4. Численные методы минимизации.

Рассматривается задача безусловной минимизации целевой функции $f(x): R^n \to R$ (без ограничений):

$$f(x) \to \min, \quad x \in R^n$$

Идея методов приближенного решения задачи состоит в построении последоваптельности точек
$x^{(0)}, x^{(1)}, \ldots, x^{(k)}, \ldots,$ удовлетворяющих условию $f(x^{(0)}) \geq f(x^{(1)}) \geq \ldots \geq f(x^{(k)}) \geq \ldots$. Такие последовательности $\{ x^{(k)} \}$ называются релаксационными, а методы – методами спуска.

Необходимое условие экстремума функции - $\nabla f(x^*) = 0$, где x^* – вектор.

Метод градиентного спуска

Процедура построения последовательности векторов $\{x^{(k)}\}$ организована следующим образом:

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} y^{(k)}, \quad k = 0,1,2 \ldots \quad y^{(k)} = -\nabla f(x^{(k)}) \qquad (4.1)$$

где величины шага спуска $\alpha^{(k)} > 0$ выбираются достаточно малыми для того, чтобы выполнялось условие последовательного убывания значений целевой функции $f(x)$ в точках $x^{(k)}$.

Итерационная процедура спуска с постоянным шагом $\alpha = \text{const}$ проста, но при реализации требует знания величины шага α, который на практике определяется подбором. При этом выбор большого шага α может привести к нарушению условия убывания целевой функции

$$f(x^{(k+1)}) = f(x^{(k)} + \alpha y^{(k)}) < f(x^{(k)}) \qquad (4.2)$$

Пример 4.1. Минимизировать целевую функцию

$$f(x) = f(x_1, x_2) = x_1^2 + 2 * x_2^2 + e^{x_1 + x_2}, \quad x \in R^2$$

методом градиентного спуска.

Расчет завершить при $\left|\frac{\partial f(x^{(k)})}{\partial x_i}\right| \leq 0.005, \ i = 1,2.$

Решение:

Имеем

$$f(x) = f(x_1, x_2) = x_1^2 + 2 * x_2^2 + e^{x_1 + x_2}$$

$$\frac{\partial f}{\partial x_1} = 2 * x_1 + e^{x_1 + x_2}$$

$$\frac{\partial f}{\partial x_2} = 4 * x_2 + e^{x_1 + x_2}$$

$$\nabla f(x_1, x_2) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2} \right)$$

$$\left|\frac{\partial f}{\partial x}\right| = \sqrt{\left[\frac{\partial f}{\partial x_1}\right]^2 + \left[\frac{\partial f}{\partial x_2}\right]^2} = \left((2 * x_1 + e^{x_1+x_2})^2 + (4 * x_2 + e^{x_1+x_2})^2 \right)^{0.5}$$

Итерационная процедура строится

$$x^{(k+1)} = x^{(k)} - \alpha^{(k)} \nabla f(x^{(k)}), \quad k = 0,1,2 \ldots$$

при ограничениях (4.2): $f(x^{(k+1)}) = f(x^{(k)} + \alpha y^{(k)}) < f(x^{(k)})$

Выберем начальное приближение $x^{(0)} = (0,0)$ и величину шага спуска $\alpha = 1$. Будем последовательно уменьшать шаг спуска в два раза, если функция не убывает.

Программа поиска минимума [Python]

```
from mpl_toolkits.mplot3d import axes3d
import matplotlib.pyplot as plt
import numpy as np
import math

def funct(x1,x2):
    return (x1**2+2*x2**2+math.exp(x1+x2))
#   Программы расчета компонент градиента функции и модуля градиента
def dfdx1(x1,x2):
    return (2*x1+math.exp(x1+x2))
def dfdx2(x1,x2):
    return (4*x2+math.exp(x1+x2))
def absdfdx(x1,x2):
     return (math.sqrt(dfdx1(x1,x2)**2 + dfdx2(x1,x2)**2 ) )
eps = 1.0e-4
x1 = 0
x2 = 0
alfa = 1
i = 0
f = funct(x1,x2)
gradf = np.zeros(2)
while  (absdfdx(x1,x2) > eps):
    i = i + 1
    gradf[0] = dfdx1(x1,x2)
    gradf[1] = dfdx2(x1,x2)
    x1new = x1 - alfa*gradf[0]
    x2new = x2 - alfa*gradf[1]
    fnew = funct(x1new,x2new)
    if (fnew > f):
        alfa = 0.5*alfa
    else:
```

```
        x1 = x1new
        x2 = x2new
    print("i=%d   x1=%0.4f x2=%0.4f   f=%0.8f" % (i,x1,x2,funct(x1,x2)))
```

