МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАВЧАЛЬНО-НАУКОВИЙ КОМПЛЕКС

"ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ"

НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

ФАКУЛЬТЕТ СИСТЕМНИХ ДОСЛІДЖЕНЬ

КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

КОМП’ЮТЕРНИЙ ПРАКТИКУМ

З КУРСУ "ЧИСЕЛЬНІ МЕТОДИ ОПТИМІЗАЦІЇ"

**ЛАБОРАТОРНА РОБОТА №3**

Тема: **«Числові методи нелінійної умовної оптимізації.**

**Метод проекції градієнта»**

Виконала: студентка 3-го курсу

групи КА-41

Лочман Я.В.

Київ – 2017

**ЗАВДАННЯ (ВАРІАНТ 7)**

Мінімізувати методом проекції градієнта:

***f(x, y, z) = 2x + 3y + z***

***X = { (x, y, z) | 3x 2 + 2y 2 + z 2 ≤ 1 }***

**КОД ПРОГРАМИ**

%matplotlib inline

import numpy as np

from numpy.linalg import norm

from math import sqrt

from scipy.optimize import minimize

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import axes3d

### SETTINGS

MAX\_COUNT\_ITER = 500

EPS = 10\*\*(-5)

DIM = 3

h = np.array([0.001, 0.001, 0.001], dtype = float)

###CONSTRAINTS for ellipsoid x^2/a^2 + y^2/b^2 + z^2/c^2 <= 1

A = 3.0 #actually it's a^2

B = 2.0 #actually it's b^2

C = 1.0 #actually it's c^2

### TARGET FUNCTIONS

def f\_test(x):

return 2.0 \* x[0] + 3.0 \* x[1] + x[2]

def f\_myvar(x):

return 2.0 \* x[0] + 3.0 \* x[1] + x[2]

f = f\_test

x0 = np.array([-2, -3, -7], dtype = float)

def df1(f,x,h):

return (f([x[0] + h, x[1], x[2]]) - f([x[0] - h, x[1], x[2]])) / (2.0 \* h)

def df2(f,x,h):

return (f([x[0], x[1] + h, x[2]]) - f([x[0], x[1] - h, x[2]])) / (2.0 \* h)

def df3(f,x,h):

return (f([x[0], x[1], x[2] + h]) - f([x[0], x[1], x[2] - h])) / (2.0 \* h)

def grad(f,x,h):

return np.array([df1(f,x,h[0]), df2(f,x,h[1]), df3(f,x,h[2])], dtype = float)

def alpha\_split(f,x,step,b=1,l=0.5,q=0.1):

alpha = b

while f(x+alpha\*step) > f(x) + q \* alpha \* grad(f,x,h).dot(step):

alpha \*= l

return alpha

### STOP CONDITIONS

def stop1(x1,x2,k):

d = norm(x2-x1)

plt.xlabel('iteration')

plt.ylabel('|| x\_new - x\_old || ')

plt.scatter(k, d)

return d<=EPS

def stop2(f,x1,x2,k):

d = abs(f(x2)-f(x1))

plt.xlabel('iteration')

plt.ylabel('| f(x\_new) - f(x\_old) | ')

plt.scatter(k, d)

return d<=EPS

###PROJECTIONS

def projection\_ellipsoid(point):

if point[0]\*\*2/A + point[1]\*\*2/B + point[2]\*\*2/C <= 1:

return point

dist = lambda x: norm(x - point)

ellipse = ( {'type': 'ineq', 'fun': lambda x: 1 - A \* x[0]\*\*2 - B \* x[1]\*\*2 - C \* x[2]\*\*2})

return minimize(dist, (0.0, 0.0, 0.0), method='SLSQP', constraints = ellipse).x

def projection\_ball(point, a = np.zeros((DIM)), r=1):

nrm = norm(point - a)

prj = np.array([a[i] + r \* (point[i]-a[i])/nrm for i in range(DIM)])

return prj

def my\_projection\_ellipsoid(point):

if point[0]\*\*2/A + point[1]\*\*2/B + point[2]\*\*2/C <= 1:

return point

temp = projection\_ball(point)

prj = np.array([sqrt(1/A) \* temp[0], sqrt(1/B) \* temp[1], sqrt(1/C) \* temp[2]])

return prj

def projection\_gradient\_method(f,x0,h):

fout = open('output.txt', 'w')

fout.write('The initial point is ({}, {}, {})\n\n'.format(\*x0))

x\_new = x0

k = 0

while (k<300):

k += 1

x\_old = x\_new

step = - grad(f,x\_old,h)

alpha = alpha\_split(f,x\_old,step)

t = x\_old + alpha \* step

fout.write('{iter:>3}. alpha = {al:<5.3f}, x\_{iter:<3} = ({:>7.4f}, {:>7.4f}, {:>7.4f}), '.format(iter=k, \*t, al=alpha))

x\_new = projection\_ellipsoid(t)

