① a.

i. הפונק' מוגדרת לכל $x\in\mathbb{R}$, ברור כי $\mathbb{R}$ קמור, נבדוק קמירות בתחום:

$$f(x)=e^{ax} \longrightarrow f'(x)=ae^{ax} \longrightarrow f''(x)=a^2e^{ax}$$

$a^2\geq 0$, $e^{ax}\geq 0$ לכן $f''(x)\geq 0$ כלומר הפונק' קמורה.

ii. $f(x)=-\log(x)$, תחום ההגדרה הוא $\mathbb{R}^+$. נראה כי התחום קמור:

יהי $\alpha\in[0,1]$ ויהיו $x_1,x_2\in\mathbb{R}^+$, נראה כי $\alpha x_1+(1-\alpha)x_2\in\mathbb{R}^+$

$x_1>0$ וגם $x_2>0$, $0\leq\alpha\leq 1$ לכן $0<\alpha x_1+(1-\alpha)x_2$ כלומר בתוך $\mathbb{R}^+$.

נניח שמדובר בפונק' log לפי בסיס 10:

$$f''(x)=\frac{1}{\ln(10)x^2}>0$$

לכן הפונק' קמורה.

iii. ת.ה: $\mathbb{R}^+$ קמור.

$$f''(x)=-\frac{1}{\ln(10)x^2}<0$$

כלומר הפונק' קעורה.

iv. $a\geq 1$, $f(x)=|x|^a=\begin{cases}x^a & x\geq 0\\ (-x)^a & x<0\end{cases}$

ת.ה הוא $\mathbb{R}$, תחום קמור.

נחלק למקרים, כאשר $a=1$: $f(x)=|x|$ (זאת פונק' שאנחנו יודעים שהיא קמורה)

נראה קמירות לפי הגדרה: $f(\alpha x+(1-\alpha)y)\leq\alpha f(x)+(1-\alpha)f(y)$

$f(\alpha x + (1-\alpha)y) = |\alpha x + (1-\alpha)y| \le |\alpha x| + |(1-\alpha)y| = \alpha f(x) + (1-\alpha) f(y)$ ✓

כאשר $a>1$:

נבדוק את הפונק' גזירה:

$$\lim_{h\to 0^+} \frac{f(0+h)-f(0)}{h} = \lim_{h\to 0^+} h^{a-1} = 0$$

$$\lim_{h\to 0^-} \frac{f(0+h)-f(0)}{h} = \lim_{h\to 0^-} (-h)^{a-1} = 0$$

לכן f גזירה, וזו נגזרתה:

$$f'(x) = \begin{cases} -a(-x)^{a-1} & x<0 \\ ax^{a-1} & x\ge 0 \end{cases}$$

נראה שלכל $x,y\in\mathbb{R}$ מתקיים כי:
כך ש- $y>x$

$f(x) \ge f(y) + f'(y)(x-y)$

$f(x) - f(y) - f'(y)(x-y) \ge 0$

נגדיר: $h(x) = f(x)-f(y)-f'(y)(x-y)$, נראה ש- $h(x) \ge 0$

$$\frac{\partial h}{\partial x} = f'(x) - f'(y)$$

כאשר $x,y\ge 0$:

$h'(x) = f'(x) - f'(y) = ax^{a-1} - ay^{a-1} = a(x^{a-1} - y^{a-1})$

אז $h'(x) \le 0$ כלומר הפונק' יורדת בתחום הנ"ל

$h(y) = 0$, כלומר כאשר $x=y$ הפונק' מתאפסת, לכן $h(x) \ge 0$

דימון כנדרש.

כאשר $x, y \leq 0$:

$$h'(x) = f'(x) - f'(y) = -a(-x)^{a-1} + a(-y)^{a-1} = a((-y)^{a-1} - (-x)^{a-1})$$

שוב מתקיים כי $h'(x) \leq 0$ ו- $h(y) = 0$ לכן $h(x) \geq 0$.

כאשר $x \leq 0 \leq y$:

$$h'(x) = -a(-x)^{a-1} - ay^{a-1} = -a((-x)^{a-1} + y^{a-1})$$

בתחום הנ"ל $h'(x) \leq 0$

$$h(0) = f(0) - f(y) - f'(y)(-y) = 0 - y^{a} - ay^{a-1}(-y) = -y^{a} + ay^{a}$$
$$= y^{a}(a-1) \geq 0$$

לכן בתחום $h(x) \geq 0$.

כאשר $y < 0 < x$: לא מתקיים ע"פ ההנחה כי $y \geq x$.

לכן הראינו שלכל $x, y \in \mathbb{R}$ $f(x) \geq f(y) + f'(y)(x-y)$

כלומר הפונ' קמורה.

V. $f(x) = x^3$: ת.ה: $\mathbb{R}$ קמור

$$f''(x) = 6x$$

$$\left\{\begin{array}{ll} f''(x) \geq 0 & \text{כאשר } x \geq 0: \\ f''(x) < 0 & \text{כאשר } x < 0: \end{array}\right.$$

לכן הפונ' אינה קמורה או קעורה.

.6

$f(x)=x^TAx+b^Tx+c$

נראה כי פונק' זו קמורה:

כלומר נראה כי לכל $x,y\in R$ ולכל $\alpha\in[0,1]$ מתקיים:

$$f(\alpha x+(1-\alpha)y)\le \alpha f(x)+(1-\alpha)f(y)$$

$$f(\alpha x+(1-\alpha)y)=(\alpha x+(1-\alpha)y)^TA(\alpha x+(1-\alpha)y)+b^T(\alpha x+(1-\alpha)y)+c=$$

$$=\alpha^2x^TAx+(1-\alpha)\alpha y^TAx+\alpha(1-\alpha)x^TAy+(1-\alpha)^2y^TAy+\alpha b^Tx+(1-\alpha)b^Ty+c\overset{*}{=}$$

נניח כי A סימטרית

$$=\alpha^2x^TAx+2\alpha(1-\alpha)y^TAx+(1-\alpha)^2y^TAy+\alpha b^Tx+(1-\alpha)b^Ty+c$$