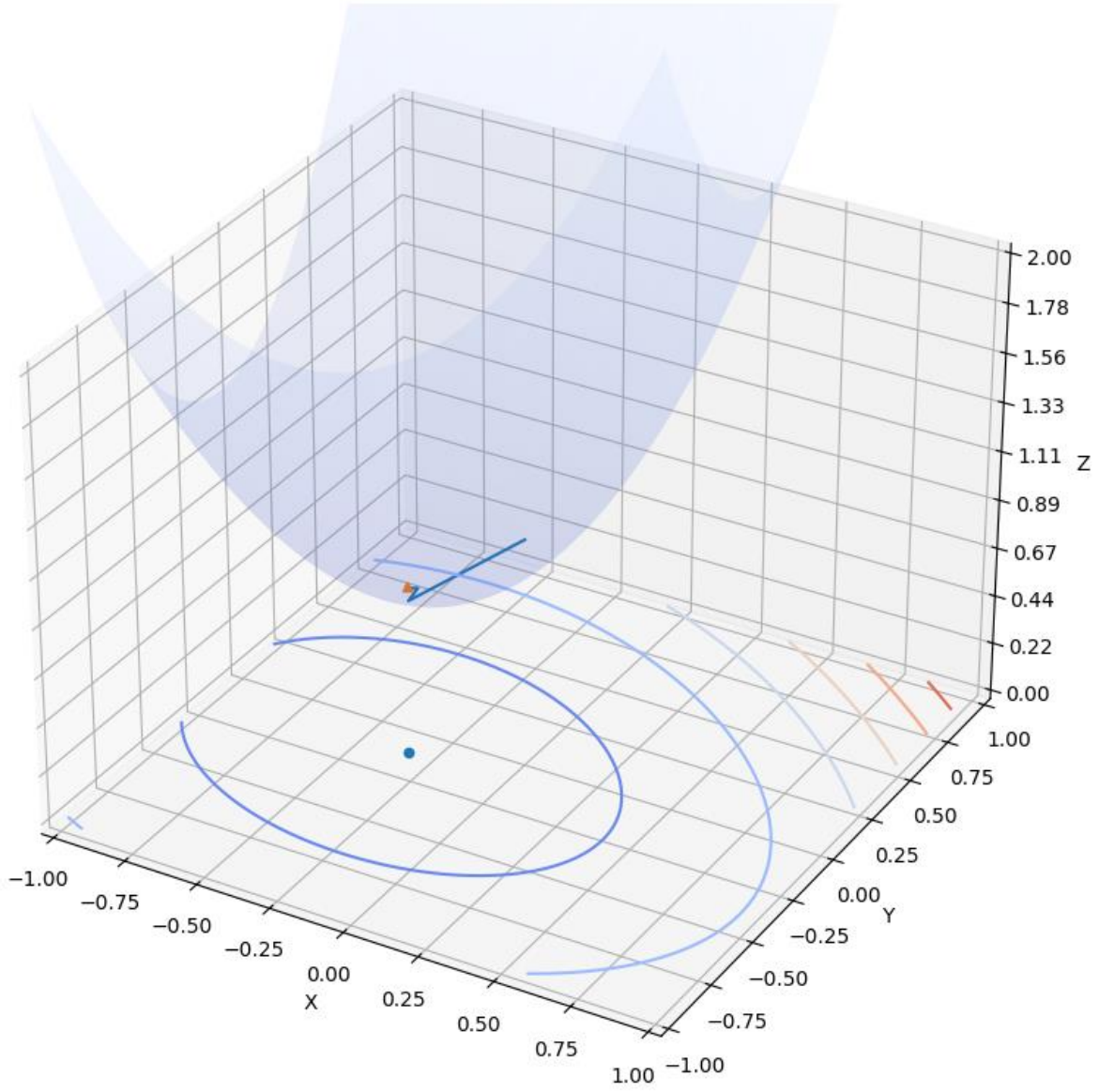
Листинг с ответом.

```
i=1   x1=0.0000 x2=0.0000   f=1.00000000
i=2   x1=0.0000 x2=0.0000   f=1.00000000
i=3   x1=-0.2500 x2=-0.2500   f=0.79403066
i=4   x1=-0.2766 x2=-0.1516   f=0.77414905
i=5   x1=-0.3012 x2=-0.1629   f=0.77249448
i=6   x1=-0.3078 x2=-0.1572   f=0.77229983
i=7   x1=-0.3109 x2=-0.1570   f=0.77227289
i=8   x1=-0.3120 x2=-0.1566   f=0.77226893
i=9   x1=-0.3125 x2=-0.1565   f=0.77226833
i=10   x1=-0.3127 x2=-0.1564   f=0.77226824
i=11   x1=-0.3127 x2=-0.1564   f=0.77226823
i=12   x1=-0.3128 x2=-0.1564   f=0.77226823
```

