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 №2**

Тема: **«Чисельні методи безумовної оптимізації**

**другого порядку. Метод Ньютона та його варіації»**

Виконала: студентка 3-го курсу

групи КА-41

Лочман Я.В.

Київ – 2017

**ЗАВДАННЯ (ВАРІАНТ 6)**

Мінімізувати функцію одним з методів другого порядку (типу Ньютона).

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

**КОД ПРОГРАМИ**

import numpy as np

from numpy.linalg import norm

from numpy.linalg import inv

import matplotlib.pyplot as plt

import pylab

from mpl\_toolkits.mplot3d import Axes3D

### SETTINGS

MAX\_COUNT\_ITER = 100000

EPS = 10\*\*(-5)

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

x0 = np.array([-1, 2], dtype = float)

### TARGET FUNCTIONS

def f\_teacher(x):

return x[0]+x[1]

def f\_myvar(x):

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

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

#The value of function in this point is 2.5626251

def f\_Rosenbrock(x):

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

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

#The value of function in this point is 0

# DIM >2 #

#def f\_Powell(x):

# return (x[0] + 10.0 \* x[1])\*\*2 + 5.0 \* (x[2] - x[3])\*\*2 + (x[1] - 2.0 \* x[2])\*\*4 + 10.0 \* (x[0] - x[3])\*\*4

#def f\_hard(x):

# return 3.0 \* (x[0] - 1)\*\*2 + 2.0 \* (x[1] - 2)\*\*2 + (x[2] - 3)\*\*2

f\_type = 'quadratic'

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

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

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

f = f\_myvar

### APPROXIMATE DERIVATIVES

def df1(x,h1):

return (f([x[0] + h1, x[1]]) - f([x[0] - h1, x[1]])) / (2 \* h1)

def df2(x,h2):

return (f([x[0], x[1] + h2]) - f([x[0], x[1] - h2])) / (2 \* h2)

def df11(x,h1):

return (f([x[0] + h1, x[1]]) - 2 \* f([x[0], x[1]])+ f([x[0] - h1, x[1]])) / (h1 \* h1)

def df22(x,h2):

return (f([x[0], x[1] + h2]) - 2 \* f([x[0], x[1]])+ f([x[0], x[1] - h2])) / (h2 \* h2)

def df12(x,h1,h2):

return (f([x[0] + h1, x[1] + h2]) + f([x[0] - h1, x[1] - h2])

- f([x[0] + h1, x[1] - h2]) - f([x[0] - h1, x[1] + h2])) / (4 \* h1 \* h2)

def grad(x,h):

return np.array([df1(x,h[0]), df2(x,h[1])])

def gesse(x,h):

return np.array([ [df11(x,h[0]), df12(x,h[0],h[1])], [df12(x,h[0],h[1]), df22(x,h[1])] ] )

### STOP CONDITIONS

def stop1(x1,x2,k):

plt.ylabel('|| x\_k - x\_k-1 || ')

d = norm(x2-x1, ord=2)

plt.scatter(k, d)

return d<=EPS

def stop2(x1,x2,k):

plt.ylabel('| f(x\_k) - f(x\_k-1) | ')

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

plt.scatter(k, d)

return d<=EPS

def stop3(x,h,k):

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

d = norm(grad(x,h), ord=2)

plt.scatter(k, d)

return d<=EPS

### THE NEWTON METHOD

# функція повинна бути опуклою та двічі диференційовною на R^n, матриця Гессе - невироджена на R^n

# якщо функція сильно опукла та двічі диференційовна, то метод збігається незалежно від вибору x0 із надлін. швидк.

# якщо ще й до цього ||f''(x)-f''(y)||<=M||x-y||або квадр. швидк.

def newton\_method(x0,h):

print('NEWTON\'S METHOD:')

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

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

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

x\_new = x0

k = 0

plt.xlabel('iteration (k)')

while (k<MAX\_COUNT\_ITER):

x\_old = x\_new

matr = inv(gesse(x\_old,h))

step = - matr.dot(grad(x\_old,h))

x\_new = x\_old + step

k += 1

fout.write('{iter:>3}. x\_{iter:<3} = ({x:>18.15f}, {y:>18.15f})\n'.format(iter=k, x=x\_new[0], y=x\_new[1]))

if (stop1(x\_old,x\_new,k)): ### STOP CONDITION 1

#if (stop2(x\_old,x\_new,k)): ### STOP CONDITION 2

#if (stop3(x\_new,h,k)): ### STOP CONDITION 3

break

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

print('> Approximate x\* = ({x:>9.7f}, {y:>9.7f})'.format(x=x\_new[0], y=x\_new[1]))

print('> Approximate f(x\*) = {f:>10.7f}'.format(f=f(x\_new)))

fout.write('\nThe approximate solution of the problem is ({x:>10.7f}, {y:>10.7f})\n'.format(x=x\_new[0], y=x\_new[1]))

fout.write('The value of function in this point is {v:>10.7f}\n'.format(v=f(x\_new)))

fout.close()

return x\_new

### PROGRAM

minim = newton\_method(x0,h)

plt.show()

