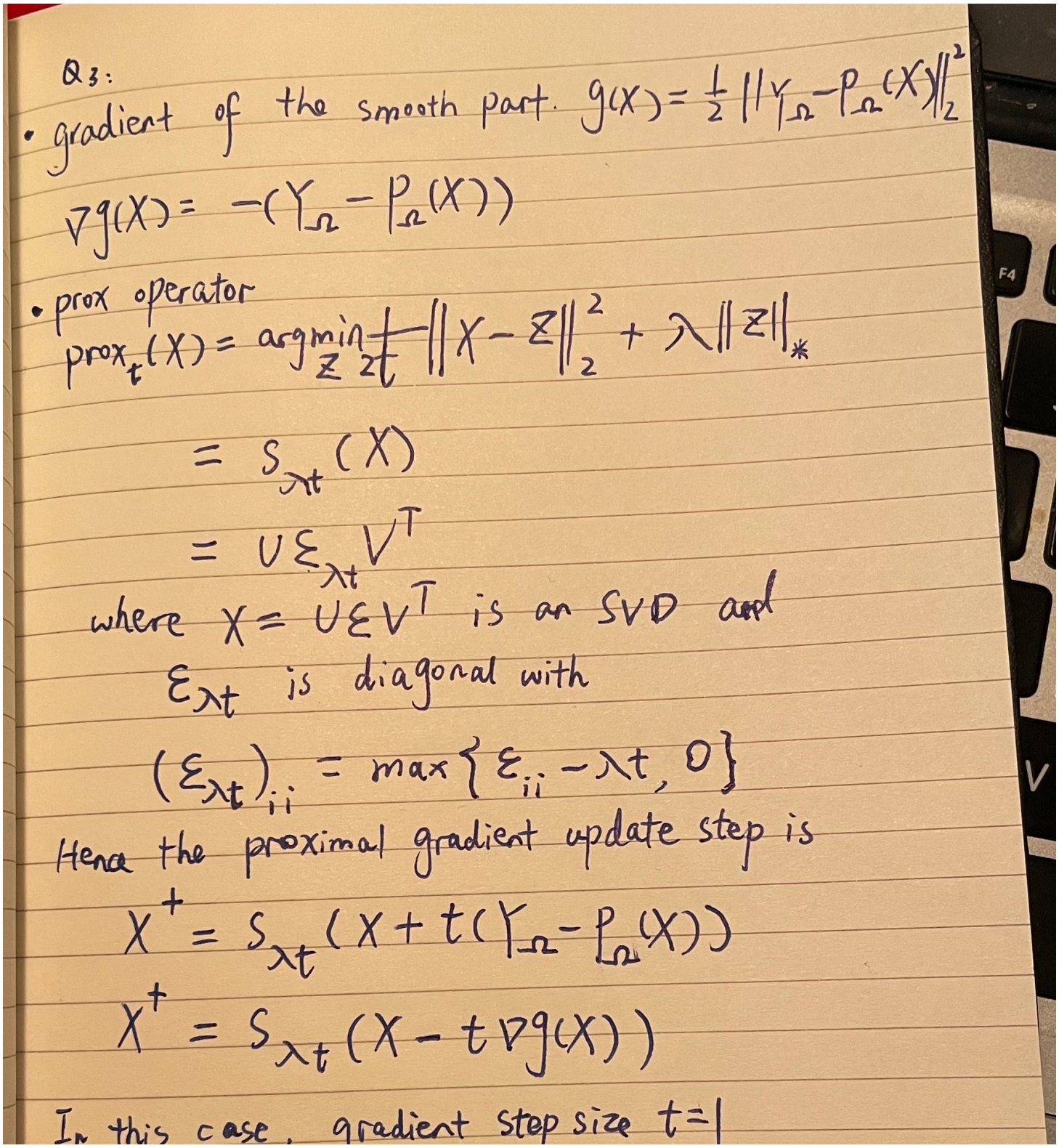
