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 №4**

Тема: **«Методи спряжених напрямків. Метод спряжених градієнтів»**

Виконала: студентка 3-го курсу

групи КА-41

Лочман Я.В.

Київ – 2017

**ЗАВДАННЯ (ВАРІАНТ 6)**

Мінімізувати методом спряжених градієнтів:

***f(x, y) = x 2 + 4y 2 + 0.001xy - y***

**КОД ПРОГРАМИ**

%matplotlib inline

import numpy as np

from math import sqrt

from numpy.linalg import norm

from scipy.optimize import minimize

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

import pylab

### SETTINGS

MAX\_COUNT\_ITER = 5000 # global

EPS = 10\*\*(-5) # global

DIM = 2 # global

H = 0.0001 \* np.ones(DIM) # global

### INITIAL CONDITIONS

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

### TARGET FUNCTIONS

f\_type = 'quadratic'

def f\_test(x):

return 4 \* pow(x[0], 2) + pow(x[1], 2)

#return (x[1] - x[0])\*\*2 + x[0] \*\* 2 + 1

def f\_myvar(x):

# The true solution of the problem is (-0.0000625, 0.1250000)

# The value of function in this point is 2.5626251

return x[0]\*\*2 + 4.0 \* x[1]\*\*2 + 0.001 \* x[0] \* x[1] - x[1]

def f\_Rosenbrock(x):

# The true solution of the problem is (1, 1)

# The value of function in this point is 0

return 100.0 \* (x[1] - x[0]\*\*2)\*\*2 + (1.0 - x[0])\*\*2

if (f\_type == 'quadratic'):

A = np.array([[2, 0.001], [0.001, 8]], dtype = float)

b = np.array([0, -1], dtype = float)

def f(x):

a1 = 1/2 \* A[0,0]

a12 = A[0,1]

a2 = 1/2 \* A[1,1]

return a1 \* x[0] \*\* 2 + a12 \* x[0] \* x[1] + a2 \* x[1] \*\* 2 + b[0] \* x[0] + b[1] \* x[1]

else:

f = f\_Rosenbrock

def visualize():

x = np.arange (-50, 50, 0.5)

y = np.arange (-50, 50, 0.5)

xgrid, ygrid = np.meshgrid(x, y)

zgrid = f([xgrid, ygrid])

fig = pylab.figure()

axes = Axes3D(fig)

axes.plot\_surface(xgrid, ygrid, zgrid)

pylab.show()

return 0

visualize()

def grad(f,x):

if (f\_type == 'quadratic'):

return A.dot(x)+b

else:

I = np.eye(DIM)

return np.array([(f(x + H \* I\_vec) - f(x - H \* I\_vec)) / (2.0 \* (H @ I\_vec)) for I\_vec in I])

def hesse(f, x):

I = np.eye(DIM)

return np.array([(grad(f, x + H \* I\_vec) - grad(f, x - H \* I\_vec))/(2.0 \* (H @ I\_vec)) for I\_vec in I])

### FINDING ALPHA

def alpha\_quadr(f,x,step):

return -np.dot((A.dot(x)+b),step)/ (A.dot(step) @ step)

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) @ (step)):

alpha \*= l

return alpha

### FINDING BETA

def beta\_quadr(f,x\_prev,x\_next,step\_prev):

return (grad(f,x\_next) @ A.dot(step\_prev)) / (step\_prev @ A.dot(step\_prev))

def beta\_non\_quadr(f,x\_prev,x\_next,step\_prev):

return (grad(f,x\_next) @ grad(f,x\_next) - grad(f,x\_prev)) / (grad(f,x\_next) @ grad(f,x\_next))

### STOP CONDITIONS

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

return stop1(f,x1,x2,k) and stop2(f,x1,x2,k) and stop3(f,x2,k)

def stop1(f,x1,x2,k):

d = norm(x2-x1)

plt.xlabel('iteration')

plt.ylabel('|| x\_next - x\_prev || ')

if d>EPS:

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\_next) - f(x\_prev) | ')

if d>EPS:

plt.scatter(k, d)

return d<=EPS

def stop3(f,x,k):

plt.ylabel('|| f\'(x\_next) || ')

d = norm(grad(f,x), ord=DIM)

if d>EPS:

plt.scatter(k, d)

return d<=EPS

### OUTPUT

def output\_init(fout,x0,step):

print('The initial point is ({x}, {y})\n'.format(x=x0[0], y=x0[1]))

print(' k|'+' '\*8+'x'+' '\*9+'|'+' alpha '+ '|'+' '\*7+'step'+' '\*7)

fout.write('The initial point is ({x}, {y})\n\n'.format(x=x0[0], y=x0[1]))

fout.write(' k|'+' '\*16+'x'+' '\*17+'|'+' '\*5+'alpha'+' '\*5+ '|'+' '\*15+'step'+' '\*15+'\n')

#fout.write('{i:>3}| ({x:>14.11f}, {y:>14.11f}) |{al}|{st}\n'.format(i=0, x=x0[0], y=x0[1], al=' '\*15, st=' '\*34))

pass

def print\_inter(fout,k,alpha,x,step):

print('{i:>3}| ({x:>6.3f}, {y:>6.3f}) | {al:<5.3f} | ({st\_x:>6.3f}, {st\_y:>6.3f}) '

.format(i=k, x=x[0], y=x[1], al=alpha, st\_x = step[0], st\_y = step[1]))

pass

def output\_inter(fout,k,alpha,x,step):

fout.write('{i:>3}| ({x:>14.11f}, {y:>14.11f}) | {al:<13.11f} | ({st\_x:>14.11f}, {st\_y:>14.11f}) \n'

