# Nonlinear Optimization Fall 2021

## HW3 Sample Solutions

**Q1** (a) Consider the one-dimensional function

$$\delta_{[-1,1]}(x_i) = \begin{cases} 0 & \text{if } x_i \in [-1,1] \\ \infty & \text{otherwise.} \end{cases}$$

Then $\text{prox}_{\delta_{[-1,1]}}(x_i) = \text{proj}_{[-1,1]}(x_i)$

"orthogonal projection on $[-1,1]$"

$$= \text{argmin}\{\|z - x_i\|_2 \mid z \in [-1,1]\}$$

$$= \begin{cases} 1 & \text{if } x_i \geq 1 \\ -1 & \text{if } x_i \leq -1 \\ x_i & \text{otherwise.} \end{cases}$$

Then $f(x)$ seperates over its coordinates as $\sum_{i=1}^{d} \delta_{[-1,1]}(x_i)$.

$\Rightarrow$ By Lemma in lecture, $\text{prox}_f(x) = \text{prox}_{\delta_{[-1,1]}}(x_1) \times \dots \text{prox}_{\delta_{[-1,1]}}(x_d)$

$$= \begin{bmatrix} \text{proj}_{[-1,1]}(x_1) \\ \vdots \\ \text{proj}_{[-1,1]}(x_d) \end{bmatrix}$$

each of these is given by the piecewise formula above.

(b) This function also seperates over its coordinates.

It suffices to consider $\text{prox}_{\alpha|x|^3}(x_i)$ and apply this element-wise.

Noting $\alpha|x_i|^3 + \frac{1}{2}(x_i)^2$ is convex and differentiable it suffices to find the unique solution to

$$\left(\alpha|z|^3 + \frac{1}{2}(z - x_i)^2\right)' = 0$$

Letting $\text{sign}(x) = \begin{cases} +1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0, \end{cases}$ we have

$$\left(\alpha|z|^3 + \tfrac{1}{2}(z-x_i)^2\right)' = 3\alpha z^2 \text{sign}(z) + z - x_i = 0$$

First lets look at solutions $z > 0$. Then

The other root has $z < 0$.

$$3\alpha z^2 + z - x_i = 0 \iff z = \frac{-1 + \sqrt{1 + 4 \cdot 3\alpha x_i}}{6\alpha}$$

For any $x_i > 0$, this formula gives a positive $z$.

Hence this must be the unique value of $\text{prox}_{\alpha|z|^3}(x_i)$.
(when $x_i > 0$)

Now, if $z = 0$. Then $3\alpha z^2 + z - x_i = 0 \iff x_i = 0$.

Hence, for $x_i = 0$, $z = 0$ is the unique value of $\text{prox}_{\alpha|z|^3}(x_i)$.

Lastly, consider $z < 0$. Then

The other root has $z > 0$.

$$-3\alpha z^2 + z - x_i = 0 \iff z = \frac{-1 + \sqrt{1 - 4 \cdot 3\alpha x_i}}{-6\alpha} = \frac{1 - \sqrt{1 - 12\alpha x_i}}{6\alpha}.$$

For any $x_i < 0$, this formula gives a negative $z$.

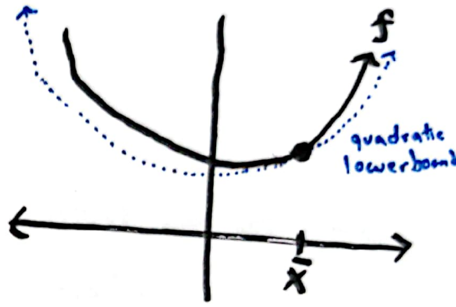
Hence this is the unique value of $\text{prox}_{\alpha|z|^3}(x_i)$.

$$\text{Then } \left[\text{prox}_f(x)\right]_i = \begin{cases} \dfrac{-1 + \sqrt{1 + 12\alpha x_i}}{6\alpha} & \text{if } x_i > 0 \\ 0 & \text{if } x_i = 0 \\ \dfrac{1 - \sqrt{1 - 12\alpha x_i}}{6\alpha} & \text{if } x_i < 0. \end{cases}$$

