**תרגיל בית 4 – מודלים לא לינאריים בחקר ביצועים**

**מגישים :**

**אלעד בוכריס – 206202426**

**משה דידי – 311395834**

שאלה 1 :

א.

def generic\_grad(f, gf, lsearch, x0, eps):  
 xk = x0  
 xk\_1 = xk - lsearch(f, xk, gf(xk)) \* gf(xk)  
 fs, gs, ts = [f(xk)], [np.linalg.norm(gf(xk))], [time.time()]  
 while np.abs(f(xk) - f(xk\_1)) > eps:  
 xk = xk\_1  
 xk\_1 = xk - lsearch(f, xk, gf(xk)) \* gf(xk)  
 fs.append(f(xk))  
 gs.append(np.linalg.norm(gf(xk)))  
 ts.append(time.time())  
 return xk, fs, gs, ts

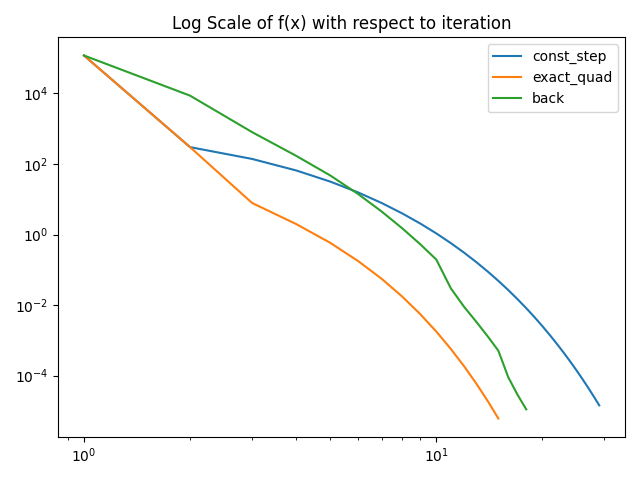
ב.

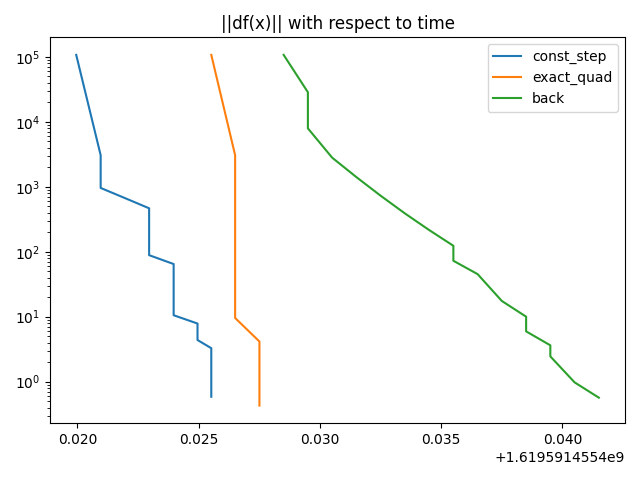
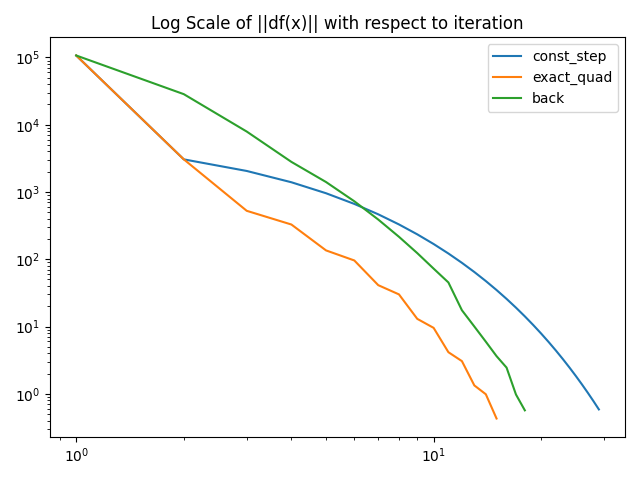
def const\_step(s):  
 return lambda f, xk, gk: s  
  
  
def exact\_quad(A):  
 ATA = np.dot(A.T, A)  
 np.linalg.cholesky(ATA)  
  
 def lsearch(f, xk, gk):  
 return 0.5 \* np.square(np.linalg.norm(gk)) / np.square(np.linalg.norm(np.dot(A, gk)))  
  
 return lsearch  
  
  
def back(alpha, beta, s):  
 def lsearch(f, xk, gk):  
 t = s  
 while f(xk - t \* gk) >= f(xk) - alpha \* t \* np.square(np.linalg.norm(gk)):  
 t \*= beta  
 return t  
  
 return lsearch

ג.