fout.write('prx\_{iter:<3} = ({:>7.4f}, {:>7.4f}, {:>7.4f}), f(x\_{iter:<}) = {f:>7.4f}\n'.format(iter=k, \*x\_new, f=f(x\_new)))

if (stop1(x\_old,x\_new,k) and stop2(f,x\_old,x\_new,k)):

break

print('Approximate solution found in {} iterations'.format(k))

print('> Approximate x\* = ({:>8.5f}, {:>8.5f}, {:>8.5f})'.format(\*x\_new))

print('> Approximate f(x\*) = {:>8.5f}'.format(f(x\_new)))

fout.write('\nThe approximate solution of the problem is ({:>8.5f}, {:>8.5f}, {:>8.5f})\n'.format(\*x\_new))

fout.write('The value of function in this point is {:>8.5f}\n'.format(f(x\_new)))

fout.close()

return x\_new

minim = projection\_gradient\_method(f,x0,h)

plt.show()

print()

def projection\_gradient\_method\_f(f,x0,h):

fout = open('output.txt', 'w')

fout.write('The initial point is ({}, {}, {})\n\n'.format(\*x0))

x\_new = x0

k = 0

while (k<MAX\_COUNT\_ITER):

k += 1

x\_old = x\_new

step = - grad(f,x\_old,h)

alpha = alpha\_split(f,x\_old,step)

t = x\_old + alpha \* step

fout.write('{iter:>3}. alpha = {al:<5.3f}, x\_{iter:<3} = ({:>7.4f}, {:>7.4f}, {:>7.4f}), '.format(iter=k, \*t, al=alpha))

x\_new = my\_projection\_ellipsoid(t)

fout.write('prx\_{iter:<3} = ({:>7.4f}, {:>7.4f}, {:>7.4f}), f(x\_{iter:<}) = {f:>7.4f}\n'.format(iter=k, \*x\_new, f=f(x\_new)))

if (stop1(x\_old,x\_new,k) and stop2(f,x\_old,x\_new,k)):

break

print('Approximate solution found in {} iterations'.format(k))

print('> Approximate x\* = ({:>8.5f}, {:>8.5f}, {:>8.5f})'.format(\*x\_new))

print('> Approximate f(x\*) = {:>8.5f}'.format(f(x\_new)))

fout.write('\nThe approximate solution of the problem is ({:>8.5f}, {:>8.5f}, {:>8.5f})\n'.format(\*x\_new))

fout.write('The value of function in this point is {:>8.5f}\n'.format(f(x\_new)))

fout.close()

return x\_new

minim = projection\_gradient\_method\_f(f,x0,h)

plt.show()

print()

**РЕЗУЛЬТАТИ РОБОТИ ПРОГРАМИ**

( реалізовано для h = [0.001, 0.001, 0.001]; eps = 10\*\*(-5); x0 = [-2, -3, -7] )

***Використовуючи функцію мінімізації з бібліотеки для пошуку проекції на еліпсоїд:***

***output.txt***

The initial point is (-2.0, -3.0, -7.0)

1. alpha = 1.000, x\_1 = (-4.0000, -6.0000, -8.0000), prx\_1 = (-0.1510, -0.3334, -0.8422), f(x\_1) = -2.1443

2. alpha = 1.000, x\_2 = (-2.1510, -3.3334, -1.8422), prx\_2 = (-0.2419, -0.5324, -0.5075), f(x\_2) = -2.5885

3. alpha = 1.000, x\_3 = (-2.2419, -3.5324, -1.5075), prx\_3 = (-0.2521, -0.5643, -0.4152), f(x\_3) = -2.6124

4. alpha = 1.000, x\_4 = (-2.2521, -3.5643, -1.4152), prx\_4 = (-0.2543, -0.5714, -0.3911), f(x\_4) = -2.6140

5. alpha = 1.000, x\_5 = (-2.2543, -3.5714, -1.3911), prx\_5 = (-0.2548, -0.5732, -0.3848), f(x\_5) = -2.6141

6. alpha = 1.000, x\_6 = (-2.2548, -3.5732, -1.3848), prx\_6 = (-0.2550, -0.5737, -0.3831), f(x\_6) = -2.6141

7. alpha = 1.000, x\_7 = (-2.2550, -3.5737, -1.3831), prx\_7 = (-0.2550, -0.5738, -0.3827), f(x\_7) = -2.6141

8. alpha = 1.000, x\_8 = (-2.2550, -3.5738, -1.3827), prx\_8 = (-0.2550, -0.5738, -0.3826), f(x\_8) = -2.6141