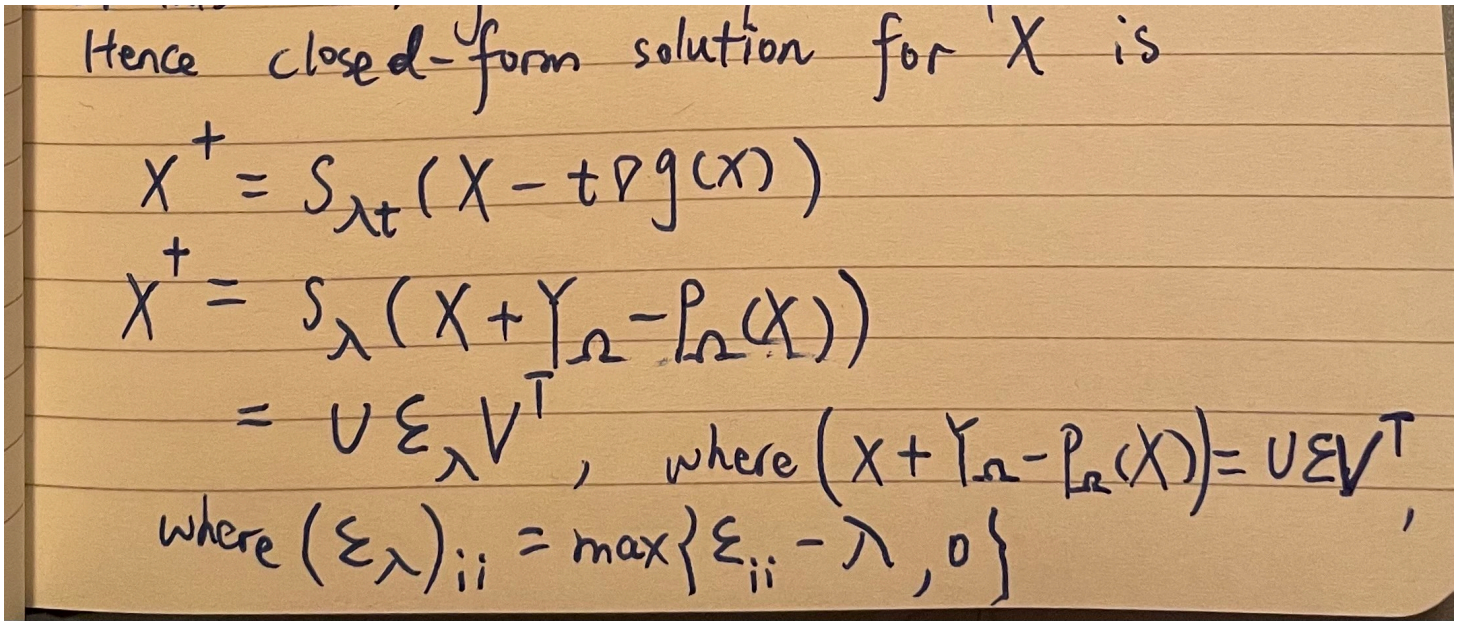
Question 3