def f(x):  
 A = np.array([[100, 2, 3, 4, 5],  
 [6, 100, 8, 9, 10],  
 [11, 12, 100, 14, 15],  
 [16, 17, 18, 100, 20],  
 [21, 22, 23, 24, 100]])  
 return ((np.dot(x.T, A.T)).dot(A)).dot(x)  
  
  
def gf(x):  
 A = np.array([[100, 2, 3, 4, 5],  
 [6, 100, 8, 9, 10],  
 [11, 12, 100, 14, 15],  
 [16, 17, 18, 100, 20],  
 [21, 22, 23, 24, 100]])  
 return 2 \* ((np.dot(A.T, A)).dot(x))

def generic\_grad(f, gf, lsearch, x0, eps):  
 xk = x0  
 xk\_1 = xk - lsearch(f, xk, gf(xk)) \* gf(xk)  
 fs, gs, ts = [f(xk)], [np.linalg.norm(gf(xk))], [time.time()]  
 while np.abs(f(xk) - f(xk\_1)) > eps:  
 xk = xk\_1  
 xk\_1 = xk - lsearch(f, xk, gf(xk)) \* gf(xk)  
 fs.append(f(xk))  
 gs.append(np.linalg.norm(gf(xk)))  
 ts.append(time.time())  
 return xk, fs, gs, ts  
  
  
def q1():  
 A = np.array([[100, 2, 3, 4, 5],  
 [6, 100, 8, 9, 10],  
 [11, 12, 100, 14, 15],  
 [16, 17, 18, 100, 20],  
 [21, 22, 23, 24, 100]])  
 s = 1 / (2 \* np.max(np.linalg.eigvals(np.dot(A.T, A))))  
 x0 = np.array([1, 1, 1, 1, 1])  
 eps = 1 / np.power(10, 5)  
  
 \_const = generic\_grad(f, gf, const\_step(s), x0, eps)  
 \_exact = generic\_grad(f, gf, exact\_quad(A), x0, eps)  
 \_back = generic\_grad(f, gf, back(0.5, 0.5, 1), x0, eps)  
 plt.loglog(np.arange(1, len(\_const[1]) + 1), \_const[1], label="const\_step")  
 plt.loglog(np.arange(1, len(\_exact[1]) + 1), \_exact[1], label="exact\_quad")  
 plt.loglog(np.arange(1, len(\_back[1]) + 1), \_back[1], label="back")  
 plt.title("Log Scale of f(x) with respect to iteration")  
 plt.legend()  
 plt.show()  
  
 plt.loglog(np.arange(1, len(\_const[2]) + 1), \_const[2], label="const\_step")  
 plt.loglog(np.arange(1, len(\_exact[2]) + 1), \_exact[2], label="exact\_quad")  
 plt.loglog(np.arange(1, len(\_back[2]) + 1), \_back[2], label="back")  
 plt.title("Log Scale of ||df(x)|| with respect to iteration")  
 plt.legend()  
 plt.show()  
  
 plt.semilogy(\_const[3], \_const[2], label="const\_step")  
 plt.semilogy(\_exact[3], \_exact[2], label="exact\_quad")  
 plt.semilogy(\_back[3], \_back[2], label="back")  
  
 plt.title("||df(x)|| with respect to time")  
 plt.legend()  
 plt.show()

תרשימים:



שאלה 2 :

1. *כאשר :*

*- המרחק בין אובייקט i לאוביקט j.*

*נשים לב כי בעיית האופטימיזציה מוצאת ערכי כך שלכל מתקיים*

*כלומר, בעיית האופטימיזציה מנסה למצוא אשר קרובה ככל הניתן ל- .*

*ולכן היא מוצאת קונפיגורציה עבור כל אובייקט.*

1. *ראשית, ניתן לראות כי גזיר ולכן גזירה כסכום של פונקציות גזירות.*

*בתרגול, ראינו כי עבור כל פונקציה חסומה מלרע שיטת הגרדיאנט מתכנסת. ניתן לראות כי לאור העובדה ש - הינה סכום ריבועים מתקיים :*