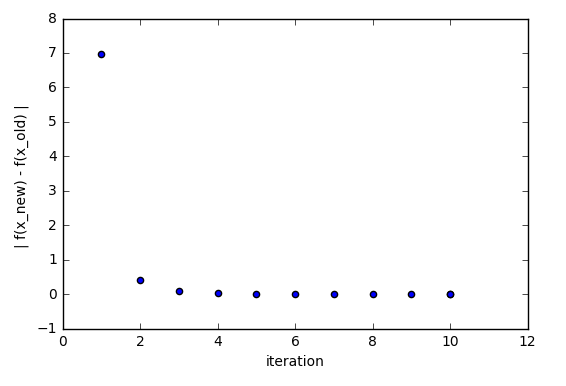
9. alpha = 1.000, x\_9 = (-2.2550, -3.5738, -1.3826), prx\_9 = (-0.2550, -0.5738, -0.3826), f(x\_9) = -2.6141

10. alpha = 1.000, x\_10 = (-2.2550, -3.5738, -1.3826), prx\_10 = (-0.2550, -0.5738, -0.3825), f(x\_10) = -2.6141

The approximate solution of the problem is (-0.25503, -0.57382, -0.38255)

The value of function in this point is -2.61406

***on the screen***



***Використовуючи власну функцію приблизного пошуку проекції на еліпсоїд:***

***output.txt***

The initial point is (-2.0, -3.0, -7.0)

1. alpha = 1.000, x\_1 = (-4.0000, -6.0000, -8.0000), prx\_1 = (-0.2144, -0.3939, -0.7428), f(x\_1) = -2.3534

2. alpha = 1.000, x\_2 = (-2.2144, -3.3939, -1.7428), prx\_2 = (-0.2898, -0.5440, -0.3951), f(x\_2) = -2.6068

3. alpha = 1.000, x\_3 = (-2.2898, -3.5440, -1.3951), prx\_3 = (-0.2975, -0.5639, -0.3139), f(x\_3) = -2.6006

4. alpha = 1.000, x\_4 = (-2.2975, -3.5639, -1.3139), prx\_4 = (-0.2988, -0.5677, -0.2960), f(x\_4) = -2.5967

5. alpha = 1.000, x\_5 = (-2.2988, -3.5677, -1.2960), prx\_5 = (-0.2991, -0.5685, -0.2920), f(x\_5) = -2.5957

6. alpha = 1.000, x\_6 = (-2.2991, -3.5685, -1.2920), prx\_6 = (-0.2991, -0.5687, -0.2912), f(x\_6) = -2.5955

7. alpha = 1.000, x\_7 = (-2.2991, -3.5687, -1.2912), prx\_7 = (-0.2992, -0.5687, -0.2910), f(x\_7) = -2.5954

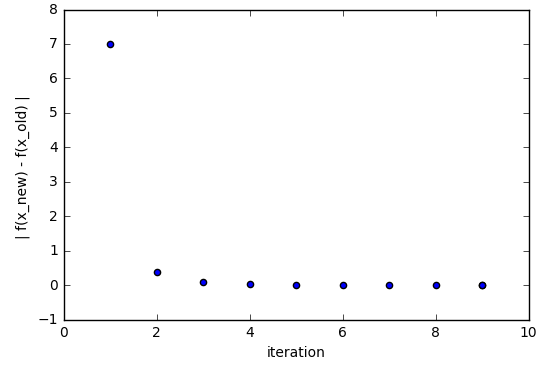
8. alpha = 1.000, x\_8 = (-2.2992, -3.5687, -1.2910), prx\_8 = (-0.2992, -0.5687, -0.2909), f(x\_8) = -2.5954

9. alpha = 1.000, x\_9 = (-2.2992, -3.5687, -1.2909), prx\_9 = (-0.2992, -0.5687, -0.2909), f(x\_9) = -2.5954

The approximate solution of the problem is (-0.29916, -0.56871, -0.29094)

The value of function in this point is -2.59539

***on the screen***



**ВИСНОВКИ**

Під час виконання лабораторної роботи, я реалізувала метод проекції градієнта для розв’язку оптимізаційної задачі. В процесі написання програми я реалізувала два способи знаходження проекції точки на дану множину (еліпсоїд). Перший полягав у використанні функції мінімізації з бібліотеки в безпосередній формулі з означення проекції точки на множину. Другий спосіб полягав у знаходженні проекції точки на одиничну сферу та застосуванні до неї афінного перетворення (дія матриці, що одиничний шар перетворює на даний еліпсоїд); цей спосіб є менш точним – в разі, якщо точка простору не знаходиться в еліпсоїді, то цей спосіб знаходить точку на ньому, яка є дуже близькою до проекції.

За першим способом метод знайшов точку мінімуму за 10 ітерацій (початкова точка [-2,-3,-7]). За другим способом метод знайшов точку мінімуму за 9 ітерацій (аналогічно початкова точка [-2,-3,-7]), похибка значення функції виявилася порядку *-2* .

Також я побудувала графік, що наглядно показує, як швидко змінюється норма різниці f(xk) та f(xk+1) (або xk та xk+1) на перших ітераціях, і як повільно — на останніх.