Программа поиска минимума [Python] с графикой (Шакиров)

```
import matplotlib.pyplot as plt
import numpy as np

from matplotlib import cm

fig, ax = plt.subplots(subplot_kw={"projection": "3d"})

# Make data for matplotlib
X = np.arange(-1, 1, 0.01)
Y = np.arange(-1, 1, 0.01)
X, Y = np.meshgrid(X, Y)
Z = X**2+2*Y**2+np.exp(X+Y)

def funct(x1,x2):
    return (x1**2+2*x2**2+np.exp(x1+x2))
#   Программы расчета компонент градиента функции и модуля градиента
def dfdx1(x1,x2):
    return (2*x1+np.exp(x1+x2))
def dfdx2(x1,x2):
    return (4*x2+np.exp(x1+x2))
def absdfdx(x1,x2):
     return (np.sqrt(dfdx1(x1,x2)**2 + dfdx2(x1,x2)**2 ) )

eps = 1.0e-4
x1, x2 = 0, 0
alfa = 1
f = funct(x1,x2)
```

```
gradf = np.zeros(2)
x1arr = [x1,]
x2arr = [x2,]
farr = [f,]
i = 0
while  (absdfdx(x1,x2) > eps):
    i = i + 1
    gradf[0] = dfdx1(x1,x2)
    gradf[1] = dfdx2(x1,x2)
    x1new = x1 - alfa*gradf[0]
    x2new = x2 - alfa*gradf[1]
    fnew = funct(x1new,x2new)
    if (fnew > f):
        alfa = 0.5*alfa
    else:
        x1 = x1new
        x2 = x2new
        f = fnew
        x1arr.append(x1new)
        x2arr.append(x2new)
        farr.append(fnew)
    print("i=%d  alfa=%0.4f x1=%0.4f x2=%0.4f   f=%0.8f" %
(i,alfa,x1,x2,funct(x1,x2)))

# Plot the surface
ax.plot_surface(X, Y, Z, cmap=cm.coolwarm, alpha=.1)
# O marker
ax.scatter(x1arr[-1], x2arr[-1], 0, marker='o')
# ^ marker
ax.scatter(x1arr[-1], x2arr[-1], farr[-1], marker='^')

# Печатаем линию градиентного спуска
ax.plot(x1arr, x2arr, farr)

ax.contour(X, Y, Z, zdir='z', offset=0, cmap='coolwarm')
#ax.contour(X, Y, Z, zdir='x', offset=-1, cmap='coolwarm')
#ax.contour(X, Y, Z, zdir='y', offset=-1, cmap='coolwarm')

ax.set(xlim=(-1, 1), ylim=(-1, 1), zlim=(0, 2),
       xlabel='X', ylabel='Y', zlabel='Z')

plt.show()
```



Программа поиска минимума [GNU Octave]

Программа расчета f(x) (в файле fgd.m)

```
function ft = fgd(x1,x2)
    ft = x1^2 + 2*x2^2 + exp(x1+x2);
endfunction
```

Программы расчета компонент градиента функции (файлы dfdx1.m, dfdx2.m)

```
function dfdx1t = dfdx1(x1,x2)
    dfdx1t = 2*x1+ exp(x1+x2);
endfunction
function dfdx2t = dfdx2(x1,x2)
    dfdx2t = 4*x2+ exp(x1+x2);
endfunction
```