*כלומר, חסומה מלרע ולכן סדרת הערכים מתכנסת.*

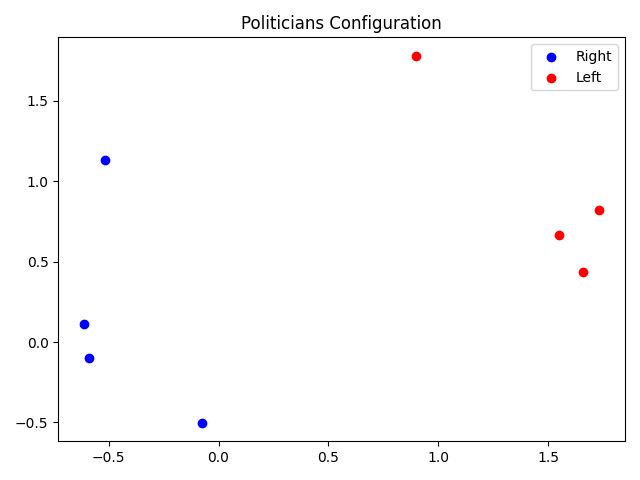
1. *נגדיר :*

*ו-*

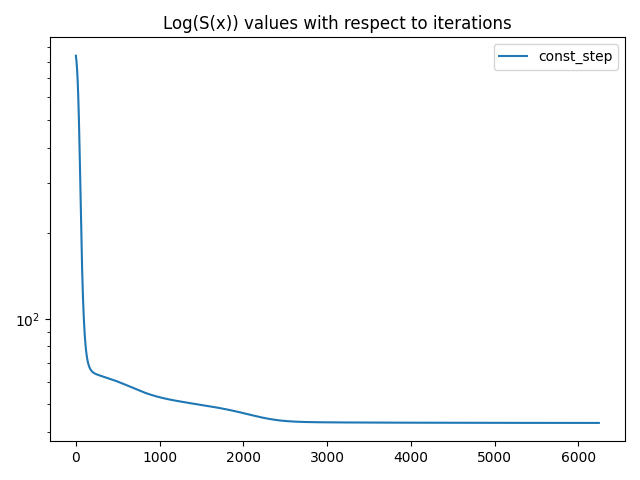
*ולכן :*

1. *בהינתן קונפיגורציה*

*אם נאתחל את שיטת הגרדיאנט עם קונפיגורציה זו אזי כל צעד שווה לקודמו ולכן השיטה תתכנס תמיד לאחר איטרציה אחת לאותה נקודה התחלתית. המחשה: .*

1. *קופיגורציית חברי הכנסת:*

*ניתן לראות כי קיים הבדל בין מפלגת הימין למפלגת השמאל.*

**

*השיטה התכנסה לאחר סדר גודל של 1000 איטרציות.*

*קוד שאלה 2:*

def S(X):  
 O = np.array([[1, 0, 0, -1, -1, -1, -1, -1, 0],  
 [0, 1, -1, -1, -1, -1, -1, -1, -1],  
 [1, 1, -1, -1, -1, -1, -1, -1, -1],  
 [0, 0, -1, 1, -1, -1, -1, -1, -1],  
 [-1, -1, -1, 1, 1, 1, 1, 1, 0],  
 [-1, -1, 1, 1, 1, 0, 1, 0, 1],  
 [-1, -1, 1, 1, 1, 1, 1, 1, 1],  
 [-1, -1, 1, 1, 0, 1, 1, 0, 0]])  
 res = 0  
 for i in range(len(X)):  
 for j in range(len(X)):  
 res += np.square(np.square(np.linalg.norm(X[i, :] - X[j, :])) - D(O[i, :], O[j, :]))  
 return res  
  
  
def gS(X):  
 O = np.array([[1, 0, 0, -1, -1, -1, -1, -1, 0],  
 [0, 1, -1, -1, -1, -1, -1, -1, -1],  
 [1, 1, -1, -1, -1, -1, -1, -1, -1],  
 [0, 0, -1, 1, -1, -1, -1, -1, -1],  
 [-1, -1, -1, 1, 1, 1, 1, 1, 0],  
 [-1, -1, 1, 1, 1, 0, 1, 0, 1],  
 [-1, -1, 1, 1, 1, 1, 1, 1, 1],  
 [-1, -1, 1, 1, 0, 1, 1, 0, 0]])  
 res = []  
 for i in range(len(X)):  
 res.append(  
 sum([(X[i, :] - X[j, :]) \*  
 (np.square(np.linalg.norm(X[i, :] - X[j, :])) - D(O[i, :], O[j, :])) for j in range(len(X))]))  
 return np.array(res)  
  
  
def D(o1, o2):  
 return np.linalg.norm(o1 - o2)  
  
  
def q2():  
 s = 1 / 1000  
 X = np.random.rand(8, 2)  
 eps = 1 / np.power(10, 5)  
  
