         ls=0
         lsa=0
         dx=0.d0
         istop=5
   20    continue
           ls=ls+1
           dx=dx+hs*dg
           do i=1,n
              x(i)=x(i)-hs*g(i)
           enddo
           call calcfg(n,f,g2,x)
           if(f.lt.fr) then
             fr=f
```

```
            do i=1,n
               xr(i)=x(i)
            enddo
          endif
          d=0.
          do i=1,n
             d=d+g(i)*g2(i)
          enddo
          if(ls.gt.500) goto 100
          if(ls.gt.nh) hs=hs*q2
        if(d.gt.0.d0) go to 20
        if(ls.eq.1) hs=hs*q1
        nls=nls+ls
        nlsa=nlsa+ls
        if(itn.eq.lp) then
          write(iprint,3100) itn,f,fr,nlsa,nls
          nlsa=0
          lp=lp+intp
        endif
        do i=1,n
           g(i)=g2(i)
        enddo
        istop=3
        if(dabs(dx).lt.epsx) goto 100
      enddo
      itn=itn-1
      istop=4
  100 continue
      if(intp.ge.0) write(iprint,3100) itn,f,fr,nlsa,nls
      return
 3000 format(/10x,' Протокол процесса негладкой оптимизации '/2x,
     a        '  Itn.',7x,'..f(x)..',13x,'...f(x_r)...',
     b        5x,'LS',4x,'LSa')
 3100 format(2x,i5,2(2x,1pd18.10),3(2x,i5))
      end
```

Список литературы

[1] Шор Н.З., Журбенко Н.Г. Метод минимизации, использующий операцию растяжения пространства в направлении разности двух последовательных градиентов. – Кибернетика, 1971. – N3. – С. 51-59.

[2] Шор Н.З. Методы минимизации недифференцируемых функций и их приложения. - Киев: Наукова думка, 1979. - 199с.

[3] Шор Н.З., Стеценко С.И. Квадратичные экстремальные задачи и недифференцируемая оптимизация. – Киев: Наук. думка, 1989. – 208с.

[4] Журбенко Н.Г., Марчук Т.В. Алгоритм минимизации негладких функционалов /$r(\alpha)$-алгоритм/, АН УССР, РФАП, N 22, 1976 г.

[5] Шор Н.З. Минимизация матричных функций и недифференцируемая оптимизация // Обозрение прикладной и промышленной математики. – 1995. – Т2. – С. 113-138.

[6] Shor N.Z. Nondifferentiable optimization and polynomial problems. – Kluwer Academic Publishers. – 1998. – 394 p.

[7] Шор Н.З., Стецюк П.И. Использование модификации r-алгоритма для нахождения глобального минимума полиномиальных функций // Кибернетика и систем. анализ. – 1997. – N4. – С. 28-49.