*שאלה 4:*

*ב. קוד:*

def ex5(s, p, lamda, eps):  
 A = create\_A(s.shape[0])  
 x, fv = grad\_proj(f=f(s, lamda, A),  
 gf=df(s, lamda, A),  
 proj=proj(p, eps=eps),  
 t=(1 / (1 + 4 \* lamda)),  
 x0=s,  
 eps=eps)  
 return x, fv

*שיטת החצייה, פונקציית grad\_proj ופונקציות עזר:*

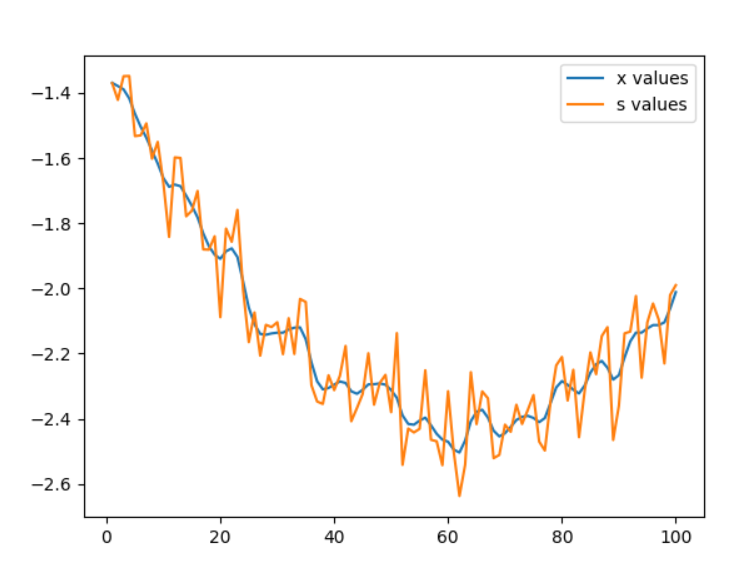
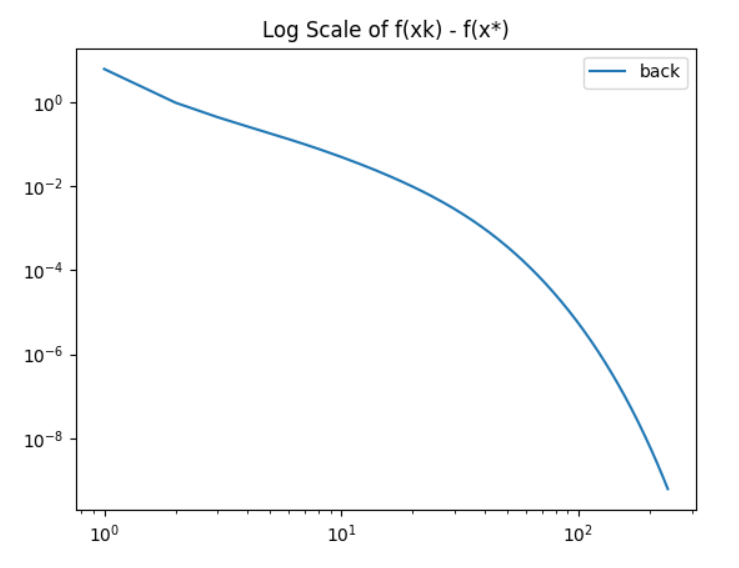
def generic\_bisect(f, l, u, eps):  
 *"""Running the generic bisect algorithm"""* x = (u + l) / 2  
 while np.abs(u - l) >= eps:  
 if f(u) \* f(x) > 0:  
 u = x  
 else:  
 l = x  
 x = (u + l) / 2  
 return x  
  
  
def grad\_proj(f, gf, proj, t, x0, eps):  
 fs = []  
 xk = x0  
 fs.append(f(x0))  
 xk\_1 = proj(xk - t \* gf(xk))  
 fs.append(f(xk\_1))  
 while np.linalg.norm(xk\_1 - xk) > eps:  
 xk = xk\_1  
 xk\_1 = proj(xk - t \* gf(xk))  
 fs.append(f(xk\_1))  
 return xk\_1, fs  
  
  
def f(s, lamda, A):  
 return lambda x: 0.5 \* np.square(np.linalg.norm(x - s)) + (0.5 \* lamda) \* np.square(np.linalg.norm(A.dot(x)))  
  
  
def df(s, lamda, A):  
 return lambda x: (x - s) + lamda \* A.T.dot(A.dot(x))  
  
  
def h(p, x):  
 return lambda mu: (x / (1 + 2 \* mu \* p)).T.dot(np.diag(p)).dot((x / (1 + 2 \* mu \* p))) - 1

def proj(p, eps):  
 def P(y):  
 if p.dot(np.square(y)) <= 1:  
 return y  
 else:  
 l, u = 1, 0  
 g = h(p, y)  
 while g(l) >= 0:  
 l \*= 2  
 mu = generic\_bisect(g, l, u, np.square(eps))  
 return y / (1 + 2 \* mu \* p)  
  
 return P  
  
  
def create\_A(n):  
 coef = [-0.5, 1, -0.5]  
 A = np.zeros((n, n))  
 for i in range(1, n - 1):  
 temp = np.concatenate((np.zeros(i - 1), coef, np.zeros(n - (i - 1) - 3)), axis=None)  
 A[i, :] = temp  
 return A

*ג. קוד:*

def cvxpy\_sol(s, p, lamda):  
 A = create\_A(s.shape[0])  
 x = cp.Variable(tuple([s.shape[0]]))  
 obj = cp.Minimize(0.5 \* cp.square(cp.norm(x - s)) + (0.5 \* lamda) \* cp.square(cp.norm(A @ x)))  
 constraints = [p.T @ cp.square(x) <= 1]  
  
 prob = cp.Problem(obj, constraints)  
 try:  
 result = prob.solve()  
 except SolverError:  
 result = prob.solve(solver=SCS)  
  
 return prob.value  
  
  
def q4\_b\_c():  
 s = np.transpose(  
 np.cos(np.linspace(0, 2, 100)) + 2 \* np.cos(np.linspace(3, 3, 100)) + np.cos(np.linspace(2, 5, 100)))  
 s = s + np.random.randn(\*s.shape) / 10  
 p = (1 / 1000) \* np.ones(100)  
 lam = 10  
 eps = 10 \*\* -6  
 x, fv = ex5(s.astype('float64'), p.astype('float64'), lam, eps)  
 f\_x\_star = cvxpy\_sol(s, p, lam)  
  
 plt.plot(np.arange(1, len(x) + 1), x, label="x values")  
 plt.plot(np.arange(1, len(s) + 1), s, label="s values")  
 plt.legend()  
 plt.show()  
  
 plt.loglog(np.arange(1, len(fv) + 1), np.array(fv) - f\_x\_star, label="back")  
 plt.title("Log Scale of f(xk) - f(x\*)")  
 plt.legend()  
 plt.show()

*הגרפים המתקבלים:*

**