# функція повинна бути опуклою та двічі диференційовною на R^n, матриця Гессе - невироджена на R^n

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

alpha = b

#print('f(xk) = {0}, f(xk+1) = {1}, alpha = {2}'.format(f(x),f(x+alpha\*step),alpha))

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

alpha \*= l

#print('f(xk) = {0}, f(xk+1) = {1}, alpha = {2}'.format(f(x),f(x+alpha\*step),alpha))

return alpha

def newton\_method\_alpha(x0,h):

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

print('NEWTON\'S METHOD WITH STEP ADJUSTMENT:')

fout.write('NEWTON\'S METHOD WITH STEP ADJUSTMENT:\n')

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

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

x\_new = x0

k = 0

plt.xlabel('iteration (k)')

while (k<MAX\_COUNT\_ITER):

x\_old = x\_new

matr = inv(gesse(x\_old,h))

step = - matr.dot(grad(x\_old,h))

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

x\_new = x\_old + alpha \* step

k += 1

fout.write('{iter:>3}. x\_{iter:<3} = ({x:>18.15f}, {y:>18.15f})\n'.format(iter=k, x=x\_new[0], y=x\_new[1]))

if (stop1(x\_old,x\_new,k)): ### STOP CONDITION 1

#if (stop2(x\_old,x\_new,k)): ### STOP CONDITION 2

#if (stop3(x\_new,h,k)): ### STOP CONDITION 3

break

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

print('> Approximate x\* = ({x:>9.7f}, {y:>9.7f})'.format(x=x\_new[0], y=x\_new[1]))

print('> Approximate f(x\*) = {f:>10.7f}'.format(f=f(x\_new)))

fout.write('\nThe approximate solution of the problem is ({x:>10.7f}, {y:>10.7f})\n'.format(x=x\_new[0], y=x\_new[1]))

fout.write('The value of function in this point is {v:>10.7f}\n'.format(v=f(x\_new)))

fout.close()

return x\_new

### PROGRAM

minim = newton\_method\_alpha(x0,h)

plt.show()

**РЕЗУЛЬТАТИ РОБОТИ ПРОГРАМИ**

***Функція мого варіанта:***

***output.txt***

NEWTON'S METHOD:

The initial point is (-20.0, -20.0)

1. x\_1 = (-0.000058925500330, 0.125000368860498)

2. x\_2 = (-0.000062500003907, 0.125000007812501)

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

The value of function in this point is -0.0625000

NEWTON'S METHOD WITH STEP ADJUSTMENT:

The initial point is (-1.0, 2.0)

1. x\_1 = (-0.000062498233976, 0.125000007342426)

2. x\_2 = (-0.000062498676457, 0.125000007459945)

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

The value of function in this point is -0.0625000

***on the screen***

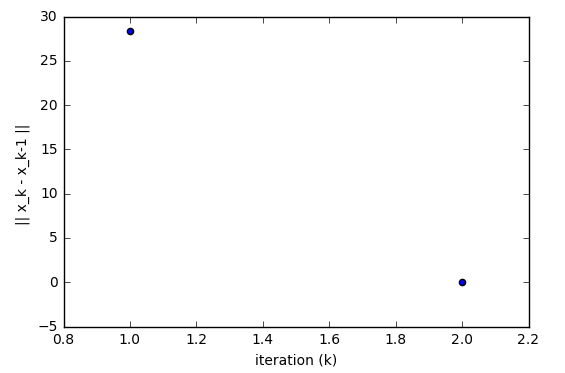
NEWTON'S METHOD:

The initial point is (-20.0, -20.0)

Approximate solution found in 2 iterations

> Approximate x\* = (-0.0000625, 0.1250000)

> Approximate f(x\*) = -0.0625000



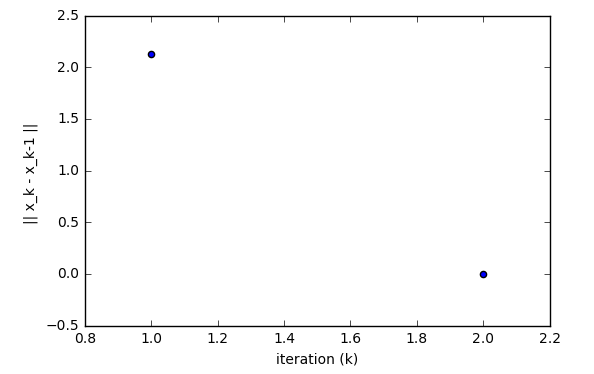
NEWTON'S METHOD WITH STEP ADJUSTMENT:

The initial point is (-1.0, 2.0)

Approximate solution found in 2 iterations

> Approximate x\* = (-0.0000625, 0.1250000)

> Approximate f(x\*) = -0.0625000



***Функція Розенброка:***

***output.txt***

NEWTON'S METHOD:

The initial point is (-1.0, 2.0)

1. x\_1 = (-1.010051261361092, 1.020102522685026)

2. x\_2 = ( 0.960195025437167, -2.959895943792009)

3. x\_3 = ( 0.960245983905069, 0.922072352396275)

4. x\_4 = ( 0.999804017076943, 0.998043234574766)

5. x\_5 = ( 0.999800987621081, 0.999602014838912)

6. x\_6 = ( 0.999800039897255, 0.999600119777655)

The approximate solution of the problem is ( 0.9998000, 0.9996001)

The value of function in this point is 0.0000000

NEWTON'S METHOD WITH STEP ADJUSTMENT:

The initial point is (-1.0, 2.0)

1. x\_1 = (-1.010051261361092, 1.020102522685026)

2. x\_2 = (-0.763770475511310, 0.522602714375396)

3. x\_3 = (-0.629617799268915, 0.378421632613329)

4. x\_4 = (-0.452452027302221, 0.164326655981825)

5. x\_5 = (-0.292433439831881, 0.059911368395896)

6. x\_6 = (-0.081286443554648, -0.037975568131034)

7. x\_7 = ( 0.027751998912440, -0.011119208491986)

8. x\_8 = ( 0.171661005267755, 0.002813007651586)

9. x\_9 = ( 0.302494182280730, 0.074385410106400)

10. x\_10 = ( 0.460160107841249, 0.186888780765503)

11. x\_11 = ( 0.550542750137730, 0.294928297700724)

12. x\_12 = ( 0.721143907997965, 0.490943780979285)

13. x\_13 = ( 0.762004453200100, 0.578981202542578)

14. x\_14 = ( 0.851149934878727, 0.715674502663244)

15. x\_15 = ( 0.905088978807755, 0.816276639099284)

16. x\_16 = ( 0.964969481952521, 0.927580426442847)

17. x\_17 = ( 0.985256469781043, 0.970318749370249)

18. x\_18 = ( 0.998695413441075, 0.997211923621542)

19. x\_19 = ( 0.999761641043262, 0.999522202060216)

20. x\_20 = ( 0.999800035102811, 0.999600108717479)

21. x\_21 = ( 0.999800039992528, 0.999600119969060)

The approximate solution of the problem is ( 0.9998000, 0.9996001)

The value of function in this point is 0.0000000

***on the screen***

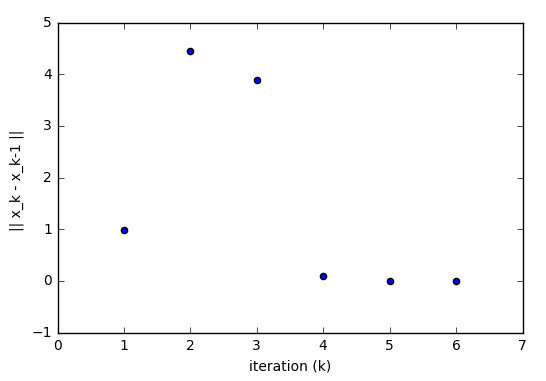
NEWTON'S METHOD:

The initial point is (-1.0, 2.0)

Approximate solution found in 6 iterations

> Approximate x\* = (0.9998000, 0.9996001)

> Approximate f(x\*) = 0.0000000



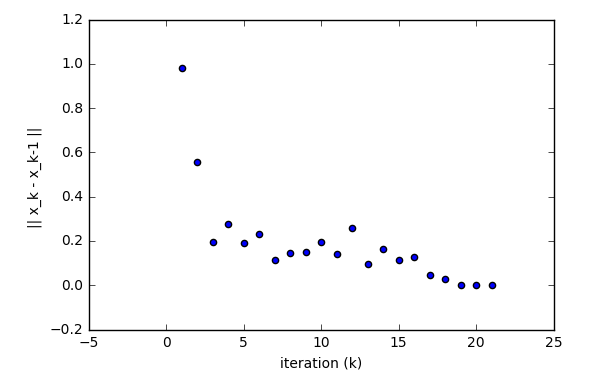
NEWTON'S METHOD WITH STEP ADJUSTMENT:

The initial point is (-1.0, 2.0)

Approximate solution found in 21 iterations

> Approximate x\* = (0.9998000, 0.9996001)

> Approximate f(x\*) = 0.0000000



**ВИСНОВКИ**

Під час виконання лабораторної роботи, я реалізувала метод другого порядку (Ньютона) для розв’язку оптимізаційної задачі а також метод Ньютона з регулюванням кроку.

Обидва методи показують достатньо високі результати, для моєї функції методи збігаються за дуже малу кількість кроків (до 10). Другий метод для функції Розенброка збігся за більшу кількість кроків (21) , ніж перший (6).

Також я побудувала графік, який наглядно показує, як змінюється норма різниці xk та xk+1. Також є можливість вибрати іншу з трьох умов зупинки.