Программа расчета модуля градиента функции (файл absgradf.m)

```
function moddf = absgradf(x1,x2)
    moddf = ( (2*x1 + exp(x1+x2))^2 + (4*x2+ exp(x1+x2))^2 )^0.5;
endfunction
```

Программа поиска минимума функции методом градиентного спуска (файл GradientDescent.m)

```
strftime ("%Y-%m-%d %H:%M:%S", localtime (time ()))   % Печать даты расчета
x = [0;0];
alfa = 1; i = 0;
while (absgradf(x(1), x(2)) > 0.005 )
    i=i+1;                      % Номер итерации
    f = fgd(x(1),x(2));      % Функция в точке x
    gradf = [dfdx1(x(1), x(2)); dfdx2(x(1), x(2) ) ];  % Градиент в точке x
    xnew = x - alfa*gradf;
    fnew = fgd(xnew(1),xnew(2));      % Функция в точке x+ alfa*grad(f)
    if (fnew > f )
       alfa = 0.5*alfa;
     else
       x = xnew;
    endif
 fprintf(' \n i=  '); disp(i);
%  fprintf(' alfa=,  f=   x= ');
 display(alfa);
 display (f); display (x');
endwhile
fprintf('\n Fmin(x) =   '); disp(fnew);
fprintf(' x =  '); disp(xnew'  );
```

Листинг результатов работы программы

```
>> GradientDescent

ans = 2022-09-23 18:14:38

 i=  1
alfa = 0.5000
f = 1
   0   0

 i=  2
alfa = 0.2500
f = 1
   0   0

 i=  3
alfa = 0.2500
f = 1
  -0.2500  -0.2500

 i=  4
alfa = 0.2500
f = 0.7940
  -0.2766  -0.1516

 i=  5
alfa = 0.2500
f = 0.7741
  -0.3012  -0.1629

i=  6
alfa = 0.2500
f = 0.7725
```

```
 -0.3078  -0.1572

 i=  7
alfa = 0.2500
f = 0.7723
  -0.3109  -0.1570

 Fmin(x) =   0.7723
 x =    -0.3109  -0.1570
```

Таким образом, решение задачи -

$x^* \approx x^{(7)} = (-0.3109, -0.1570)$, $f^* \approx f(-0.3109, -0.1570) = 0.7723$.

Задача 4.1 Минимизировать целевую функцию

$$f(x) = f(x_1, x_2, x_3) = (x_1 + x_2 + x_3)^2 + e^{x_1^2},$$

при $x^{(0)} = (1,1,1)$, $a = 0.1$

Метод наискорейшего спуска

Процедура построения последовательности векторов $\{x^{(k)}\}$ организована следующим образом:

$x^{(k+1)} = x^{(k)} + \alpha^{(k)} y^{(k)}$, $k = 0,1,2 \ldots$ $y^{(k)} = -\nabla f(x^{(k)})$

В качестве величины шага спуска $\alpha^{(k)} > 0$ выбирают решение задачи одномерной минимизации:

$$g^{(k)}(a^{(k)}) = \min g^{(k)}(\alpha), \quad g^{(k)}(a) = f(x^{(k)} + a y^{(k)})$$

Таким образом, в данном методе спуск производится в направлении наибыстрейшего убывания целевой функции и при этом с максимально возможным шагом.

Пример 4.2. Минимизировать целевую функцию

$$f(x) = f(x_1, x_2) = x_1^2 + 2 * x_2^2 + e^{x_1 + x_2}, \quad x \in R^2$$

методом наискорейшего спуска, завершив вычисления при

$$\left|\frac{\partial f(x^{(k)})}{\partial x_i}\right| \le 0.005, \quad i = 1,2.$$

Решение:

$$f(x) = f(x_1, x_2) = x_1^2 + 2 * x_2^2 + e^{x_1 + x_2}$$

$$\frac{\partial f}{\partial x_1} = 2 * x_1 + e^{x_1 + x_2}$$

$$\frac{\partial f}{\partial x_2} = 4 * x_2 + e^{x_1 + x_2}$$

$$\nabla f(x_1, x_2) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}\right)$$

Шаг 0. Положим начальное приближение $x^{(0)} = (0,0)$, тогда

$$\frac{\partial f}{\partial x_1} = 2 * x_1 + e^{x_1+x_2} = 1,$$
$$\frac{\partial f}{\partial x_2} = 4 * x_2 + e^{x_1+x_2} = 1,$$

$g^{(0)}(a) = f(x^{(0)} - a \nabla f(x^{(0)}) = f(0\text{-}1a, 0\text{-}1a) = 3a^2 + e^{-2a}$
a отвечает $\min g^{(0)}(a) = 3a^2 + e^{-2a}$

Методом перебора можно найти $a^{(0)} = 0.22$. Тогда имеем

$x^{(1)} = (0,0) - 0.22*(1,1) = (-0.22, -0.22)$.