$\boxed{\text{Q2}}$ (a) By the equivalent characterization of strong convexity shown in lecture, we know for any $\bar{x}, x \in \mathbb{R}^d$

$$f(x) \geq \underbrace{f(\bar{x}) + \nabla f(\bar{x})^T(x-\bar{x}) + \frac{\mu}{2}\|x-\bar{x}\|_2^2}.$$

Then any $f(x) \leq f(\bar{x})$ has




$$f(\bar{x}) \geq f(\bar{x}) + \nabla f(\bar{x})^T(x-\bar{x}) + \frac{\mu}{2}\|x-\bar{x}\|_2^2$$

$$\Leftrightarrow \quad -\nabla f(\bar{x})^T(x-\bar{x}) \geq \frac{\mu}{2}\|x-\bar{x}\|_2^2.$$

By Cauchy-Schwarz, we conclude our distance bound as

$$\|\nabla f(\bar{x})\|_2 \|x-\bar{x}\|_2 \geq \frac{\mu}{2}\|x-\bar{x}\|_2^2$$

$$\Rightarrow \|x-\bar{x}\|_2 \leq \frac{2\|\nabla f(\bar{x})\|_2}{\mu}.$$

Noting that the minimum of $f$ is lower bounded by the minimum of our quadratic model we have

$$f(x) \geq \min_{y \in \mathbb{R}^d} f(y) \geq \min_y \left\{ f(\bar{x}) + \nabla f(\bar{x})^T(y-\bar{x}) + \frac{\mu}{2}\|y-\bar{x}\|_2^2 \right\}$$

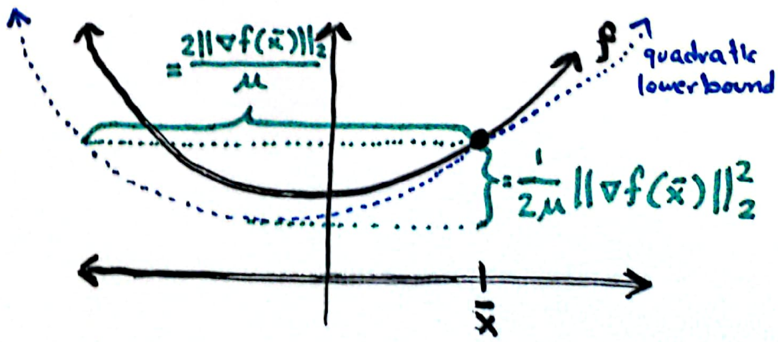
completing the square

$$= \min_y \left\{ f(\bar{x}) - \frac{1}{2\mu}\|\nabla f(\bar{x})\|_2^2 + \frac{\mu}{2}\left\|y - \left(\bar{x} - \frac{1}{\mu}\nabla f(\bar{x})\right)\right\|_2^2 \right\}$$

minimum at $y = \bar{x} - \frac{1}{\mu}\nabla f(\bar{x})$

$$= f(\bar{x}) - \frac{1}{2\mu}\|\nabla f(\bar{x})\|_2^2. \qquad \square$$

In terms of our picture, we bound the width and height of our quadratic as...



(b) Note (a) ensures $\inf_{x \in \mathbb{R}^d} f(x) \geq f(\bar{x}) - \frac{1}{2\mu} \|\nabla f(\bar{x})\|_2^2$ is bounded below. Further since every $x$ with $f(x) \leq f(\bar{x})$ lies in the compact ball $B = \{ x \mid \|x - \bar{x}\|_2 \leq \frac{2\|\nabla f(\bar{x})\|_2}{\mu} \}$, we can restrict our inf without changing its value to

$$\inf_{x \in B} f(x) = \inf_{x \in \mathbb{R}^d} f(x).$$

This inf is over a compact set, so it must be attained by some $x^*$. Hence a minimizer exists.

(c) Let $x^*$ be one such minimizer.

Suppose for contradiction another minimizer $y^* \neq x^*$ exists.

By HW2, Q3(a), $f(y^*) \geq f(x^*) + \frac{\mu}{2} \underbrace{\|y^* - x^*\|^2}_{>0 \text{ since } y^* \neq x^*}$

$> f(x^*)$

$\Rightarrow y^*$ is not a global minimizer since $x^*$ is better, contradicting our premise.