 \_const = generic\_grad(S, gS, const\_step(s), X, eps)  
 plt.scatter(\_const[0][:, 0][:4], \_const[0][:, 1][:4], label="Right", color='blue')  
 plt.scatter(\_const[0][:, 0][4:], \_const[0][:, 1][4:], label="Left", color='red')  
 plt.title("Politicians Configuration")  
 plt.legend()  
 plt.show()  
  
 plt.semilogy(np.arange(len(\_const[1])), \_const[1], label="const\_step")  
 plt.title("Log(S(x)) values with respect to iterations")  
 plt.legend()  
 plt.show()

***שאלה 3 :***

1. *נקודות סטציונריות :*

*עבור נקודת אוכף מכיוון שההסיאן לא מוגדר.*

*עבור נקודות מינימום לוקלי ממש, מכיוון שההסיאן מוגדר חיובית.*

*ננסה לחסום את הפונקציה מלמטה :*

*ננסה לחסום על ידי ערך זה*

*ולכן פונקציה זו חסומה מלמטה ונקודות אלו הן מינימום גלובלי.*

1. *האיבר הכללי נתון על ידי :*

*מכיוון שידוע כי הסדרה*

*מונוטונית יורדת ממש, אזי כאשר k שואף לאינסוף ערכי הפונקציה שואפים לנקודה מינימום של לאור נקודת ההתחלה בה נקודת המינימום של החתך הזה של הפונקציה מתקבלת בנקודה .*

*לכן,* ***לא*** *נוכל להגיע לנקודות ונהיה חייבים להתכנס לנקודה .*

1. *פלטים:*

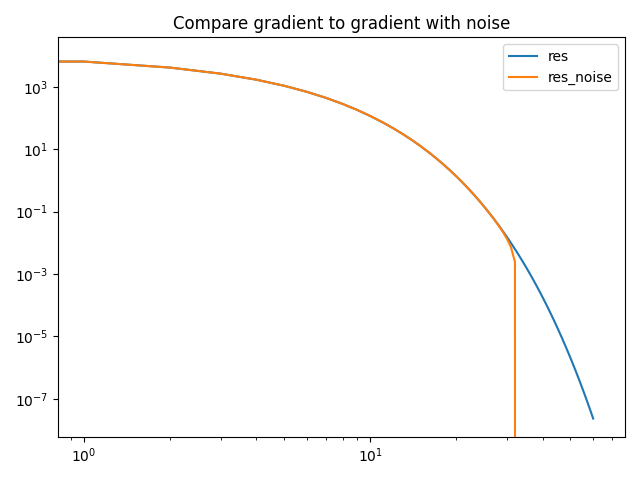
x: [0.00015325 0. ]

x\_noise: [7.84642227e-05 7.07061296e-01]

f(x): [10000, 6400.0, 4096.0, 2621.4400000000005, 1677.7216, 1073.741824, 687.19476736, 439.8046511104001, 281.4749767106561, 180.14398509481993, 115.29215046068474, 73.78697629483824, 47.22366482869648, 30.223145490365745, 19.342813113834072, 12.379400392853807, 7.922816251426436, 5.070602400912918, 3.245185536584268, 2.0769187434139313, 1.329227995784916, 0.8507059173023463, 0.5444517870735016, 0.34844914372704106, 0.22300745198530625, 0.14272476927059602, 0.09134385233318146, 0.05846006549323614, 0.03741444191567113, 0.023945242826029525, 0.015324955408658894, 0.009807971461541692, 0.006277101735386684, 0.0040173451106474784, 0.002571100870814386, 0.0016455045573212073, 0.0010531229166855726, 0.0006739986666787665, 0.0004313591466744106, 0.0002760698538716228, 0.0001766847064778386, 0.00011307821214581669, 7.237005577332267e-05, 4.631683569492652e-05, 2.9642774844752967e-05, 1.8971375900641897e-05, 1.2141680576410814e-05, 7.770675568902921e-06, 4.973232364097869e-06, 3.182868713022636e-06, 2.037035976334487e-06, 1.3037030248540719e-06, 8.343699359066062e-07, 5.339967589802278e-07, 3.4175792574734585e-07, 2.1872507247830136e-07, 1.3998404638611285e-07, 8.958978968711223e-08, 5.733746539975183e-08, 3.669597785584117e-08, 2.348542582773835e-08]