Шаг 1. $\nabla f(x^{(1)}) = (0.204, -0.236)$,
$g^{(1)}(a) = f(x^{(1)} - a \nabla f(x^{(1)}) = f(-0.22 - a*0.204, -0.22 + a*0.236) =$

$= (-0.22 - a*0.204)^2 + (-0.22 - a*0.236)^2 + \exp(-0.44 + a*0.032)$

Методом перебора $a^{(1)} = 0.32$. Тогда
$x^{(2)} = (-0.22, -0.22) - 0.32*(0.204, -0.236) = (-0.2853, -0.1445)$.
и т.д.
На шаге 3 получаем
$x^{(3)} = (-0.305, -0.162)$
$f^* \approx f(-0.305, -0.162) = 0.772$.

Программа поиска минимума функции методом наискорейшего спуска [Python]

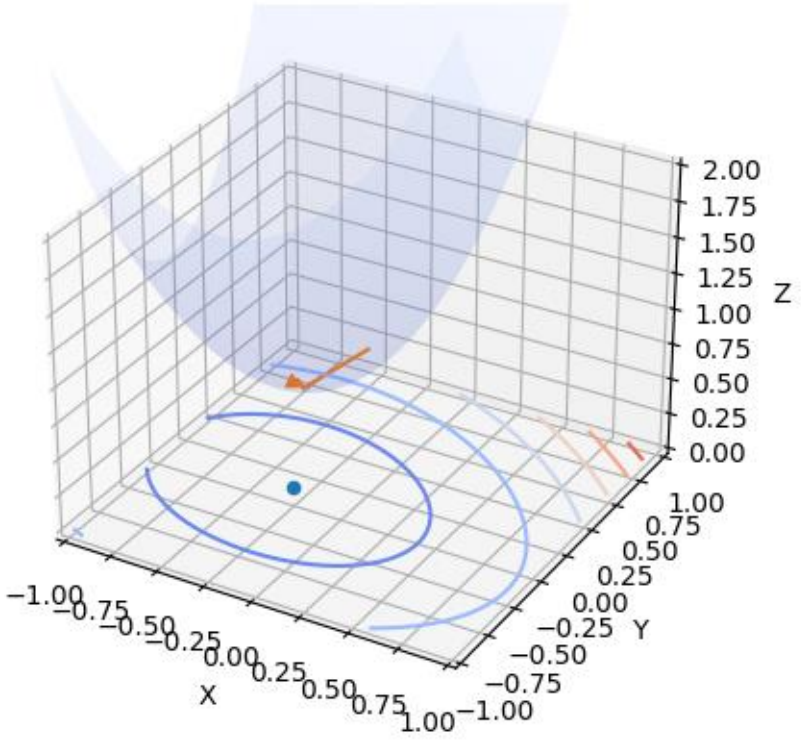
```
import matplotlib.pyplot as plt
import numpy as np
from matplotlib import cm
fig, ax = plt.subplots(subplot_kw={"projection": "3d"})
# Make data for matplotlib
X = np.arange(-1, 1, 0.01)
Y = np.arange(-1, 1, 0.01)
X, Y = np.meshgrid(X, Y)
Z = X**2+2*Y**2+np.exp(X+Y)
def funct(x1,x2):
    return (x1**2+2*x2**2+np.exp(x1+x2))
# Программы расчета компонент градиента функции и модуля градиента
def dfdx1(x1,x2):
    return (2*x1+np.exp(x1+x2))
def dfdx2(x1,x2):
```

```
        return (4*x2+np.exp(x1+x2))
def absdfdx(x1,x2):
        return (np.sqrt(dfdx1(x1,x2)**2 + dfdx2(x1,x2)**2 ) )
# Программы расчета функции при заданном значении Альфа
def functalfa(alfa,x1,x2):
        xx1 = x1-alfa*dfdx1(x1,x2)
        xx2 = x2-alfa*dfdx2(x1,x2)
        ff = funct(xx1,xx2)
        return ff
eps = 1.0e-4
sigma = eps
x1, x2 = 0, 0
f = funct(x1,x2)
gradf = np.zeros(2)
x1arr = [x1,]
x2arr = [x2,]
farr = [f,]
i = 0
while (absdfdx(x1,x2) > eps):
        i = i + 1
        gradf[0] = dfdx1(x1,x2)
        gradf[1] = dfdx2(x1,x2)
        a = -2.0
        b = 2.0
        ii = 0
 #    Поиск минимума функции одной переменной (метод деления отрезка пополам)
        while  ii < 10:
                ii=ii+1
                z1 = 0.5*(a+b-sigma)
                z2 = 0.5*(a+b+sigma)
                ff1 = functalfa(z1, x1, x2)
                ff2 = functalfa(z2, x1, x2)
                if  ff1 <= ff2:
                        anew = a
                         bnew = z2
                else:
                        anew = z1
                        bnew = b
                a = anew
                b = bnew
        alfamin = 0.5*(a+b)
        print("alfamin", alfamin)
        x1new = x1 - alfamin*gradf[0]
        x2new = x2 - alfamin*gradf[1]
```

```
        fnew = funct(x1new,x2new)
        x1 = x1new
        x2 = x2new
        f = fnew
        x1arr.append(x1new)
        x2arr.append(x2new)
        farr.append(fnew)
print("i=%d alfa=%0.4f x1=%0.4f x2=%0.4f f=%0.8f" %
(i,alfamin,x1,x2,funct(x1,x2)))
# Plot the surface
ax.plot_surface(X, Y, Z, cmap=cm.coolwarm, alpha=.1)
# O marker
ax.scatter(x1arr[-1], x2arr[-1], 0, marker='o')
# ^ marker
ax.scatter(x1arr[-1], x2arr[-1], farr[-1], marker='^')
# Печатаем линию градиентного спуска
ax.plot(x1arr, x2arr, farr)
ax.contour(X, Y, Z, zdir='z', offset=0, cmap='coolwarm')
#ax.contour(X, Y, Z, zdir='x', offset=-1, cmap='coolwarm')
#ax.contour(X, Y, Z, zdir='y', offset=-1, cmap='coolwarm')
ax.set(xlim=(-1, 1), ylim=(-1, 1), zlim=(0, 2),
xlabel='X', ylabel='Y', zlabel='Z')
plt.show()
```

