**МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ Н. Э. БАУМАНА**

**Факультет информатики и систем управления**

**Кафедра теоретической информатики и компьютерных технологий**

Лабораторная работа №2

по курсу «Методы оптимизации»

# «Численные методы поиска безусловного экстремума»

Выполнил:
студент ИУ9-111
Выборнов А. И.

Руководитель:
Каганов Ю. Т.

Москва 2016

# 1. Метод деформируемых симплексов (метод Нелдера-Мида)

## 1.1. Постановка задачи

Найти минимум функции $f(x) = 2x_1^2 + x_1x_2 + x_2^2 - 6 * x_1 + -5 * x_2$.

## 1.2. Решение на языке программирования python

```
from math import sqrt

alpha, beta, gamma = 1, 0.5, 2
n = 2
eps = 1e-9

f = lambda (x1, x2): 2*x1*x1 + x1*x2 + x2*x2 - 6*x1 - 5*x2

def get_xc(x):
    return [sum(x[i][j] for i in range(n)) *1.0 / n for j in range(n)]

def get_xr(xc, xh):
    return [(1+alpha)*xc[i] - alpha*xh[i] for i in range(n)]

def diff(xs, x2):
    fx2 = f(x2)
    return sqrt(abs(sum(f(x) - fx2 for x in xs)*1.0 / (n + 1)))

def nelder_mead(x):
    while True:
        # stage 2: sort
        x = sorted(x, key=lambda x: f(x))
        l = 0 # lowest
        g = 1 # second greatest
        h = 2 # greatest

        # stage 3: get centroid of first n points
        xc = get_xc(x)

        # stage 4: reflect xh point from xc
        xr = get_xr(xc, x[h]);

        # stage 5: compare f(xi)
        if f(xr) < f(x[l]):
            xe = [(1-gamma)*xc[i]+gamma*xr[i] for i in range(n)] #tension
            x[h] = xe if f(xe) < f(xr) else xr
            break #goto 9

        if f(x[l]) < f(xr) and f(xr) < f(x[g]):
            x[h] = xr
            break #goto 9

        if f(x[g]) < f(xr) and f(xr) < f(x[h]):
            xr, x[h] = x[h], xr

        # stage 6: compress
        xs = [beta*x[h][i] + (1-beta)*xc[i] for i in range(n)]

        #stage 7
        if f(xs) <= f(x[h]):
            x[h] = xs
            break #goto 9

        #stage 8
        else:
            for k in [1,2]:
                x[k] = [x[l][i] + (x[k][i] - x[l][i])/2.0 for i in range(n)]
```

```
        break #goto 9

    # stage 9
    if diff(x, get_xc(x)) < eps:
        return x[1]

    return nelder_mead(x)

# stage 1: select n+1 points
points = [[1,0], [0,1], [1,1]]

x = nelder_mead(points)
print x
print 'f =', f(x)
```

### 1.3. Результат работы

При значениях коэффициентов $\alpha = 1, \beta = 0.5, \gamma = 2$ и $\varepsilon = 10^{-9}$ нашли точку $[1.0, 2.0]$, являющуюся точкой минимума функции $f(x)$.

$$f([1.0, 2.0]) = -8.0.$$

## 2. Метод Левенгберга-Макрквардтна

### 2.1. Постановка задачи

Найти минимум функции $f(x) = 2x_1^2 + x_1x_2 + x_2^2 - 6 * x_1 + -5 * x_2$.

### 2.2. Решение на языке программирования python

```
from math import sqrt
import numpy as np
from copy import deepcopy

n = 2
M = 10000
eps = 1e-9
gamma = 10e4
alpha = 1

f = lambda (x1, x2): 2*x1*x1 + x1*x2 + x2*x2 - 6*x1 - 5*x2
grad = lambda (x1, x2): [4*x1 + x2 - 6, x1 + 2*x2 - 5]
H = [[4,1],
     [1,2]]

def H_plus_ykE(gamma):
    return [[4+gamma,1],
            [1,2+gamma]]

def mul(H, x):
    return [sum([ H[i][j] * x[j] for j in range(n)]) for i in range(n)]

xk = [0, 0]
k = 0

while np.linalg.norm(grad(xk)) > eps and k < M:
    dk = map(lambda x: -x, mul( np.linalg.inv(H_plus_ykE(gamma)), grad(xk)))

    xn = [xk[i] + alpha*dk[i] for i in range(n)]
```

```
    if f(xn) < f(xk):
        k += 1
        gamma /= 2
    else:
        gamma *= 2
    xk = xn

print '%s in %s iteration' % (xk, k)
print 'f = %s' % (f(xk))
```

## 2.3. Результат работы

При значениях $\varepsilon = 10^{-9}, M = 10000, \gamma^0 = 10^4, \alpha = 1$ нашли точку $[1.0000000001384377, 1.9999999996655589]$, являющуюся точкой минимума функции $f(x)$.

$$f([1.0000000001384377, 1.9999999996655589]) = -8.0.$$