נעבור לפתח את הצד הימני של אי השוויון:

$$\alpha f(x)+(1-\alpha)f(y)=\alpha(x^TAx+b^Tx+c)+(1-\alpha)(y^TAy+b^Ty+c)=$$

$$\alpha x^TAx+\alpha b^Tx+\alpha c+(1-\alpha)y^TAy+(1-\alpha)b^Ty+(1-\alpha)c$$

כעת נחסר בין 2 הביטויים ונקבל:

$$\alpha f(x)+(1-\alpha)f(y)-f(\alpha x+(1-\alpha)y)=$$

$$=\alpha x^TAx+\alpha b^Tx+\alpha c+(1-\alpha)y^TAy+(1-\alpha)b^Ty+(1-\alpha)c-\alpha^2x^TAx-2\alpha(1-\alpha)y^TAx-$$
$$(1-\alpha)^2y^TAy-\alpha b^Tx-(1-\alpha)b^Ty-c=$$

$$=x^TAx(\alpha-\alpha^2)+y^TAy(\alpha-\alpha^2)-2\alpha(1-\alpha)y^TAx=$$

$$=x^TAx\alpha(1-\alpha)+y^TAy\alpha(1-\alpha)-2\alpha(1-\alpha)y^TAx=$$

$$=\alpha(1-\alpha)(x^TAx+y^TAy-2y^TAx)=\alpha(1-\alpha)\left((x-y)^TA(x-y)\right)$$

אנו יודעים כי $0\le\alpha\le1$ ולכן $\alpha(1-\alpha)\ge0$. נשים לב כי אגף ביטוי זה יהיה גדול/שווה מאפס כאשר A מטריצה SPD.

לכן הפונק' תהיה קמורה אמ"מ A סימטרית וגם SPD.

ג) צ"ל: f קמורה $\Longleftrightarrow$ $f(y) \geq f(x) + \nabla f(x)^T (y-x)$

$\Leftarrow$: f קמורה לכן מתקיים $f(\alpha x + (1-\alpha) y) \leq \alpha f(x) + (1-\alpha) f(y)$

$\alpha \in [0,1]$

$$f(\alpha x + y - \alpha y) \leq \alpha f(x) + f(y) - \alpha f(y)$$

$$f(\alpha(x-y)+y) \leq \alpha(f(x) - f(y)) + f(y)$$

(*) נשים לב כאשר $\alpha \to 0$ $\quad f(y+\alpha(x-y)) = f(y) + \alpha \nabla f(y)^T \cdot (x-y)$

לכן:

$$\cancel{f(y)} + \cancel{\alpha} \nabla f(y)^T \cdot (x-y) \leq \cancel{\alpha}(f(x) - f(y)) + \cancel{f(y)}$$

כנדרש $\quad f(y) + \nabla f(y)^T (x-y) \leq f(x)$

שים לב שכאן משתנה לסימון x,y

$\Longrightarrow$: נניח ש- $f(y) \geq f(x) + \nabla f(x)^T (y-x)$ $\quad \forall x,y \in \mathbb{R}$

תהי נק' $(1-\alpha)x + \alpha y = t$ נק' בין x ל-y

נקבל:

$$\begin{cases} f(x) \geq f(t) + \nabla f(t)^T (x-t) \\ f(y) \geq f(t) + \nabla f(t)^T (y-t) \end{cases}$$

$$\begin{cases} \alpha f(x) \geq \alpha f(t) + \alpha \nabla f(t)^T (x-t) \\ (1-\alpha) f(y) \geq (1-\alpha) f(t) + (1-\alpha) \nabla f(t)^T (y-t) \end{cases} +$$

$\alpha x - \alpha t + y - t - \alpha y + \alpha t$

$$\alpha f(x) + (1-\alpha) f(y) \geq (1-\alpha+\alpha) f(t) + \nabla f(t)^T (\alpha(x-t) + (1-\alpha)(y-t))$$

$$\alpha f(x) + (1-\alpha) f(y) \geq f(t) + \nabla f(t)^T (\alpha x + y - t - \alpha y)$$

נציב את t שבחרנו :

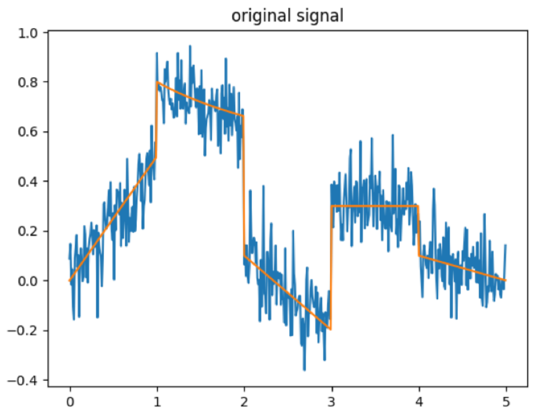
$$\alpha f(x)+(1-\alpha)f(y) \geq f(\alpha x+(1-\alpha)y)+\nabla f(\alpha x+(1-\alpha)y)^T\left(\cancel{\alpha x}+\cancel{y}-\cancel{\alpha x}-\cancel{(1-\alpha)y}-\cancel{\alpha y}\right)$$