Листинг результатов

```
        alfamin 0.21679145507812503
        alfamin 0.3300698730468751
        alfamin 0.23632221679687504
        alfamin 0.32616372070312505
        alfamin 0.240228369140625
        alfamin 0.322257568359375
        alfamin 0.240228369140625
        alfamin 0.32616372070312505
        i=8  alfa=0.3262  x1=-0.3128  x2=-0.1564  f=0.77226823
```



Задача 4.2 Минимизировать целевую функцию f(x) методом наискорейшего спуска, заканчивая вычисления при

$\left|\frac{\partial f(x^{(k)})}{\partial x_i}\right| \leq 0.005, \;\; i = 1,2.$

A) $f\ (x) = \ f(\ x_1, x_2) = x_1^2 + 4 * x_1 * x_2 \ + 17 * \ x_2^2 \ + 5x_2 \quad ;$

B) $f\ (x) = \ f(\ x_1, x_2) = x_1^2 + x_2^2 + x_1 \quad + x_2 \quad .$

Метод сопряженных градиентов

Процедура построения последовательности векторов $\{x^{(k)}\}$ организована следующим образом:

$$x^{(k+1)} = x^{(k)} - \alpha^{(k)} p^{(k)}, \quad k = 0,1,2 \ldots$$

$$p^{(0)} = \nabla f(x^{(0)}), \quad p^{(k)} = \nabla f(x^{(k)}) + \beta^{(k)} p^{(k-1)}, \quad k = 0,1,2 \ldots \qquad (4.3)$$