$\boxed{Q3}$ (a) First we show for any $i$, we have

$$g_i \in \partial(\max\{0, 1-y_i x_i^T w\})(w).$$

Namely, If $y_i x_i^T w > 1$, then we need every $v$ to have

$$\max\{0, 1-y_i x_i^T v\} \geq \underbrace{\max\{0, 1-y_i x_i^T w\}}_{\substack{= \\ 0 \\ \text{by assumption}}} + \underbrace{g_i^T(v-w)}_{\substack{= \\ 0 \\ \text{by definition}}}$$

$$= 0,$$

which is trivially the case.

If $y_i x_i^T w \leq 1$, then we need every $v$ to have

$$\max\{0, 1-y_i x_i^T v\} \geq \underbrace{\max\{0, 1-y_i x_i^T w\}}_{\substack{= \\ 1-y_i x_i^T w \\ \text{by assumption}}} + \underset{\substack{= \\ -y_i x_i \\ \text{by definition}}}{g_i^T}(v-w)$$

$$= 1-y_i x_i^T w + -y_i x_i^T(v-w)$$

$$= 1-y_i x_i^T v,$$

which is trivially the case.

Noting $\frac{\lambda}{2}\|w\|_2^2$ is convex with gradient $\lambda w$, we know for any $v$, we have the linear lower bound

$$\frac{\lambda}{2}\|v\|_2^2 \geq \frac{\lambda}{2}\|w\|^2 + \lambda w^T(v-w).$$

Summing this with our linear lowerbounds $g_i \in \partial(\max\{\cdot\}_i)(w)$

gives $$f(v) \geq f(w) + \sum_{i=1}^n g_i^T(v-w) + \lambda w^T(v-w)$$

$$= f(w) + (\sum_{i=1}^n g_i + \lambda w)^T(v-w).$$

$$\Rightarrow \sum g_i + \lambda w \in \partial f(w). \qquad \square$$

(b) To compute this prox step from $w=0$, we compute

$$\operatorname{argmin}\left\{ f(v) + \frac{1}{2\alpha}\|v - 0\|_2^2 \right\}$$

$$= \operatorname{argmin}\left\{ \sum_{i=1}^{n} \max\{0, 1 - y_i x_i^T v\} + \frac{\lambda}{2}\|v\|_2^2 + \frac{1}{2\alpha}\|v\|_2^2 \right\}$$

$$= \operatorname{argmin}\left\{ \sum_{i=1}^{n} \max\{0, 1 - y_i x_i^T v\} + \frac{(\lambda + 1/\alpha)}{2}\|v\|_2^2 \right\}.$$

This is itself a Support Vector Machine training problem with quadratic parameter $\lambda + 1/\alpha$ instead of the initial parameter $\lambda$.

(Therefore running an algorithm like the proximal point method here will need to solve a subproblem at each step that is as hard as the initial problem. Not every good...)

**Q4 (a)** Noting $\frac{1}{2}\|Ax - b\|_2^2$ is convex and differentiable, every $y \in \mathbb{R}^n$ must have

$$\frac{1}{2}\|Ay - b\|_2^2 \geq \frac{1}{2}\|Ax - b\|_2^2 + [A^T(Ax - b)]^T(y - x). \quad (*)$$

Recall from lecture that

$$\operatorname{sign}(x_i) = \begin{cases} 1 & \text{if } x_i > 0 \\ -1 & \text{otherwise} \end{cases} \in \partial|\cdot|(x_i).$$

$$\Rightarrow |y_i| \geq |x_i| + \operatorname{sign}(x_i)(y_i - x_i)$$

Summing this over $i = 1 \ldots n$

$$\|y\|_1 \geq \|x\|_1 + \operatorname{sign}(x)^T(y - x).$$

Adding this to $(*)$ gives the claim. $\square$

In [1]:
```python
import numpy as np
import matplotlib.pyplot as plt
import math

n = 1000
m = 100
A = np.random.normal(0, 1, size=(m, n)) #Draw normal random entries for A
b = np.random.normal(0, 1, size=m)      #Draw normal random entries for b

gamma = 2.0
```

In [2]:
```python
#Computing smoothness and strong convexity (eigenvalues of A^TA) for part (a)
eigv = np.linalg.eigvals(np.matmul(A.transpose(),A))
print("Lipschitz constant (Maximum Eigenvalue of A^TA): ", max(eigenvalues.real))
```

```
Lipschitz constant (Maximum Eigenvalue of A^TA):  1700.6906325735831
```