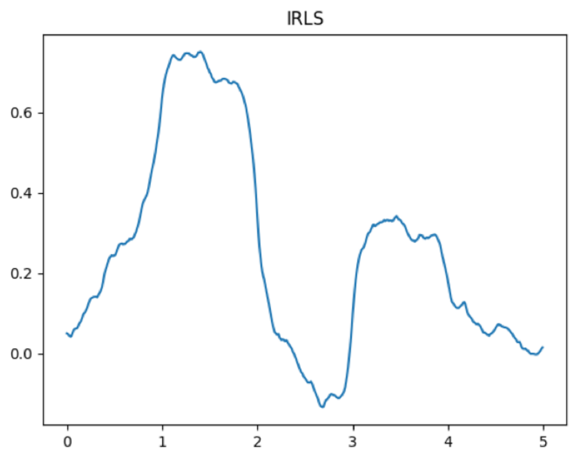
$$\alpha f(x)+(1-\alpha)f(y) \geq f(\alpha x+(1-\alpha)y)+\nabla f(\alpha x+(1-\alpha)y)^T\cdot 0$$

$$f(\alpha x+(1-\alpha)y) \leq \alpha f(x)+(1-\alpha)f(y)$$ כנדרש

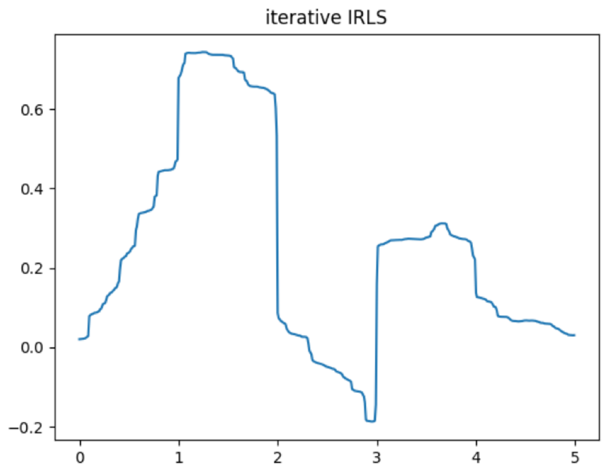
② הקוד מצורף, להלן הגרפים :

a.





b.



$$F(\theta) = \frac{1}{2}\|f(\theta)-y\|_2^2$$ .a ③

$$\nabla F(\theta) = \nabla\left(\frac{1}{2}\|f(\theta)-y\|_2^2\right) = \nabla\left(\frac{1}{2}((f(\theta)-y)^t\cdot(f(\theta)-y))\right)$$

$$= \frac{1}{2}2(f(\theta)-y)\cdot f'(\theta) = f'(\theta)\cdot(f(\theta)-y) = J^T(f(\theta)-y)$$

$$\underset{d}{\operatorname{argmin}} \frac{1}{2}\|f(\theta^K)+J(\theta^K)d-y\|_2^2$$ .b

$$\left(\frac{1}{2}\|f(\theta^K)+J(\theta^K)d-y\|_2^2\right)' = \frac{1}{2}\cdot 2\cdot J^T(\theta^K)(f(\theta^K)+J(\theta^K)d-y)$$

$$= J^T(\theta^K)\cdot f(\theta^K)+\underbrace{J^T(\theta^K)J(\theta^K)d}_{J^TJd}-J^T(\theta^K)y = 0$$

$$J^TJd = -J^T(\theta^K)\cdot f(\theta^K)+J^T(\theta^K)y = \underbrace{-J^T(f(\theta^K)+y)}_{\nabla F(\theta^K)}$$

$$J^TJd = -\nabla F(\theta^K)$$

$$d^k = \operatorname{argmin}\frac{1}{2}\|f(\theta^k)+J(\theta^k)d-y\|_2^2+\frac{\mu}{2}\|d\|_2^2$$ .c

$$\left(\frac{1}{2}\|f(\theta^k)+J(\theta^k)d-y\|_2^2+\frac{\mu}{2}\|d\|_2^2\right)' = J^T(\theta^k)(f(\theta^K)+J(\theta^K)d-y)+\mu d$$

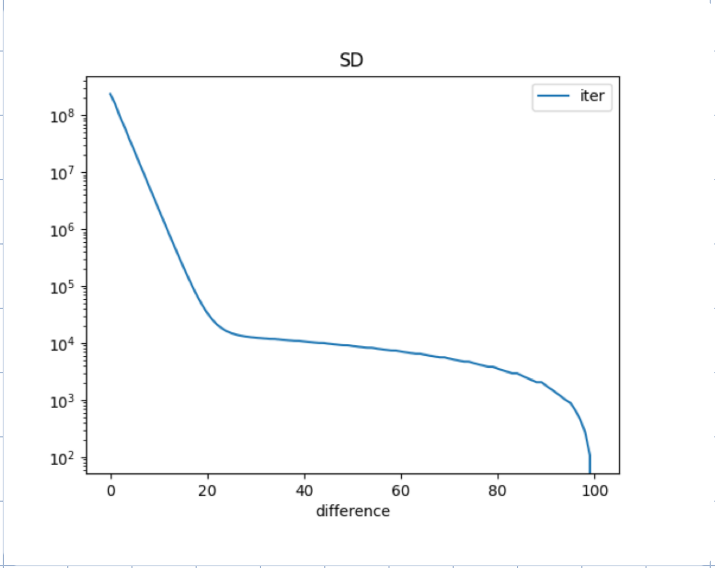
$$= J^T(\theta^K)\cdot f(\theta^K)+J^T(\theta^K)J(\theta^K)d-J^T(\theta^K)y+\mu d = 0$$

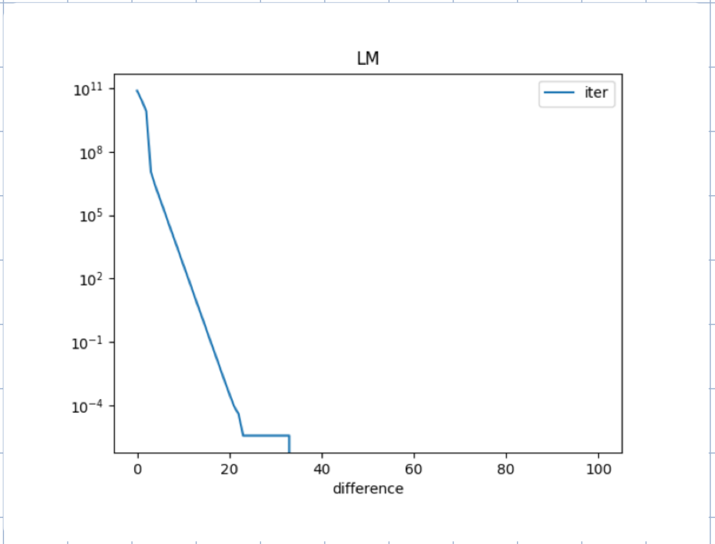
$$J^TJd+\mu d = -\nabla F(\theta)$$

$$d = -\underbrace{(J^TJ+\mu I)}_{\text{חיובית לכל סקלר }\mu}{}^{-1}\nabla F(\theta)$$

ג. נשים לב כי $\mu I + J^T J$ מטריצה חיובית ממש, לכן גם ההופכית שלה חיובית ממש. כיוון $\nabla F(\theta)$ הוא כיוון עליה גדול יותר לכן d הוא בכיוון מנוגד לגרדיאנט כלומר בכיוון ירידה.