В качестве величины шага спуска $\alpha^{(k)} > 0$ выбирают решение задачи одномерной минимизации:

$$g^{(k)}(\ a^{\ (k)}) = \min g^{(k)}\ (\alpha), \quad g^{(k)}(\ a) = f(x^{(k)} + a y^{(k)}\ ).$$

Параметр $\beta^{(k)}$ определяется по формуле

$$\beta^{(k)} = \frac{\left\|\nabla f(x^{(k)})\right\|^2}{\left\|\nabla f(x^{(k-1)})\right\|^2} = \frac{\sum_{i=1}^{n}\frac{\partial f(x^{(k)})^2}{\partial x_i}}{\sum_{i=1}^{n}\frac{\partial f(x^{(k-1)})^2}{\partial x_i}}, \qquad k = 0,1,2 \ldots$$

Таким образом, метод сопряженных градиентов отличается от метода наискорейшего спуска только выбором направления спуска (- $p^{(k)}$ вместо $y^{(k)} = -\nabla f(x^{(k)})$). При этом $p^{(k)}$ из выражения (4.3) определяется не только антиградиентом $-\nabla f(x^{(k)})$, но и направлением спуска - $p^{(k-1)}$ на предыдущем шаге. Это позволяет более полно, чем в градиентных методах учитывать особенности функции f (x) при построении последовательных приближений к ее точке минимума.

Условием окончания итерационной процедуры метода сопряженных градиентов является выполнение требования на точность решения задачи минимизации:

$$\left|\frac{\partial f(x^{(k)})}{\partial x_i}\right| \le \varepsilon, \ \ i = 1,2\ldots, n, \qquad \text{или} \qquad \left\|\nabla f(x^{(k)})\right\| \le \varepsilon, \ \ \varepsilon > 0.$$

Пример 4.3. Минимизировать целевую функцию

$$f(x) = f(x_1, x_2) = x_1^2 + 2 * x_2^2 , \quad x \in R^2$$

методом сопряженных градиентов.

Расчет завершить при $\left|\frac{\partial f(x^{(k)})}{\partial x_i}\right| \le 0.005, \ \ i = 1,2.$

Решение:

Шаг 0. Положим начальное приближение $x^{(0)} = (1,1)$, тогда

$\nabla f(x^{(0)}) = (2,4)$, $p^{(0)} = \nabla f(x^{(0)}) = (2,4)$,

$g^{(0)}(a) = f(x_1, x_2) = x_1^2 + 2 * x_2^2 = f(1-2a , 1-4a) = 36*a^2 - 20 *a +3.$

Для нахождения точки минимума $a^{(0)}$ функции $g^{(0)}(a)$ используем условие

$$\frac{dg^{(0)}(a)}{da} = 0$$

то есть $a^{(0)} = 5/18$.

Откуда $x^{(1)} = (1,1) - (5/18) *(2,4) = (4/9, -1/9)$.

Шаг 1.

$\nabla f(x^{(1)}) = \left(\frac{8}{9}, -\frac{4}{9}\right)$, $\beta^{(1)} = \frac{4}{81}$,

$$p^{(1)} = \left(\frac{8}{9}, -\frac{4}{9}\right) + \frac{4}{81}(2,4) = \left(\frac{80}{81}, -\frac{20}{81}\right),$$

$$g^{(1)}(a) = \frac{800}{729}a^2 - \frac{80}{81}a + \frac{2}{9}.$$

Минимизируя $g^{(1)}(a)$ получаем, что $a^{(1)} = \frac{9}{20}$. Имеем $x^{(2)} = \left(\frac{4}{9}, -\frac{1}{9}\right) - \frac{9}{20} * \left(\frac{80}{81}, -\frac{20}{81}\right) = (0,0)$.

Задача 4.3 Минимизировать целевую функцию f(x) методом сопряженных градиентов, заканчивая вычисления при $\left|\frac{\partial f(x^{(k)})}{\partial x_i}\right| \leq 0.005, \ i = 1,2.$

A) $f(x) = f(x_1, x_2) = x_1^2 + 2 * x_2^2 + x_1 x_2 - 7x_1 - 7x_2$; Положим начальное приближение (0, 0)

Ответ: $x^* \approx x^{(2)} = (3,1)$, $f^* \approx f(3,1) = -14$

B) $f(x) = f(x_1, x_2) = x_1^2 + x_2^2 + x_1 x_2$.