In [3]:
```python
def f(x):
    y=np.dot(A,x)-b
    return (np.linalg.norm(y)**2)/2 + gamma*np.linalg.norm(x,1)

def grad_LeastSquares(x):
    y=np.dot(A,x)-b
    return np.dot(A.transpose(),y)

def subgrad_l1(x):
    ret = np.zeros(n)
    for i in range(n):
        if x[i]<=0: ret[i]=-gamma
        else: ret[i] = gamma
    return ret

def subgrad_f(x):
    return grad_LeastSquares(x) + subgrad_l1(x)

def prox_l1(x):
    ret = np.zeros(n)
    for i in range(n):
        if x[i] > gamma/max(eigv.real): ret[i] = x[i] - gamma/max(eigv.real)
        if x[i] < -1.0*gamma/max(eigv.real): ret[i] = x[i] + gamma/max(eigv.real)
    return ret
```

In [4]:
```python
#Run subgradient descent for part (b)
x = np.zeros(n) #initialize at the origin
for i in range(100):
    x = x - subgrad_f(x)/max(eigv.real)
    print("After ", i, " steps, the objective value is ", f(x))
```

```
After  0  steps, the objective value is  16.35538155801829
After  1  steps, the objective value is  13.789289325008205
After  2  steps, the objective value is  12.924142946712387
After  3  steps, the objective value is  12.43790696141545
After  4  steps, the objective value is  12.125430441733165
After  5  steps, the objective value is  11.897254167861893
After  6  steps, the objective value is  11.6795928688299
After  7  steps, the objective value is  11.549318053720086
After  8  steps, the objective value is  11.384128051821437
After  9  steps, the objective value is  11.251373294306555
After  10  steps, the objective value is  11.15013565116805
After  11  steps, the objective value is  11.028923434740674
After  12  steps, the objective value is  10.986853477672803
After  13  steps, the objective value is  10.856764117471487
After  14  steps, the objective value is  10.821392284065464
After  15  steps, the objective value is  10.733000343890359
After  16  steps, the objective value is  10.675665401926846
After  17  steps, the objective value is  10.631327448028324
After  18  steps, the objective value is  10.52401410164006
After  19  steps, the objective value is  10.537580781911943
After  20  steps, the objective value is  10.44263935296106
After  21  steps, the objective value is  10.38739567046185
After  22  steps, the objective value is  10.391201951630654
After  23  steps, the objective value is  10.33872299258263
After  24  steps, the objective value is  10.301429781103478
After  25  steps, the objective value is  10.263966635524119
After  26  steps, the objective value is  10.211687062373247
After  27  steps, the objective value is  10.175007853121333
After  28  steps, the objective value is  10.161071097195498
After  29  steps, the objective value is  10.123288871182751
After  30  steps, the objective value is  10.077320971913423
After  31  steps, the objective value is  10.058653909575689
After  32  steps, the objective value is  10.057296439227803
After  33  steps, the objective value is  10.02928671737835
After  34  steps, the objective value is  10.012218167829754
After  35  steps, the objective value is  9.966450790409233
After  36  steps, the objective value is  9.99181633090638
After  37  steps, the objective value is  9.947035879323982
After  38  steps, the objective value is  9.935516137584376
After  39  steps, the objective value is  9.953634589255634
After  40  steps, the objective value is  9.896750250705185
After  41  steps, the objective value is  9.882417160705815
After  42  steps, the objective value is  9.857884086773202
After  43  steps, the objective value is  9.823591599064935
After  44  steps, the objective value is  9.838555171462138
After  45  steps, the objective value is  9.798309915093478
After  46  steps, the objective value is  9.799044227740866
After  47  steps, the objective value is  9.797717010867437
After  48  steps, the objective value is  9.80276467229169
After  49  steps, the objective value is  9.80024119013974
After  50  steps, the objective value is  9.796854255627391
After  51  steps, the objective value is  9.770311245379784
After  52  steps, the objective value is  9.733436914539228
After  53  steps, the objective value is  9.736732869564761
After  54  steps, the objective value is  9.70316292002403
```

```
After  55  steps, the objective value is  9.694278719686915
After  56  steps, the objective value is  9.722555887254803
After  57  steps, the objective value is  9.681136949188465
After  58  steps, the objective value is  9.674100784727042
After  59  steps, the objective value is  9.675866331194925
After  60  steps, the objective value is  9.634509948132033
After  61  steps, the objective value is  9.668840767688602
After  62  steps, the objective value is  9.636145751886136
After  63  steps, the objective value is  9.636315130860908
After  64  steps, the objective value is  9.629457637931415
After  65  steps, the objective value is  9.62786122577028
After  66  steps, the objective value is  9.607515917298729
After  67  steps, the objective value is  9.623605922299143
After  68  steps, the objective value is  9.631810682274883
After  69  steps, the objective value is  9.62005118821651
After  70  steps, the objective value is  9.552181474314468
After  71  steps, the objective value is  9.583499587830447
After  72  steps, the objective value is  9.589665702636895
After  73  steps, the objective value is  9.584880300067365
After  74  steps, the objective value is  9.587861730665434
After  75  steps, the objective value is  9.558680538006788
After  76  steps, the objective value is  9.594402194638413
After  77  steps, the objective value is  9.52763292916953
After  78  steps, the objective value is  9.566248512319383
After  79  steps, the objective value is  9.528815014409826
After  80  steps, the objective value is  9.537605292585864
After  81  steps, the objective value is  9.509078927602616
After  82  steps, the objective value is  9.532699928612717
After  83  steps, the objective value is  9.54130977430295
After  84  steps, the objective value is  9.508956659046461
After  85  steps, the objective value is  9.50706294524806
After  86  steps, the objective value is  9.514427330051522
After  87  steps, the objective value is  9.510429744204938
After  88  steps, the objective value is  9.494462869974251
After  89  steps, the objective value is  9.484961216742587
After  90  steps, the objective value is  9.49136995408602
After  91  steps, the objective value is  9.482905554617542
After  92  steps, the objective value is  9.443835467312118
After  93  steps, the objective value is  9.468330949429324
After  94  steps, the objective value is  9.46197879565526
After  95  steps, the objective value is  9.484475331510623
After  96  steps, the objective value is  9.454314213110994
After  97  steps, the objective value is  9.40661030363181
After  98  steps, the objective value is  9.41988281620666
After  99  steps, the objective value is  9.457909500215282
```