f(x\_noise): [10000, 6400.073569519582, 4096.047084454117, 2621.4701339953167, 1677.7408856773445, 1073.7541667187927, 687.2026665348478, 439.80970634444424, 281.4782117179285, 180.1460550062519, 115.29347449376193, 73.78782265326487, 47.22420502534368, 30.223489095473685, 19.343029967244586, 12.379534781513485, 7.9228959278044675, 5.070644275333603, 3.2451992059261063, 2.07690858479429, 1.329194269466719, 0.850645131390251, 0.5443564400370643, 0.3483068535613863, 0.22279938384788417, 0.14242317103610988, 0.09090838489551309, 0.05783245604535682, 0.03651077581950359, 0.02264492236386943, 0.013454969969783716, 0.007120626844719858, 0.002418750074308998, -0.0015149991898729962, -0.005346691023524981, -0.009656101055428717, -0.015017270596257909, -0.02205468902152854, -0.03147147132288892, -0.044032898195211785, -0.060474263053871574, -0.0812946697358274, -0.10642138987267839, -0.13481838111837474, -0.16427497547706144, -0.19172776950434794, -0.21426451755796067, -0.23033483497172427, -0.24024994280665773, -0.24558776459904794, -0.24814364621937834, -0.24925967472957977, -0.24971536845491116, -0.2498931107674762, -0.2499604146340919, -0.24998544007252876, -0.24999465011242814, -0.24999802455075618, -0.24999926151274127, -0.24999971760387227, -0.2499998879571338, -0.24999995306579847, -0.2499999788901266, -0.24999998970585385]

*גרף:*

**

*השיטה המורעשת התכנסה למינימום הגלובלי של בנקודה . אנו צפינו זאת, מכיוון שהוספת "הוציאה" את השיטה מהחתך ואפשרה לה להתקדם לכיוון המינימום הגלובלי של הפונקציה.*

*חסרון של השיטה המורעשת יכול להתבטא במספר רב יותר של איטרציות, מכיוון שאנו לא הולכים בדיוק נגד כיוון הגרדיאנט, אשר מביא לירידה מקסימלית. לא יכולנו לדעת לאיזו נקודה תתכנס השיטה, כי באופן דומה היא יכלה להתכנס לנקודה כפי שראינו בהרצות אחרות של הפונקציה.*

*קוד שאלה 3:*

def f\_q3(x):  
 return np.square(x[0]) + np.power(x[1], 4) - np.square(x[1])  
  
def df\_q3(x):  
 return np.array([2 \* x[0], 4 \* np.power(x[1], 3) - 2 \* x[1]])  
  
def generic\_grad\_noise(f, gf, lsearch, x0, eps, mu, sigma):  
 xk = x0  
 xk\_1 = xk - lsearch(f, xk, gf(xk)) \* gf(xk) + np.random.normal(loc=mu, scale=sigma, size=(len(xk)))  
 fs, gs, ts = [f(xk)], [np.linalg.norm(gf(xk))], [time.time()]  
 while np.abs(f(xk) - f(xk\_1)) > eps:  
 xk = xk\_1  
 xk\_1 = xk - lsearch(f, xk, gf(xk)) \* gf(xk)  
 fs.append(f(xk))  
 gs.append(np.linalg.norm(gf(xk)))  
 ts.append(time.time())  
 return xk, fs, gs, ts  
  
def ex3(mu, sigma, x0, epsilon):  
 x0 = np.array(x0)  
 res = generic\_grad(f\_q3, df\_q3, const\_step(1 / 10), x0, epsilon)  
 res\_noise = generic\_grad\_noise(f\_q3, df\_q3, const\_step(1 / 10), x0, epsilon, mu, sigma)  
 plt.loglog(np.arange(len(res[1])), res[1], label="res")  
 plt.loglog(np.arange(len(res\_noise[1])), res\_noise[1], label="res\_noise")  
 plt.title("Compare gradient to gradient with noise")  
 plt.legend()  
 plt.show()  
 return res[0], res\_noise[0], res[1], res\_noise[1]  
  
def q3():  
 res = ex3(0, 0.0005, [100, 0], 1 / np.power(10, 8))  
 print(f'x: {res[0]}\nx\_noise: {res[1]}\nf(x): {res[2]}\nf(x\_noise): {res[3]}\n')

***שאלה 4 :***

1. *יש להראות כי איטרציית הגראדינט שקולה לפתרון הבעיה :*



*לכן, זוהי נקודה סטציונרית.*

*כלומר, זוהי נקודת המינימום של הפונקציה. ולפי איטרציית הגרדיאנט היא שווה ל- .*

1. *באיטרצית הגרדיאנט מתקיים כי ולכן ההסיאן מוגדר חיובית.*
2. *יש להוכיח כי*

*ידוע כי*

*נתון כי*

*ולכן מתקיים :*

*נשים לב כי מתקיים*

*מש"ל.*