МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ИМЕНИ Н. Э. БАУМАНА

Факультет информатики и систем управления

Кафедра теоретической информатики и компьютерных технологий

Численные методы поиска безусловного экстремума

Исполнитель: Ю.А. Волкова

Группа: ИУ9-111

25 Декабря, 2016

1 Методы нулевого порядка. Метод деформируемых симплексов (Метод Недлера-Мида)

1.1 Постановка задачи

Найти безусловный минимум функции $f(x)$ многих переменных, т.е. найти такую точку $x^e \in R^n$, что

$$f(x^e) = \min_{x \in R^n} f(x)$$
$$f(x) \in C^0(X)$$

Где $f(x)$ задана уравнением:

$$f(x) = 4(x_1 - 5)^2 + 2(x_2 - 6)^4$$

1.2 Код программы

```
from math import sqrt

a = 1
b = 0.5
g = 2

n = 2
p = 0.01
eps = 0.01

def f( x ):

    x1 = x[0]
    x2 = x[1]

    return 4 * ( x1 - 5 )**2 + 2 * ( x2 - 6 )**4
```

```
def calcD( d1, d2 ):

    D = [ [ d2 ] * n for i in range( n + 1 ) ]

    for i in range( n ):
        D[ i ][ i ] = d1

    return D

def calcds():
    return  p * ( sqrt( n + 1 ) + n - 1 ) / ( n * sqrt(2) ),\
            p * ( sqrt( n + 1 ) - 1 ) / ( n * sqrt(2) )

def getMinMaxSmax( x ):

    fx = [ ( f( x[i] ), i ) for i in range( len(x) ) ]
    fx.sort()

    return  fx[0][1],\
            fx[-1][1],\
            fx[-2][1]

def getMiddle( x, h ):
    return [ sum([ x[i][j] for i in range( n + 1 ) if i != h ]) / n for j in ra

def addX( xs1, xs2 ):
    return [ x1 + x2 for x1, x2 in zip( xs1, xs2 ) ]

def subX( xs1, xs2 ):
    return [ x1 - x2 for x1, x2 in zip( xs1, xs2 ) ]
```

```
def mulX( xs, k ):
    return [ x * k for x in xs ]

def calcDiff( xs, x2 ):
    fx2 = f( x2 )
    return sqrt( sum([ f( x ) - fx2 for x in xs ]) / ( n + 1 ) )

def NedlerMid( x ):
    l, h, s = getMinMaxSmax( x )

    x2 = getMiddle( x, h )

    if calcDiff( x, x2 ) <= eps:
        return x[l], f( x[l] )

    x3 = reverse( x[h], x2 )

    if f( x3 ) <= f( x[l] ):
        x4   = tension( x2, x3 )
        x[h] = x4 if f( x4 ) < f( x[l] ) else x3

    elif f( x[s] ) < f( x3 ) and f( x3 ) <= f( x[h] ):
        x[h] = compress( x[h], x2 )

    elif f( x[l] ) < f( x3 ) and f( x3 ) <= f( x[s] ):
        x[h] = x3

    else:
        x = [ addX( x[l], mulX( subX( xj, x[l] ), 0.5 ) ) for xj in x ]
```

```
    return NedlerMid( x )


def reverse( xh, x2 ):
    return addX( x2, mulX( subX( x2, xh ), a ) )


def tension( x2, x3 ):
    return addX( x2, mulX( subX( x3, x2 ), g ) )


def compress( xh, x2 ):
    return addX( x2, mulX( subX( xh, x2 ), b ) )


def init():
    d1, d2 = calcds()
    return calcD( d1, d2 )


def main():
    x = init()

    print NedlerMid( x )


if __name__ == '__main__':
    main()
```

1.3 Результат работы программы

При значениях коэффициентов $\alpha = 1, \beta = 0.5, \gamma = 2$ и при $\epsilon = 0.01$, получили точку $x = [4.997193505583281, 5.93581556216305]$. Значение функции в этой точке $f(x) = 6.544854505667396e-05$.

2 Методы второго порядка. Метод Левенберга-Марквардта

2.1 Постановка задачи

Найти безусловный минимум функции $f(x)$ многих переменных, т.е. найти такую точку $x^e \in R^n$, что

$$f(x^e) = \min_{x \in R^n} f(x)$$

$$f(x) \in C^2(X)$$

Где $f(x)$ задана уравнением:

$$f(x) = 4(x_1 - 5)^2 + 2(x_2 - 6)^4$$

2.2 Вспомогательные вычисления

$$\nabla f(x) = [8x_1 - 40, 8x_2^3 - 144x_2^2 + 864x_2 - 1728]$$

$$H(x) = \begin{matrix} 8 & 0 \\ 0 & 24x_2^2 - 288x_2 + 864 \end{matrix}$$

2.3 Код программы

```
from numpy import linalg as la
from math import sqrt


eps = 0.01
M   = 10000
n   = 2


def f( x ):
    x1 = x[0]
    x2 = x[1]
```

```
    return 4 * ( x1 - 5 )**2 + 2 * ( x2 - 6 )**4


def H( x ):
    return [[8 , 0],[0 , 24 * x[1]**2 - 288 * x[1] + 864]]


def grad( x ):
    return [ 8 * x[0] - 40, 8 * x[1]**3 - 144 * x[1]**2 + 864 * x[1] - 1728 ]


def E(n):
    e = [[0]*n for i in range(n)]

    for i in range(n):
        e[i][i] = 1

    return e


def norm( x ):
    return sqrt( sum([ a**2 for a in x ]) )


def addM( M1, M2 ):
    return [ [ x1 + x2 for x1, x2 in zip( Mi1, Mi2 ) ] for Mi1, Mi2 in zip( M1,


def subX( xs1, xs2 ):
    return [ x1 - x2 for x1, x2 in zip( xs1, xs2 ) ]


def mulM( M, k ):
    return [[ x * k for x in xs ] for xs in M ]


def mulMx( M, x ):
    return [ sum([ M[i][j] * x[j] for j in range(n) ]) for i in range(n) ]
```

```
def lsgnonlin( f, grad, H, x, g, k ):
    gx = grad( x )

    if norm( gx ) < eps or k >= M:
        return x, f( x )

    d  = mulMx( la.inv( addM( H( x ), mulM( E(n), g ))), gx )
    xk = subX( x, d )

    if f( xk ) < f( x ):
        return lsgnonlin( f, grad, H, xk, g / 2, k + 1 )
    else:
        return lsgnonlin( f, grad, H, xk, 2 * g, k + 1 )

def main():
    x = [1, 1]
    g = 10**4

    print lsgnonlin( f, grad, H, x, g, 0 )

if __name__ == '__main__':
    main()
```

2.4 Результат работы программы

При $\epsilon = 0.01, M = 10000$, получили точку $x = [5.0, 5.9057341898153899]$. Значение функции в этой точке $f(x) = 0.00015792351932087577$.

3 Вывод

На тестовой функции первый метод сработал точнее второго. В качестве начальной точки бралась точка $x = [1, 1]$. Итоговый безусловный минимум, полученный первым методом: точка $x = [4.997193505583281, 5.93581556216305]$. Значение функции в этой точке $f(x) = 6.544854505667396e - 05$.