In [5]:
```
x #Checking x is not sparse
```

Out[5]:
```
array([-3.34493338e-04, -2.51105296e-04, -3.65849586e-04,  1.20650890e-04,
        1.69633169e-03,  4.28850984e-04,  1.25515347e-04,  7.65257841e-04,
        4.93530893e-05, -1.21100916e-03, -2.01228415e-04,  7.72224711e-04,
       -4.05535101e-04,  3.08645586e-04, -7.83026139e-04,  9.44108958e-04,
        4.84422883e-04,  4.28529379e-04, -8.39318302e-04, -1.78205575e-04,
        2.58385967e-04,  6.03138868e-04,  8.88605018e-04,  2.03675881e-03,
       -1.00716262e-04, -3.83702247e-04,  4.36749223e-04,  1.03594317e-03,
       -9.26840656e-05, -1.19572614e-02,  7.14303278e-04,  6.29441569e-04,
       -8.79853019e-04,  1.26687920e-02, -7.43887981e-04, -3.10286443e-04,
        7.84433987e-03,  8.25249169e-04,  6.96735258e-04, -1.23326236e-03,
       -9.11496549e-04, -1.03481325e-03, -1.45927191e-03, -3.94337671e-04,
        6.77918364e-04, -1.08264801e-03,  4.18840515e-04, -3.52296555e-04,
        1.22861827e-03,  7.77140726e-04,  1.97798070e-03, -4.36119200e-04,
        8.87394036e-04, -1.75873909e-03, -1.15874516e-03, -1.57510463e-03,
        3.57189597e-04, -4.37587580e-04, -1.06849665e-03, -1.44321104e-04,
        8.93652923e-04,  3.46912618e-03, -3.54405677e-04,  9.56454503e-04,
        5.11683207e-02,  4.46368266e-03,  9.70260536e-04, -1.98206267e-04,
       -6.90067298e-04, -1.74027935e-03, -7.23373447e-02,  1.42549299e-02,
        8.50574923e-04,  2.06474613e-04, -8.21194817e-04,  2.77435496e-04,
```