.format(i=k, x=x[0], y=x[1], al=alpha, st\_x = step[0], st\_y = step[1]))

pass

def output\_result(fout,k,x):

print('\nTOTAL ITERATIONS: {}'.format(k))

print('SOLUTION: x = ({x:>10.7f}, {y:>10.7f})'.format(x=x[0], y=x[1]))

print('VALUE: f(x) = {v:>10.7f}\n'.format(v=f(x)))

fout.write('\nTOTAL ITERATIONS: {}\n'.format(k))

fout.write('SOLUTION: x = ({x:>10.7f}, {y:>10.7f})\n'.format(x=x[0], y=x[1]))

fout.write('VALUE: f(x) = {v:>10.7f}\n'.format(v=f(x)))

Pass

def conjugate\_directions(f,x0):

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

k = -1

x\_next = x0

step\_next = - grad(f,x\_next)

flag = True

output\_init(fout,x0,step\_next)

find\_beta = beta\_quadr if f\_type == 'quadratic' else beta\_non\_quadr

find\_alpha = alpha\_quadr if f\_type == 'quadratic' else alpha\_split

while (k < MAX\_COUNT\_ITER):

x\_prev = x\_next

step\_prev = step\_next

alpha = find\_alpha(f,x\_prev,step\_prev)

x\_next = x\_prev + alpha \* step\_prev

beta = find\_beta(f,x\_prev,x\_next,step\_prev)

step\_next = - grad(f,x\_next) + beta \* step\_prev

k += 1

if k < 20:

print\_inter(fout,k,alpha,x\_prev,step\_prev)

else:

if flag == True:

print('...')

flag = False

output\_inter(fout,k,alpha,x\_prev,step\_prev)

if stop(f,x\_prev,x\_next,k) or (alpha == 0):

break

output\_result(fout,k,x\_next)

fout.close()

return x\_next

minim = conjugate\_directions(f,x0)

plt.xlabel('iteration')

plt.show()

print()

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

plt.show()

print()

**РЕЗУЛЬТАТИ РОБОТИ ПРОГРАМИ**

( реалізовано для H = [0.0001, 0.0001]; eps = 10\*\*(-5); x0 = [-2, -3] )

***-> Квадратична функція:***

***output.txt***

The initial point is (-2.0, -3.0)

k| x | alpha | step

0| (-2.00000000000, -3.00000000000) | 0.12738284743 | ( 4.00300000000, 25.00200000000)

1| (-1.49008646174, 0.18482595140) | 0.49064693691 | ( 3.03685572992, -0.12193277710)

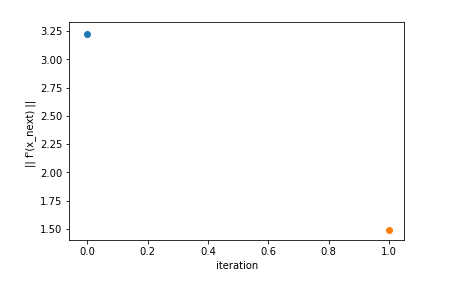
2| (-0.00006250000, 0.12500000781) | 0.12775415132 | ( 0.00000000000, 0.00000000000)

TOTAL ITERATIONS: 2

SOLUTION: x = (-0.0000625, 0.1250000)

VALUE: f(x) = -0.0625000

***on the screen***



***-> Функція Розенброка:***

***output.txt***

The initial point is (-2.0, -3.0)

k | x | alpha | step

0 | (-2.000, -3.000) | 0.125 | ( 4.003, 25.002)

1 | (-1.500, 0.125) | 0.000 | ( 8.784, 94.498)

2 | (-1.495, 0.171) | 0.000 | (14.675, 94.123) 3 | (-1.494, 0.183) | 0.000 | (22.467, 89.852) 4 | (-1.483, 0.227) | 0.000 | (32.532, 84.652)

5 | (-1.467, 0.268) | 0.001 | (45.198, 76.568) 6 | (-1.423, 0.343) | 0.002 | (59.959, 66.958) 7 | (-1.305, 0.474) | 0.001 | (74.263, 56.176) 8 | (-1.233, 0.528) | 0.004 | (88.485, 43.452) 9 | (-0.887, 0.698) | 0.001 | (99.285, 33.066) 10| (-0.790, 0.731) | 0.004 | (107.650, 22.381)

11| (-0.370, 0.818) | 0.000 | (113.827, 13.372)

12| (-0.314, 0.824) | 0.002 | (117.108, 5.438)

13| (-0.086, 0.835) | 0.001 | (119.553, -1.185)

14| ( 0.031, 0.834) | 0.001 | (120.122, -6.647)

15| ( 0.149, 0.827) | 0.001 | (119.584, -11.076)

16| ( 0.265, 0.817) | 0.001 | (117.900, -14.595)

17| ( 0.380, 0.802) | 0.000 | (115.046, -17.318)

18| ( 0.409, 0.798) | 0.000 | (111.275, -19.541)

19| ( 0.463, 0.789) | 0.000 | (107.215, -21.227)

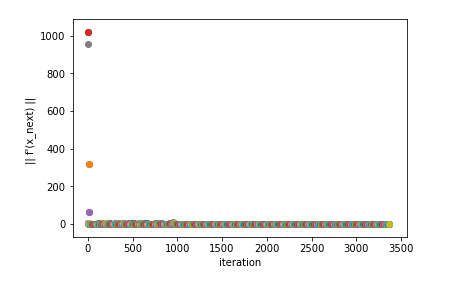
...

TOTAL ITERATIONS: 3368

SOLUTION: x = ( 1.0286766, 1.0556073)

VALUE: f(x) = 0.0014820

***on the screen***



**ВИСНОВКИ**

Під час виконання лабораторної роботи, я реалізувала метод спряжених градієнтів 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]) ]), похибка значення функції виявилася порядку *-10*. 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.