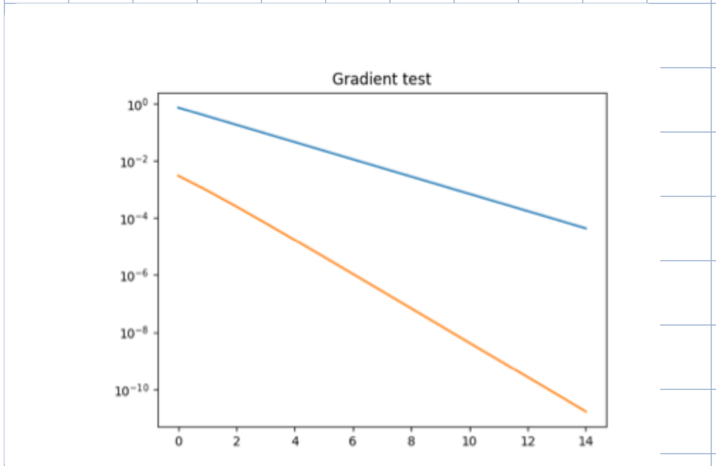
ש. קוד מצורף בהמשך, להלן הגרפים:

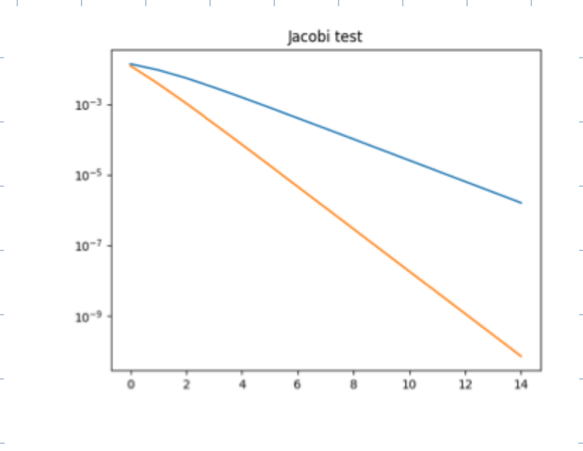




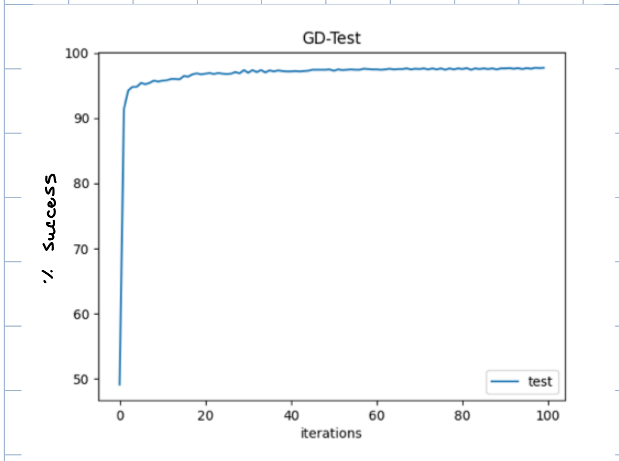
4 הקוד מצורף בהמשך, להלן הגרפים:

b.

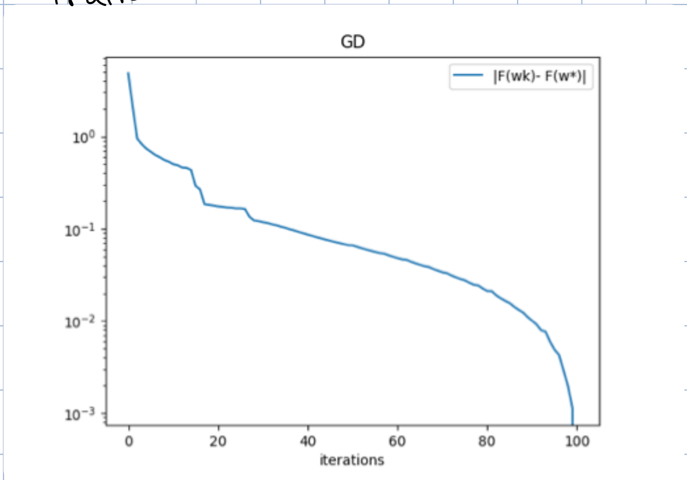


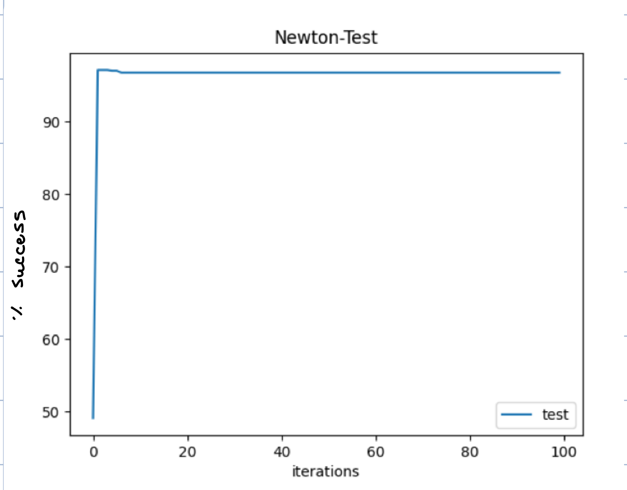


c. ק\ל:

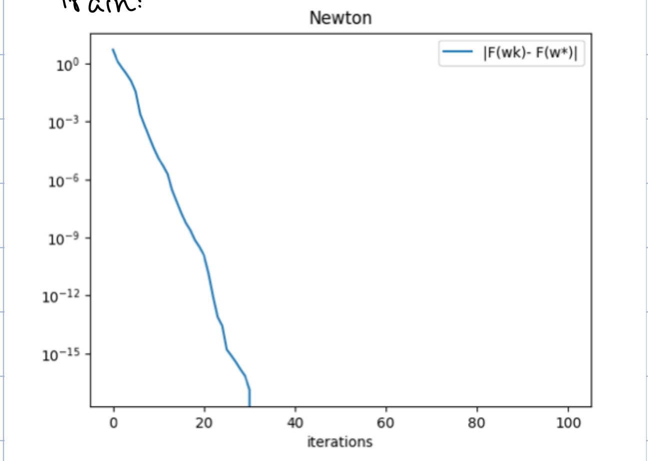


Train:

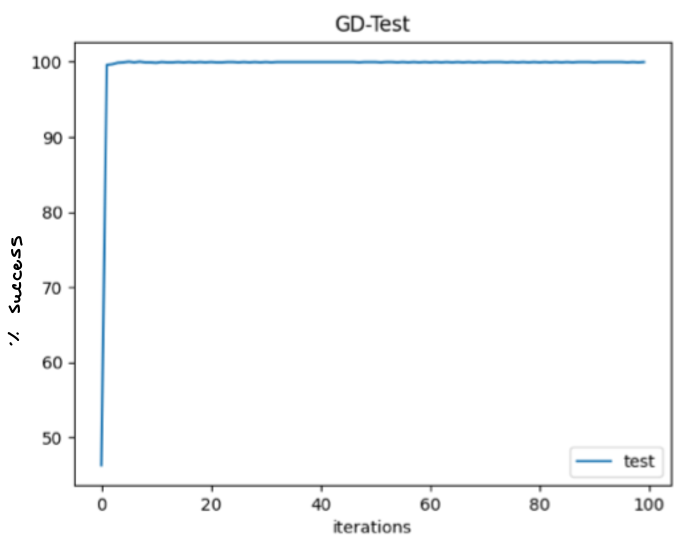




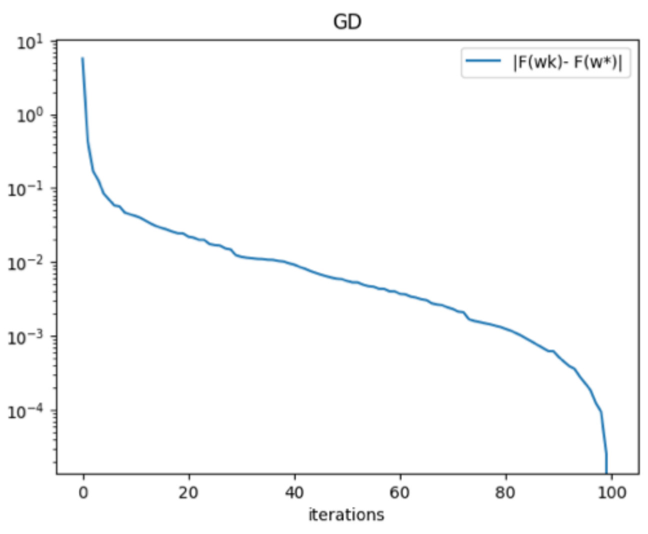
Train:

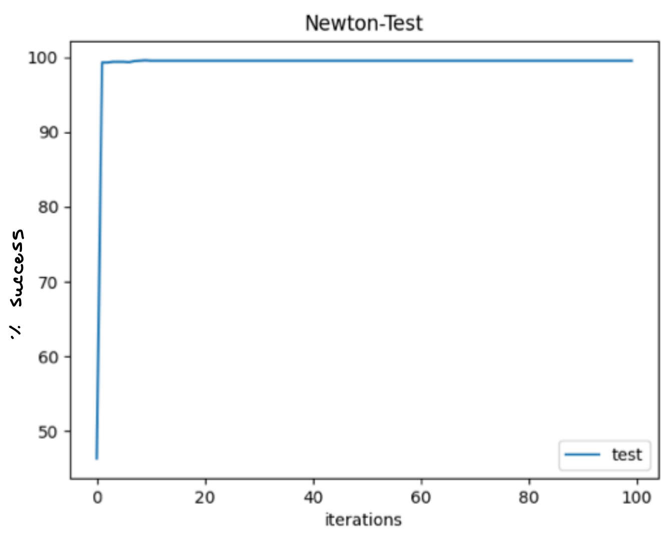


:0\1

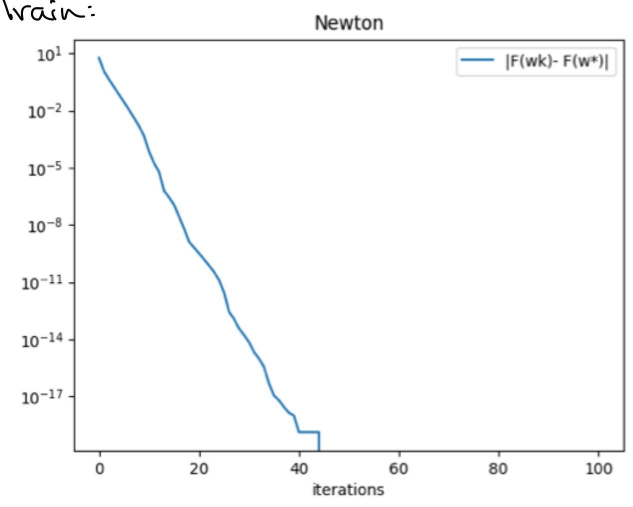


Train:





Train:



קוד מקור:

```
import numpy as np
import matplotlib.pyplot as plt
import inline as inline
import struct
from array import array
from scipy.sparse import spdiags
import numdifftools as nd

#2a
def IRLS(y, G, lamda=80):
    x = np.arange(0, 5, 0.01)
    sol = np.linalg.inv(2 * np.identity(G.shape[1]) + lamda *
np.transpose(G) @ G) @ y * 2
    plt.figure()
    plt.plot(x, np.transpose(sol[0]))
    plt.title("IRLS")
    plt.show()
    return sol

#2b
def IRLS_Iterative(y, G, lamda=1, epsilon=0.001):
    wk = np.identity(G.shape[0])
    xk = np.zeros(G.shape[1])
    for i in range(10):
        xk = np.linalg.inv(2 * np.identity(G.shape[1]) + lamda *
np.transpose(G) @ wk @ G) @ y * 2
        for j in range(G.shape[0]):
            wk[j][j] = 1 / (abs(G.getrow(j) @ xk) + epsilon)
    return xk

#q3

def readFile(fileName):
    fileObj = open(fileName, "r")  # opens the file in read mode
    words = fileObj.read().splitlines()  # puts the file into an
array
    fileObj.close()
    words = list(map(int, words))
    return words


def line_search(f, x, d, gk, alpha=1, beta=0.5, c=0.0000001, iter =
100):
    for j in range(iter):
        if (f(x + alpha * d)) <= f(x) + alpha * c * np.transpose(gk)
@ d:
            break
        else:
            alpha = alpha * beta
    return alpha


def SD():
    x = np.array(range(1, 100))
    grad1 = lambda theta: np.exp(-theta[1] * ((x - theta[2]) ** 2))
    grad2 = lambda theta: -theta[0] * (x - theta[2]) ** 2 * np.exp(-
theta[1] * ((x - theta[2]) ** 2))
    grad3 = lambda theta: theta[0] * 2 * theta[1] * (x - theta[2]) *
np.exp(-theta[1] * ((x - theta[2]) ** 2))
    gradient = lambda theta: [grad1(theta), grad2(theta),
grad3(theta)]
```

```
    y = np.array(readFile("Covid-19-USA.txt"))
    theta = np.array([1000000, 0.001, 110])
    f = lambda teta: teta[0] * np.exp(-teta[1] * (x - teta[2]) ** 2)
    bigF = lambda teta: 0.5*(np.linalg.norm(f(teta)-y)**2)
    residual = [bigF(theta)]
    for i in range(100):
        ft = f(theta)
        Jk = np.array(gradient(theta))
        gk = Jk @ (ft - y)
        alpha_SD = line_search(bigF, theta, (-1) * gk, gk)
        theta = theta - alpha_SD * gk
        residual.append(bigF(theta))
    for j in range(len(residual)):
        residual[j] = np.abs(residual[len(residual) - 1] -
residual[j])
    show_log_graph(residual,"|F(teta)- F(teta*)|","SD","iters")
    return residual

def LM():
    y = np.array(readFile("Covid-19-USA.txt"))
    theta = np.array([1000000, 0.001, 110])
    x = np.array(range(1, 100))
    f = lambda teta: teta[0] * np.exp(-teta[1] * (x - teta[2]) ** 2)
    bigF = lambda teta: 0.5 * (np.linalg.norm(f(teta) - y) ** 2)
    residual = [bigF(theta)];
    grad1 = lambda theta: np.exp(-theta[1] * ((x - theta[2]) ** 2))
    grad2 = lambda theta: -theta[0] * (x - theta[2]) ** 2 * np.exp(-
theta[1] * ((x - theta[2]) ** 2))
    grad3 = lambda theta: theta[0] * 2 * theta[1] * (x - theta[2]) *
np.exp(-theta[1] * ((x - theta[2]) ** 2))
    gradient = lambda theta: [grad1(theta), grad2(theta),
grad3(theta)]
    for i in range(100):
        jk = np.array(gradient(theta))
        mu = np.min(np.linalg.eigvals(jk @ np.transpose(jk))) + 0.004
        gk = jk @ (f(theta) - y)
        d = (-1) * np.linalg.inv(jk @ np.transpose(jk) +
mu*np.identity(jk.shape[0])) @ gk
        alpha = line_search(bigF, theta, d, gk)
        theta = theta + alpha*d
        residual.append(bigF(theta))
    for j in range(len(residual)):
        residual[j] = np.abs(residual[len(residual) - 1] -
residual[j])
    show_log_graph(residual,"|F(teta)- F(teta*)|","LM","iters")

    return residual


# q4

def LogiReg(X, labels, w):
    m = X.shape[0]
    sigmoid = lambda Xtw: 1 / (1 + np.exp(-Xtw))
    c1 = np.zeros(labels.shape[0])
    c2 = np.zeros(labels.shape[0])
    for i in range(len(labels)):
        if labels[i] == 0:
            c1[i] = 1
            c2[i] = 0
        else:
            c1[i] = 0
            c2[i] = 1
    objective = -(np.transpose(c1) @ np.log(sigmoid(np.transpose(X) @
w)) + np.transpose(c2) @ np.log(1 - sigmoid(np.transpose(X) @ w))) /
m
    gradient =  (X @ (sigmoid(np.transpose(X) @ w) - c1)) / m
    D = np.diag((sigmoid(np.transpose(X) @ w)) * (1 -
sigmoid(np.transpose(X) @ w)))
    hessian = (X @ D @ np.transpose(X)) / m
    return objective, gradient,hessian


def testGradient():
    rndLabel = np.array([1.] * 12 + [0.] * 13)
    rndMat = np.random.rand(25,25)
    rndVec = np.random.rand(25)
    d = np.random.random(25)
    eps = 0.25
    firstOrderTest = []
    secondOrderTest = []
    f0 = LogiReg(rndMat, rndLabel, rndVec)[0]
    g0 = LogiReg(rndMat, rndLabel, rndVec)[1]
    for i in range(15):
        epsilon = eps * 0.5 ** i
        f_1 = f0 + epsilon * np.transpose(g0) @ d
        fk = LogiReg(rndMat, rndLabel, rndVec + epsilon * d)[0]
        firstOrderTest.append(np.abs(fk-f0))
        secondOrderTest.append(np.abs(fk-f_1))

    plt.semilogy(firstOrderTest, label = "O(e)")
    plt.semilogy(secondOrderTest, label="O(e^2)")
    plt.title("Gradient test")
    plt.show()


def testJacobi():
    rndLabel = np.array([1.] * 12 + [0.] * 13)
    rndMat = np.random.rand(25, 25)
    rndVec = np.random.rand(25)
    d = np.random.random(25)
    eps = 0.25
    firstOrderTest = []
    secondOrderTest = []
    g0 = LogiReg(rndMat, rndLabel, rndVec)[1]
    h0 = LogiReg(rndMat, rndLabel, rndVec)[2]
    for i in range(15):
        epsilon = eps * 0.5 ** i
        gk = LogiReg(rndMat, rndLabel, rndVec + epsilon * d)[1]
        g1 = g0 + epsilon * h0 @ d
        firstOrderTest.append(np.linalg.norm(gk-g0))
        secondOrderTest.append(np.linalg.norm(gk-g1))
    plt.semilogy(firstOrderTest, label = "O(e)")
    plt.semilogy(secondOrderTest, label="O(e^2)")
    plt.title("Jacobi test")
    plt.show()


class MnistDataloader(object):
    def __init__(self, training_images_filepath,
training_labels_filepath,
                 test_images_filepath, test_labels_filepath):
```

```
        self.training_images_filepath = training_images_filepath
        self.training_labels_filepath = training_labels_filepath
        self.test_images_filepath = test_images_filepath
        self.test_labels_filepath = test_labels_filepath

    def read_images_labels(self, images_filepath, labels_filepath):
        labels = []
        with open(labels_filepath, 'rb') as file:
            magic, size = struct.unpack(">II", file.read(8))
            if magic != 2049:
                raise ValueError('Magic number mismatch, expected
2049, got {}'.format(magic))
            labels = array("B", file.read())

        with open(images_filepath, 'rb') as file:
            magic, size, rows, cols = struct.unpack(">IIII",
file.read(16))
            if magic != 2051:
                raise ValueError('Magic number mismatch, expected
2051, got {}'.format(magic))
            image_data = array("B", file.read())
        images = []
        for i in range(size):
            images.append([0] * rows * cols)
        for i in range(size):
            img = np.array(image_data[i * rows * cols:(i + 1) * rows
* cols])
            img = img.reshape(28, 28)
            images[i][:] = img

        return images, labels

    def load_data(self):
        x_train, y_train =
self.read_images_labels(self.training_images_filepath,
self.training_labels_filepath)
        x_test, y_test =
self.read_images_labels(self.test_images_filepath,
self.test_labels_filepath)
        return (x_train, y_train), (x_test, y_test)
    #


# Verify Reading Dataset via MnistDataloader class
#

inline
import random
import matplotlib.pyplot as plt

#
# Set file paths based on added MNIST Datasets
#

training_images_filepath = 'train-images.idx3-ubyte'
training_labels_filepath = 'train-labels.idx1-ubyte'
test_images_filepath = 't10k-images.idx3-ubyte'
test_labels_filepath = 't10k-labels.idx1-ubyte'



#
```

```
# Helper function to show a list of images with their relating titles
#
def show_images(images, title_texts):
    cols = 5
    rows = int(len(images) / cols) + 1
    plt.figure(figsize=(30, 20))
    index = 1
    for x in zip(images, title_texts):
        image = x[0]
        title_text = x[1]
        plt.subplot(rows, cols, index)
        plt.imshow(image, cmap=plt.cm.gray)
        if (title_text != ''):
            plt.title(title_text, fontsize=15);
        index += 1


#
# Load MINST dataset
#
mnist_dataloader = MnistDataloader(training_images_filepath,
training_labels_filepath, test_images_filepath,
                                    test_labels_filepath)
(x_train, y_train), (x_test, y_test) = mnist_dataloader.load_data()


def filter(images, labels):
    eight_nine_labels = []
    eight_nine_images = []
    zero_one_labels = []
    zero_one_images = []
    for i in range(30000):
        if labels[i] == 0 or labels[i] == 1:
            zero_one_labels.append(labels[i])
            zero_one_images.append(images[i])
        if labels[i] == 8 or labels[i] == 9:
            eight_nine_labels.append(labels[i])
            eight_nine_images.append(images[i])

    return zero_one_images, zero_one_labels, eight_nine_images,
eight_nine_labels


def create_cs(labels, num):
    c1 = np.zeros(len(labels))
    c2 = np.zeros(len(labels))
    n = len(labels)
    for i in range(n):
        if labels[i] == num:
            c1[i] = 1
            c2[i] = 0
        else:
            c1[i] = 0
            c2[i] = 1
    return c1, c2

def logistic_GD(data_2_train, c1, c2, test_data, test_labels, num):
    sig = lambda Xtw: 1 / (1 + np.exp(-Xtw))
    loss = lambda w: -(np.transpose(c1) @
np.log(sig(np.transpose(data) @ w)) + np.transpose(c2) @ np.log(
        1 - sig(np.transpose(data) @ w))) / m
    gradient = lambda w: (data @ (sig(np.transpose(data) @ w) - c1))
/ m
    rate = []
    data = []
    for i in range(len(data_2_train)):
        data.append(np.ndarray.flatten(data_2_train[i]))
    data = np.array(data)
    data = np.transpose(data)
    m = len(data)
    wk = np.zeros(784)
    rate.append(loss(wk))
    success_rate=[]
    for i in range(100):
        success_rate.append(test_w(wk, test_data, test_labels, num))
        prev =wk
        gk = gradient(wk)
        alpha_SD = line_search(loss, wk, -gk, gk)
        wk = wk - alpha_SD * gk
        fw = loss(wk)
        rate.append(fw)
        if np.linalg.norm(wk - prev)/np.linalg.norm(prev) < 0.002:
            break

    fw_star = fw
    error = []
    for x in rate:
        error.append(np.abs(x - fw_star))

    show_log_graph(error, '|F(wk)- F(w*)|', 'GD', 'iterations')
    show_graph_not_log(success_rate,"test","GD-Test", 'iterations')
    return wk


def logistic_Newton(data_2_train, c1, c2, test_data, test_labels,
num):
    sig = lambda Xtw: 1 / (1 + np.exp(-Xtw))
    loss = lambda w: -(np.transpose(c1) @
np.log(sig(np.transpose(data) @ w)) + np.transpose(c2) @ np.log(
        1 - sig(np.transpose(data) @ w))) / m
    gradient = lambda w: (data @ (sig(np.transpose(data) @ w) - c1))
/ m
    data = []
    for i in range(len(data_2_train)):
        data.append(np.ndarray.flatten(data_2_train[i]))
    data = np.array(data)
    data = np.transpose(data)
    rate = []
    m = len(data)

    wk = np.zeros(784)

    rate.append(loss(wk))
    hessian = lambda w : (data @ (np.diag((sig(np.transpose(data) @
w)) * (1 - sig(np.transpose(data) @ w)))) @ np.transpose(data)) / m
    success_rate=[]
    for i in range(100):
        success_rate.append(test_w(wk, test_data, test_labels, num))
        prev = wk
        hk = hessian(wk)
        gk = gradient(wk)
        epsilon = 0.004
        d = (-1) * np.linalg.inv(hk + epsilon*np.identity(784)) @ gk
        alpha_SD = line_search(loss, wk, d, gk)
        wk = wk + alpha_SD * d
        wk = np.clip(wk,-1,1)
        fw = loss(wk)
        rate.append(fw)

        if np.linalg.norm(wk - prev)/np.linalg.norm(prev) < 0.001:
            break
    w_star = wk
    fw_star = loss(w_star)
    error_vals = []
    for x in rate:
        error_vals.append(np.abs(x - fw_star))
    show_log_graph(error_vals, "|F(wk)- F(w*)|", "Newton",
"iterations")
    show_graph_not_log(success_rate, "test", "Newton-Test",
'iterations')

    return wk

def show_log_graph(arr,_label,title,xlabel):
    plt.semilogy(arr, label = _label)
    plt.xlabel(xlabel)
    plt.title(title)
    plt.legend()
    plt.show()

def show_graph_not_log(arr, _label, title, xlabel):
    plt.plot(arr, label=_label)
    plt.xlabel(xlabel)
    plt.title(title)
    plt.legend()
    plt.show()


def test_w(w, test_images, test_labels,digit):
    f = lambda x: x @ np.transpose(w)
    images = []
    approx_vec = []

    for i in range(len(test_images)):
        images.append(np.ndarray.flatten(test_images[i]))
    images = np.transpose(np.array(images))
    for i in range(images.shape[1]):
        if f(images[: ,i]) < 0:
            approx_vec.append(digit+1)
        else:
            approx_vec.append(digit)

    sum = 0
    for j in range(len(approx_vec)):
        if approx_vec[j] == test_labels[j]:
            sum += 1
    return (sum / len(approx_vec)) * 100

if __name__ == '__main__':
    # q2

    x = np.arange(0, 5, 0.01)
    n = np.size(x)
    one = int(n / 5)
    f = np.zeros(x.shape)
    f[0:one] = 0.0 + 0.5 * x[0:one]
    f[(one):2 * one] = 0.8 - 0.2 * np.log(x[100:200])
    f[(2 * one):3 * one] = 0.7 - 0.3 * x[(2 * one):3 * one]
    f[(3 * one):4 * one] = 0.3
    f[(4 * one):(5 * one)] = 0.5 - 0.1 * x[(4 * one):(5 * one)]
    G = spdiags([-np.ones(n), np.ones(n)], np.array([0, 1]), n - 1,
n)
    etta = 0.1 * np.random.randn(np.size(x))
    y = f + etta
    plt.figure();
    plt.plot(x, y);
    plt.title("original signal")
    plt.plot(x, f);
    plt.show()
    IRLS(y,G)

    #2b
    sol = IRLS_Iterative(y,G)
    plt.figure();
    plt.plot(x, sol);
    plt.title("iterative IRLS")
    plt.show()

    q3
    sd = SD()
    array= []
    for i in range(len(sd)):
        array.append(np.abs(sd[len(sd) - 1] - sd[i]))
    show_log_graph(array, "iter", "SD", "difference")

    lm = LM()
    array= []
    for i in range(len(lm)):
        array.append(np.abs(lm[len(lm) - 1] - lm[i]))
    show_log_graph(array, "iter", "LM", "difference")

# 4b

    testJacobi()
    testGradient()
# 4 c

    zero_one_images, zero_one_labels, eight_nine_images,
eight_nine_labels = filter(x_train,y_train,30000)
    zero_one_images_T, zero_one_labels_T, eight_nine_images_T,
eight_nine_labels_T = filter(x_test,y_test,10000)
    c1,c2 = create_cs(eight_nine_labels, 8)
    c3,c4 = create_cs(zero_one_labels, 0)

    w_star = logistic_GD(np.array(eight_nine_images), np.array(c1),
np.array(c2), np.array(eight_nine_images_T),
np.array(eight_nine_labels_T), 8)
    w_star = logistic_Newton(np.array(eight_nine_images),
np.array(c1), np.array(c2), np.array(eight_nine_images_T),
np.array(eight_nine_labels_T), 8)
    w_star = logistic_GD(np.array(zero_one_images), np.array(c3),
np.array(c4), np.array(zero_one_images_T),
np.array(zero_one_labels_T), 0)
    w_star = logistic_Newton(np.array(zero_one_images), np.array(c3),
np.array(c4), np.array(zero_one_images_T),
np.array(zero_one_labels_T), 0)
```