In [6]:
```
#Run proximal gradient descent for part (c)
x = np.zeros(n) #initialize at the origin
for i in range(100):
    x = prox_l1(x - grad_LeastSquares(x)/max(eigv.real))
    print("After ", i, " steps, the objective value is ", f(x))
```

```
After  0  steps, the objective value is  17.341726809410208
After  1  steps, the objective value is  13.573150391462642
After  2  steps, the objective value is  12.420852580518057
After  3  steps, the objective value is  11.865882137807564
After  4  steps, the objective value is  11.513030029704858
After  5  steps, the objective value is  11.249583949247263
After  6  steps, the objective value is  11.041878682728326
After  7  steps, the objective value is  10.869582764997789
After  8  steps, the objective value is  10.717925897379528
After  9  steps, the objective value is  10.583979961208593
After  10  steps, the objective value is  10.463156573314867
After  11  steps, the objective value is  10.35669107561181
After  12  steps, the objective value is  10.261440891425476
After  13  steps, the objective value is  10.17510420629387
After  14  steps, the objective value is  10.095559065627992
After  15  steps, the objective value is  10.022642993656302
After  16  steps, the objective value is  9.955369033475343
After  17  steps, the objective value is  9.892504760055491
After  18  steps, the objective value is  9.832878675582675
After  19  steps, the objective value is  9.776498858398394
After  20  steps, the objective value is  9.722834072770565
After  21  steps, the objective value is  9.672369197151877
After  22  steps, the objective value is  9.624960282055778
After  23  steps, the objective value is  9.580355723929674
After  24  steps, the objective value is  9.538046946624274
After  25  steps, the objective value is  9.498856498782095
After  26  steps, the objective value is  9.46149972268259
After  27  steps, the objective value is  9.426449650339945
After  28  steps, the objective value is  9.392868450187148
After  29  steps, the objective value is  9.36180215303147
After  30  steps, the objective value is  9.332615288208048
After  31  steps, the objective value is  9.304681518847733
After  32  steps, the objective value is  9.278102687558235
After  33  steps, the objective value is  9.252883073094935
After  34  steps, the objective value is  9.228364445696036
After  35  steps, the objective value is  9.20440649830248
After  36  steps, the objective value is  9.181165866556968
After  37  steps, the objective value is  9.158960115212407
After  38  steps, the objective value is  9.137603895456095
After  39  steps, the objective value is  9.117808862283695
After  40  steps, the objective value is  9.099151100921535
After  41  steps, the objective value is  9.081009388566454
After  42  steps, the objective value is  9.063658182774224
After  43  steps, the objective value is  9.04702164661255
After  44  steps, the objective value is  9.031039894686096
After  45  steps, the objective value is  9.015727665014856
After  46  steps, the objective value is  9.000999982250299
After  47  steps, the objective value is  8.986886131155979
After  48  steps, the objective value is  8.973282939780422
After  49  steps, the objective value is  8.960104249476304
After  50  steps, the objective value is  8.94733182350712
After  51  steps, the objective value is  8.934856356994832
After  52  steps, the objective value is  8.92286510350678
After  53  steps, the objective value is  8.911452056238842
After  54  steps, the objective value is  8.900400235948638
```

```
After  55  steps, the objective value is  8.889493507411999
After  56  steps, the objective value is  8.878932037811323
After  57  steps, the objective value is  8.868715840376252
After  58  steps, the objective value is  8.858892149496258
After  59  steps, the objective value is  8.849292062223352
After  60  steps, the objective value is  8.839960450158031
After  61  steps, the objective value is  8.830983458790275
After  62  steps, the objective value is  8.822185346883423
After  63  steps, the objective value is  8.81349387637228
After  64  steps, the objective value is  8.805002754754682
After  65  steps, the objective value is  8.796747682112466
After  66  steps, the objective value is  8.78879126239454
After  67  steps, the objective value is  8.780916863356557
After  68  steps, the objective value is  8.773152647102346
After  69  steps, the objective value is  8.765536072893157
After  70  steps, the objective value is  8.758022296648011
After  71  steps, the objective value is  8.750648672836988
After  72  steps, the objective value is  8.743495809808579
After  73  steps, the objective value is  8.736487726342965
After  74  steps, the objective value is  8.729531536194562
After  75  steps, the objective value is  8.722620335483033
After  76  steps, the objective value is  8.7157862017468
After  77  steps, the objective value is  8.709019273592304
After  78  steps, the objective value is  8.702329862186192
After  79  steps, the objective value is  8.695860605829438
After  80  steps, the objective value is  8.689531801769368
After  81  steps, the objective value is  8.683255993786567
After  82  steps, the objective value is  8.677078386890122
After  83  steps, the objective value is  8.670976775783892
After  84  steps, the objective value is  8.66498372030557
After  85  steps, the objective value is  8.659056262535852
After  86  steps, the objective value is  8.653237371730553
After  87  steps, the objective value is  8.647498974292313
After  88  steps, the objective value is  8.641883245166483
After  89  steps, the objective value is  8.63633198879228
After  90  steps, the objective value is  8.630852103923516
After  91  steps, the objective value is  8.625411359958388
After  92  steps, the objective value is  8.620023182458304
After  93  steps, the objective value is  8.614706717629602
After  94  steps, the objective value is  8.609536433063116
After  95  steps, the objective value is  8.604398894431254
After  96  steps, the objective value is  8.599304049683251
After  97  steps, the objective value is  8.594425111048299
After  98  steps, the objective value is  8.5896251747752
After  99  steps, the objective value is  8.584862165365044
```