The closed form solution for matrix X is shown below.





The pseudo code for the algorithm is as follows.

The complete algorithm is

Input : observation samples $Y_{ij}, (i,j) \in \Omega$ of matrix $Y \in \mathbb{R}^{m \times n}$

$k = 0$

Initialize : $X_0 = 0$, $\lambda = 10$, gradient step size $t = 1$

while not converged do

$$X_{k+1} \leftarrow S_\lambda(X_k + Y_\Omega - P_\Omega(X_k))$$

$$= U \mathcal{E}_\lambda V^T$$

~~where $U \mathcal{E}_\lambda V^T =$~~

where $(U, \mathcal{E}, V^T) = \text{svd}(X_k + Y_\Omega - P_\Omega(X_k))$

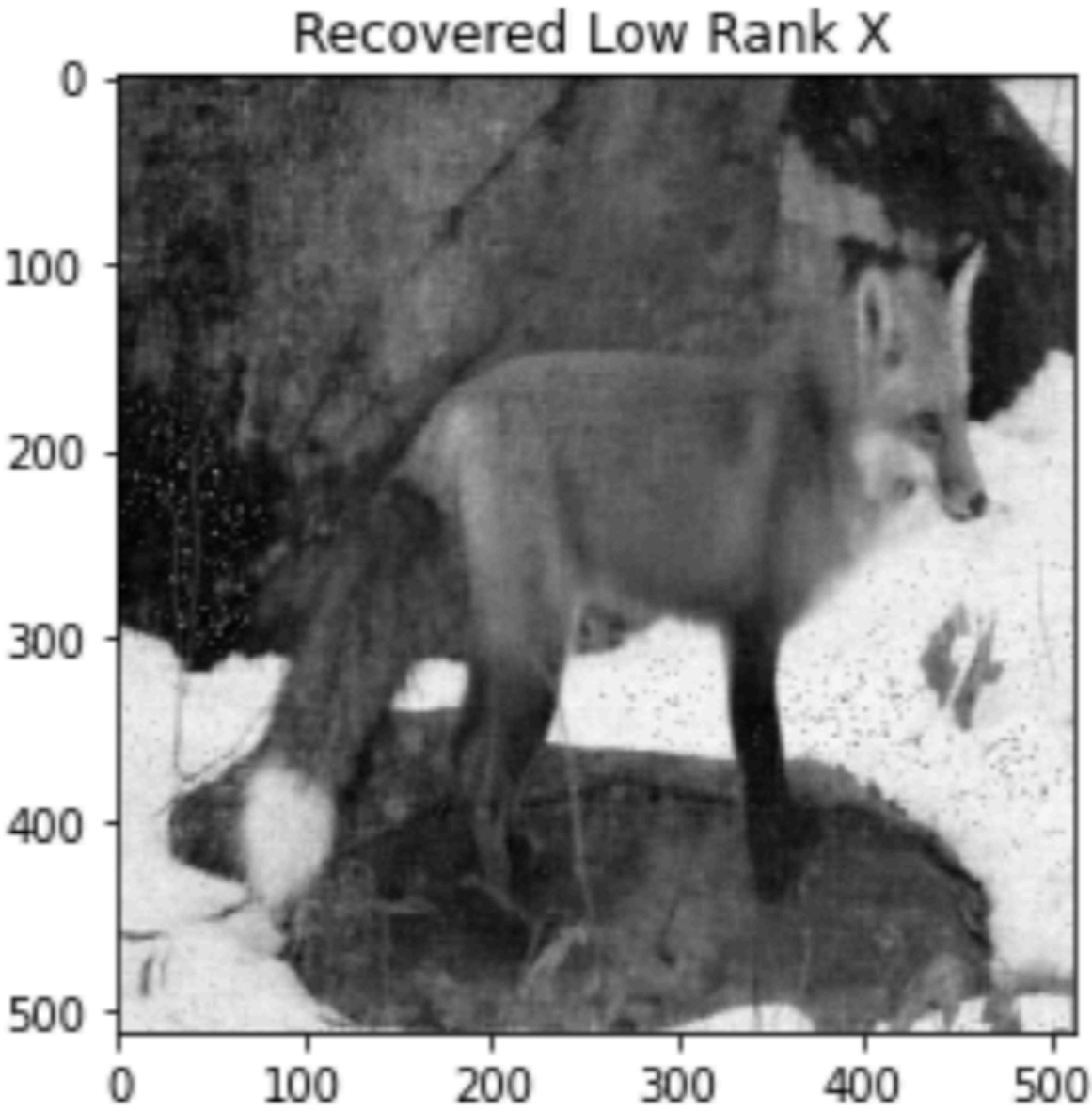
$$(\mathcal{E}_\lambda)_{ii} = \max\{\mathcal{E}_{ii} - \lambda, 0\}$$

$k \leftarrow k+1$

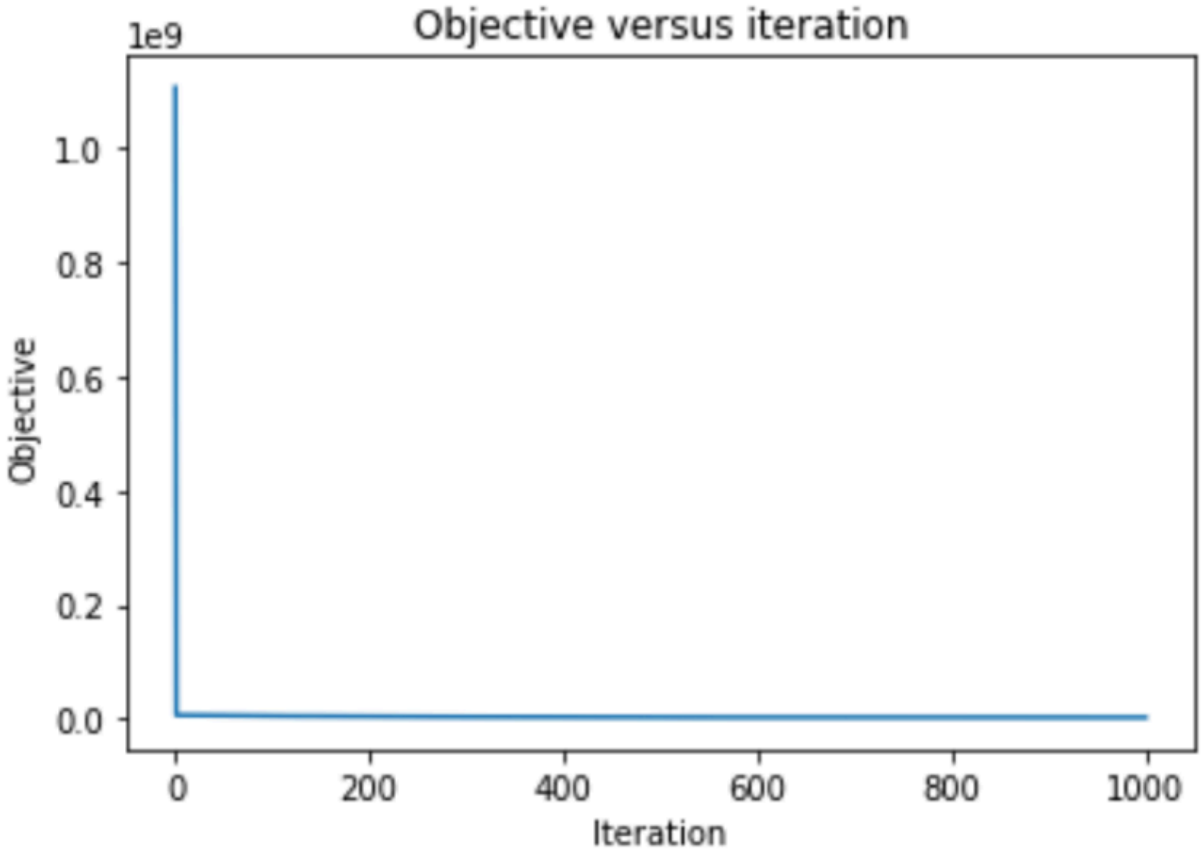
end while

output $X \in \mathbb{R}^{m \times n}$

The output image X is displayed below.



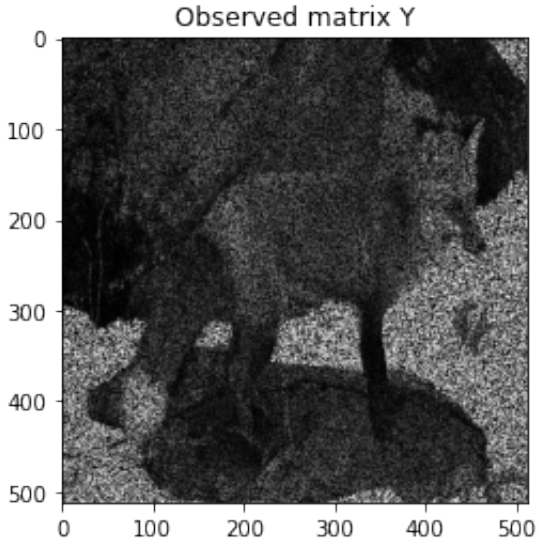
The objective function versus iteration is plotted as follows.