In [7]:
```
x #Checking x is mostly sparse
```

Out[7]:
```
array([ 0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        , -0.01442554,
        0.        ,  0.        ,  0.        ,  0.0147894 ,  0.        ,
        0.        ,  0.00991835,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.00406378,  0.        ,  0.        ,  0.05197149,
        0.0044958 ,  0.        ,  0.        ,  0.        ,  0.        ,
       -0.07515872,  0.01608296,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.01326881, -0.02437293,  0.        ,
        0.        , -0.01940207, -0.03269631,  0.        ,  0.        ,
        0.01318797,  0.        ,  0.        ,  0.        ,  0.        ,
       -0.00369789, -0.06015729,  0.        ,  0.        , -0.01322898,
```

In [8]:

```python
#Run accelerated proximal gradient descent for part (d)
x = np.zeros(n) #initialize at the origin
y = np.zeros(n)       #auxillary sequence for the accelerated method
y_prev = np.zeros(n)  #tracking previous iterations value of y_prev
lam = 0
lam_prev=lam
for i in range(100):
    lam_prev = lam
    lam = (1+math.sqrt(1+4*(lam**2)))/2
    y = prox_l1(x - grad_LeastSquares(x)/max(eigv.real))
    x = y + (lam_prev-1.0)/lam*(y-y_prev)
    y_prev = y
    print("After ", i, " steps, the objective value is ", f(x))
```

```
After  0  steps, the objective value is  42.78069198762608
After  1  steps, the objective value is  17.341726809410208
After  2  steps, the objective value is  13.005000447035945
After  3  steps, the objective value is  11.904280434166054
After  4  steps, the objective value is  11.366293185176083
After  5  steps, the objective value is  10.987841680388396
After  6  steps, the objective value is  10.650162201899072
After  7  steps, the objective value is  10.385290492127242
After  8  steps, the objective value is  10.144077420388562
After  9  steps, the objective value is  9.933019637713532
After  10  steps, the objective value is  9.735345334833918
After  11  steps, the objective value is  9.591408320851256
After  12  steps, the objective value is  9.439021277841391
After  13  steps, the objective value is  9.326331576135379
After  14  steps, the objective value is  9.203899836472162
After  15  steps, the objective value is  9.127925184053147
After  16  steps, the objective value is  9.04542174623059
After  17  steps, the objective value is  8.977074797221753
After  18  steps, the objective value is  8.91402272755401
After  19  steps, the objective value is  8.844312757106644
After  20  steps, the objective value is  8.798640259045047
After  21  steps, the objective value is  8.751155009204966
After  22  steps, the objective value is  8.710013251615896
After  23  steps, the objective value is  8.657859408195728
After  24  steps, the objective value is  8.622862656344891
After  25  steps, the objective value is  8.597147136689882
After  26  steps, the objective value is  8.5569667358078
After  27  steps, the objective value is  8.520655556233379
After  28  steps, the objective value is  8.491919858247703
After  29  steps, the objective value is  8.487755443830485
After  30  steps, the objective value is  8.439822004055198
After  31  steps, the objective value is  8.4146547008932
After  32  steps, the objective value is  8.406457283869047
After  33  steps, the objective value is  8.388300511371078
After  34  steps, the objective value is  8.3634717451988
After  35  steps, the objective value is  8.35235322093676
After  36  steps, the objective value is  8.327665994225931
After  37  steps, the objective value is  8.317128662495639
After  38  steps, the objective value is  8.306834817158654
After  39  steps, the objective value is  8.296576954255166
After  40  steps, the objective value is  8.297668115000198
After  41  steps, the objective value is  8.28953382741361
After  42  steps, the objective value is  8.274304724051179
After  43  steps, the objective value is  8.27043092039844З
After  44  steps, the objective value is  8.266702856059974
After  45  steps, the objective value is  8.267752542201585
After  46  steps, the objective value is  8.262881114918375
```

```
After  47  steps, the objective value is  8.254174336589617
After  48  steps, the objective value is  8.250282255994737
After  49  steps, the objective value is  8.248020196331705
After  50  steps, the objective value is  8.248545142689807
After  51  steps, the objective value is  8.244191379756547
After  52  steps, the objective value is  8.237827453154097
After  53  steps, the objective value is  8.235604343103073
After  54  steps, the objective value is  8.23397207065773
After  55  steps, the objective value is  8.233063622578209
After  56  steps, the objective value is  8.235361046996834
After  57  steps, the objective value is  8.229509888183095
After  58  steps, the objective value is  8.226009930287264
After  59  steps, the objective value is  8.228389466309663
After  60  steps, the objective value is  8.226017734766446
After  61  steps, the objective value is  8.22392302960314
After  62  steps, the objective value is  8.222400682597772
After  63  steps, the objective value is  8.220228449397085
After  64  steps, the objective value is  8.218981052001908
After  65  steps, the objective value is  8.217056381873551
After  66  steps, the objective value is  8.216252483382394
After  67  steps, the objective value is  8.215496662937003
After  68  steps, the objective value is  8.215591488396646
After  69  steps, the objective value is  8.216470167075546
After  70  steps, the objective value is  8.214861627228922
After  71  steps, the objective value is  8.21285798992346
After  72  steps, the objective value is  8.214545463553513
After  73  steps, the objective value is  8.212685204015497
After  74  steps, the objective value is  8.212542179823208
After  75  steps, the objective value is  8.210853852890924
After  76  steps, the objective value is  8.209519344610356
After  77  steps, the objective value is  8.208698588193837
After  78  steps, the objective value is  8.209759286204218
After  79  steps, the objective value is  8.208248047954228
After  80  steps, the objective value is  8.207074340417318
After  81  steps, the objective value is  8.20875774371633
After  82  steps, the objective value is  8.205703584231811
After  83  steps, the objective value is  8.206911345049077
After  84  steps, the objective value is  8.208484350694594
After  85  steps, the objective value is  8.204441731339097
After  86  steps, the objective value is  8.203997627603698
After  87  steps, the objective value is  8.203919753789798
After  88  steps, the objective value is  8.203517056707048
After  89  steps, the objective value is  8.203821718679803
After  90  steps, the objective value is  8.2041451964619
After  91  steps, the objective value is  8.202453091341917
After  92  steps, the objective value is  8.20220893124423
After  93  steps, the objective value is  8.202045163546806
After  94  steps, the objective value is  8.202601444158336
After  95  steps, the objective value is  8.201846264410086
After  96  steps, the objective value is  8.202444444690434
After  97  steps, the objective value is  8.200954511532426
After  98  steps, the objective value is  8.200846063563723
After  99  steps, the objective value is  8.20180156696413
```

In [9]:
```
x #Checking x is mostly sparse
```

Out[9]:
```
array([ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        0.00000000e+00, -5.67535245e-02,  0.00000000e+00,  0.00000000e+00,
        0.00000000e+00,  2.19875038e-02,  0.00000000e+00,  0.00000000e+00,
        3.48128330e-02,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        9.67581026e-02,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        0.00000000e+00,  0.00000000e+00, -1.10463871e-01,  2.95855667e-02,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
```