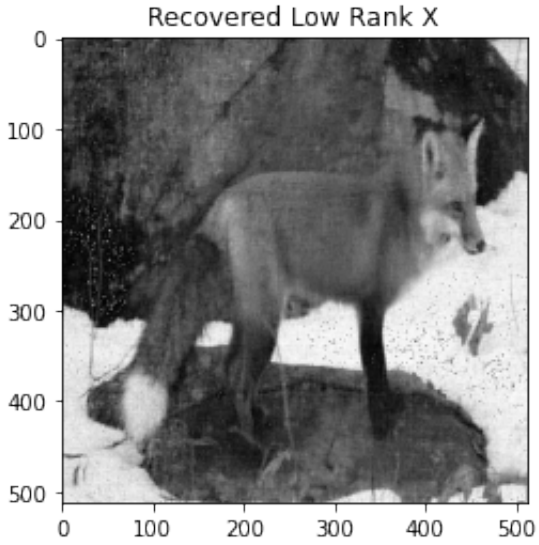
In [26]:
```
import scipy.io
import numpy as np
from scipy.linalg import svd
import time
import matplotlib.pyplot as plt
```

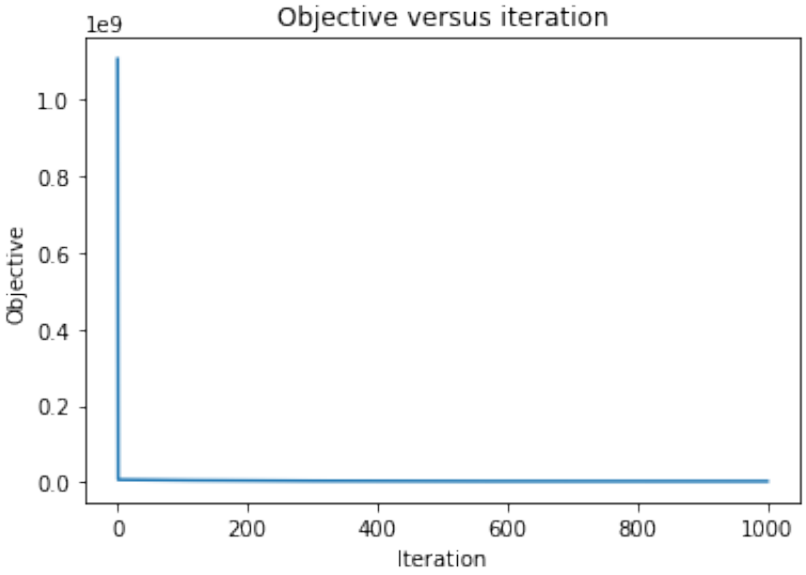
In [4]:
```
omg = scipy.io.loadmat('Question3-1.mat')['P']
Y = plt.imread('Question3.jpg')
```

In [8]:
```
plt.figure()
plt.title('Observed matrix Y')
plt.imshow(Y,cmap='gray')
plt.show()
```



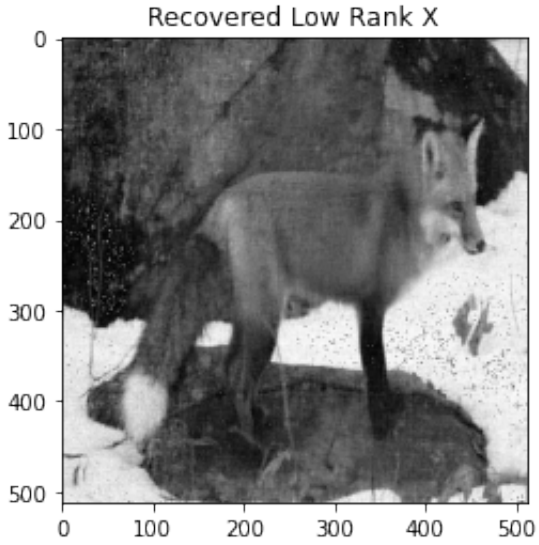
In [34]:
```
start = time.time()
lam = 10
t = 1
m, n = Y.shape
X_hat = np.zeros((m, n))
iter = 0
max_iter = 1000
obj = [lam*np.linalg.norm(X_hat, 'nuc') + 0.5*np.linalg.norm(Y*omg-X_h
at*omg)**2]

while iter < max_iter:
    iter += 1
    u, s, vh = svd(X_hat+Y*omg-X_hat*omg,full_matrices=False)
    svp = (s > lam).sum()
    X_hat = (u[:, :svp]*(s[:svp]-lam))@vh[:svp]
    obj.append(lam*np.linalg.norm(X_hat, 'nuc') + 0.5*np.linalg.norm(Y
*omg-X_hat*omg)**2)

plt.figure()
plt.imshow(X_hat.astype('uint8'), cmap='gray')
plt.title("Recovered Low Rank X")
plt.show()

plt.figure()
plt.plot(range(iter+1),obj)
plt.title('Objective versus iteration')
plt.xlabel('Iteration')
plt.ylabel('Objective')
plt.show()

print(f'The algorithm took {(time.time()-start)/60} min to converge.')
```

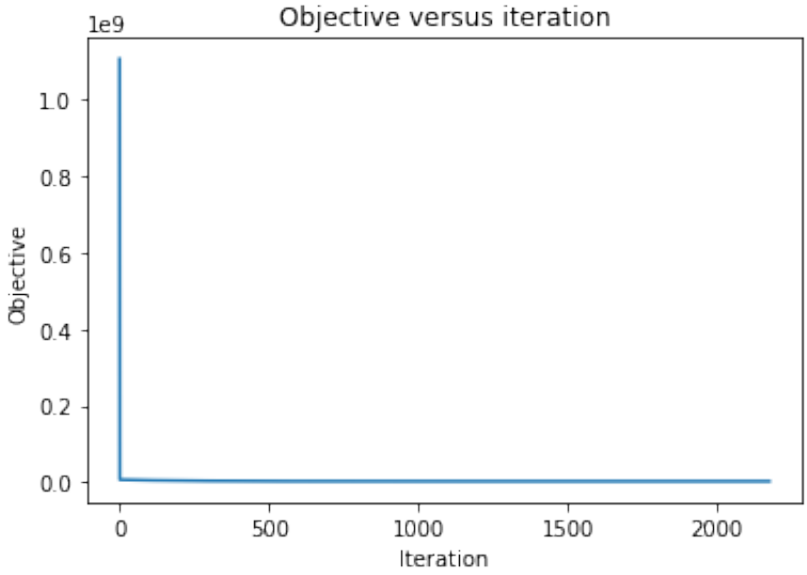




The algorithm took 2.0115618189175923 min to converge.

In [39]:
```
start = time.time()
lam = 10
t = 1
m, n = Y.shape
X_hat = np.zeros((m, n))
iter = 0
obj = [lam*np.linalg.norm(X_hat, 'nuc') + 0.5*np.linalg.norm(Y*omg-X_h
at*omg)**2]
diff = np.inf
tol = 1e-3

while diff > tol:
    iter += 1
    u, s, vh = svd(X_hat+Y*omg-X_hat*omg,full_matrices=False)
    svp = (s > lam).sum()
    X_hat = (u[:, :svp]*(s[:svp]-lam))@vh[:svp]
    obj.append(lam*np.linalg.norm(X_hat, 'nuc') + 0.5*np.linalg.norm(Y
*omg-X_hat*omg)**2)
    diff = abs(obj[-1] - obj[-2])

plt.figure()
plt.imshow(X_hat.astype('uint8'), cmap='gray')
plt.title("Recovered Low Rank X")
plt.show()

plt.figure()
plt.plot(range(iter+1),obj)
plt.title('Objective versus iteration')
plt.xlabel('Iteration')
plt.ylabel('Objective')
plt.show()

print(f'The algorithm took {(time.time()-start)/60} min to converge.')
```






```
The algorithm took 4.575347765286764 min to converge.
```

In [ ]: