Activity 18

Setup

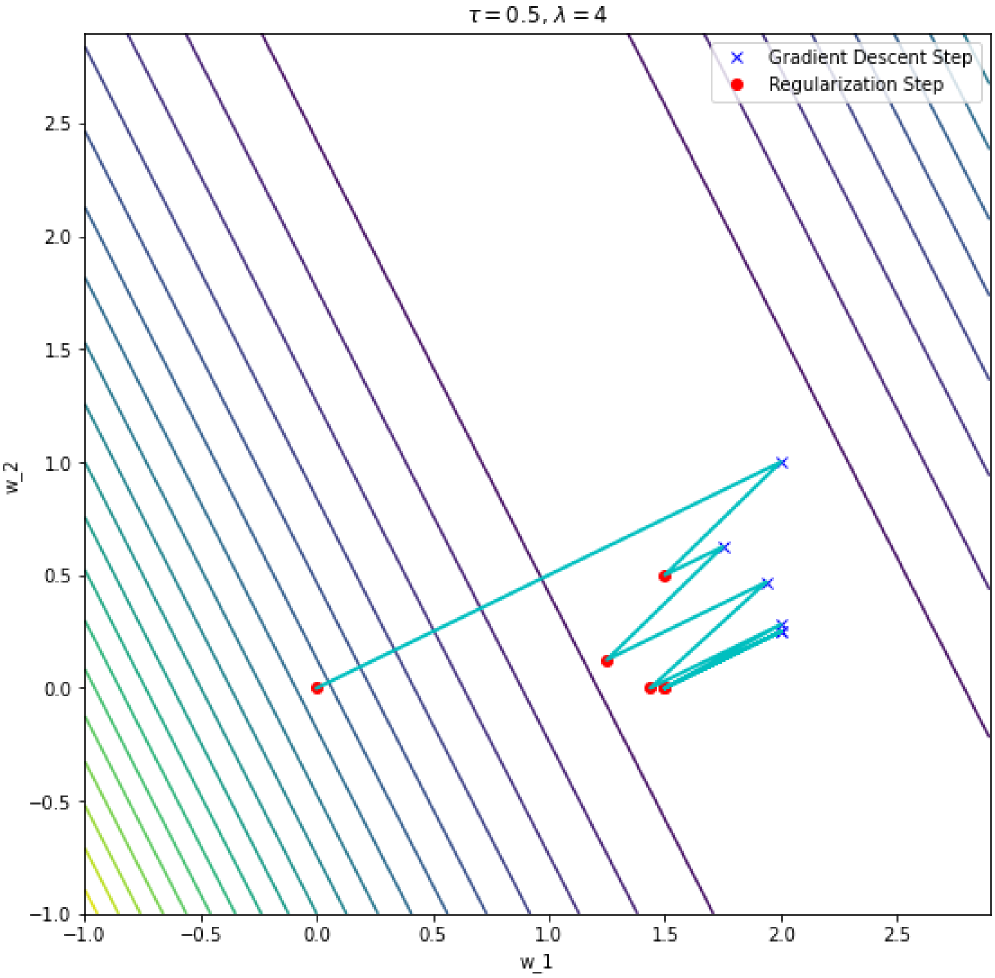
In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
def prxgraddescent_l1(X,y,tau,lam,w_init,it):

## compute it iterations of L2 proximal gradient descent starting at w1
## w_{k+1}= (w_k - tau*X'*(X*w_k - y)/(1+lam*tau)
## step size tau
    W = np.zeros((w_init.shape[0], it+1))
    Z = np.zeros((w_init.shape[0], it+1))
    W[:,[0]] = w_init
    for k in range(it):
        Z[:,[k+1]] = W[:,[k]] - tau * X.T @ (X @ W[:,[k]] - y);
        W[:,[k+1]] = np.sign(Z[:,[k+1]])* np.clip(np.abs(Z[:,[k+1]])-lam*tau/2,0,float("inf"))

    return W,Z
```

In [3]:
```
## Proximal gradient descent trajectories
## Least Squares Problem
X = np.array([[2, 1]])
y = np.array([[4]])

### Find values of f(w), the contour plot surface for
w1 = np.arange(-1,3,.1)
w2 = np.arange(-1,3,.1)
fw = np.zeros((len(w1), len(w2)))
for i in range(len(w2)):
    for j in range(len(w1)):
        w = np.array([ [w1[j]], [w2[i]] ])
        fw[i,j] = (X @ w - y)**2
```

Question 3a)

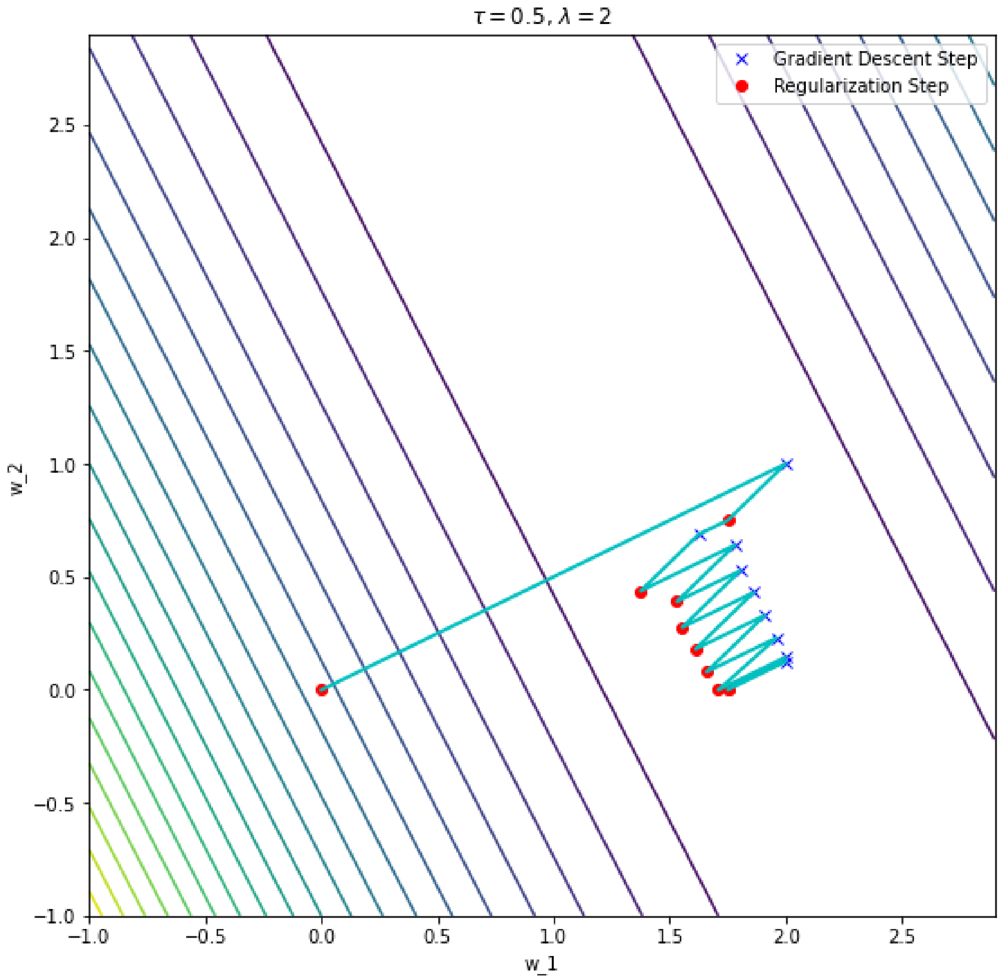
In [4]:
```
## Find and display weights generated by gradient descent

w_init = np.array([[0],[0]])
lam = 4;
it = 6
tau = 0.25
W,Z = prxgraddescent_l1(X,y,tau,lam,w_init,it)
# Concatenate gradient and regularization steps to display trajectory
G = np.zeros((2,0))
for i in range(it):
    G = np.hstack((G,np.hstack((W[:,[i]],Z[:,[i+1]]))))

plt.figure(figsize=(9,9))
plt.contour(w1,w2,fw,20)
plt.plot(Z[0,1::],Z[1,1:],'bx',linewidth=2, label="Gradient Descent Step")
plt.plot(W[0,:],W[1,:],'ro',linewidth=2, label="Regularization Step")
plt.plot(G[0,:],G[1,:],'-c',linewidth=2)
plt.legend()
plt.xlabel('w_1')
plt.ylabel('w_2')
plt.title('$\\tau = $'+str(.5)+', $\lambda = $'+str(lam));
```



It takes 5 iterations for the algorithm to converge (it=6 and it=5 yield different ressults).

Question 3b)

In [5]:
```
## Find and display weights generated by gradient descent

w_init = np.array([[0],[0]])
lam = 2;
it = 9
tau = 0.25
W,Z = prxgraddescent_l1(X,y,tau,lam,w_init,it)
# Concatenate gradient and regularization steps to display trajectory
G = np.zeros((2,0))
for i in range(it):
    G = np.hstack((G,np.hstack((W[:,[i]],Z[:,[i+1]]))))

plt.figure(figsize=(9,9))
plt.contour(w1,w2,fw,20)
plt.plot(Z[0,1::],Z[1,1:],'bx',linewidth=2, label="Gradient Descent Step")
plt.plot(W[0,:],W[1,:],'ro',linewidth=2, label="Regularization Step")
plt.plot(G[0,:],G[1,:],'-c',linewidth=2)
plt.legend()
plt.xlabel('w_1')
plt.ylabel('w_2')
plt.title('$\\tau = $'+str(.5)+', $\lambda = $'+str(lam));
```



It takes 8 iterations for the algorithm to converge after reducing lambda.

In [6]:
```
print("It converged to value of w")
print(W[:,9])
```

```
It converged to value of w
[1.75 0.  ]
```

Question 3c)

In the regularization step, w shrinks in the direction of the origin.

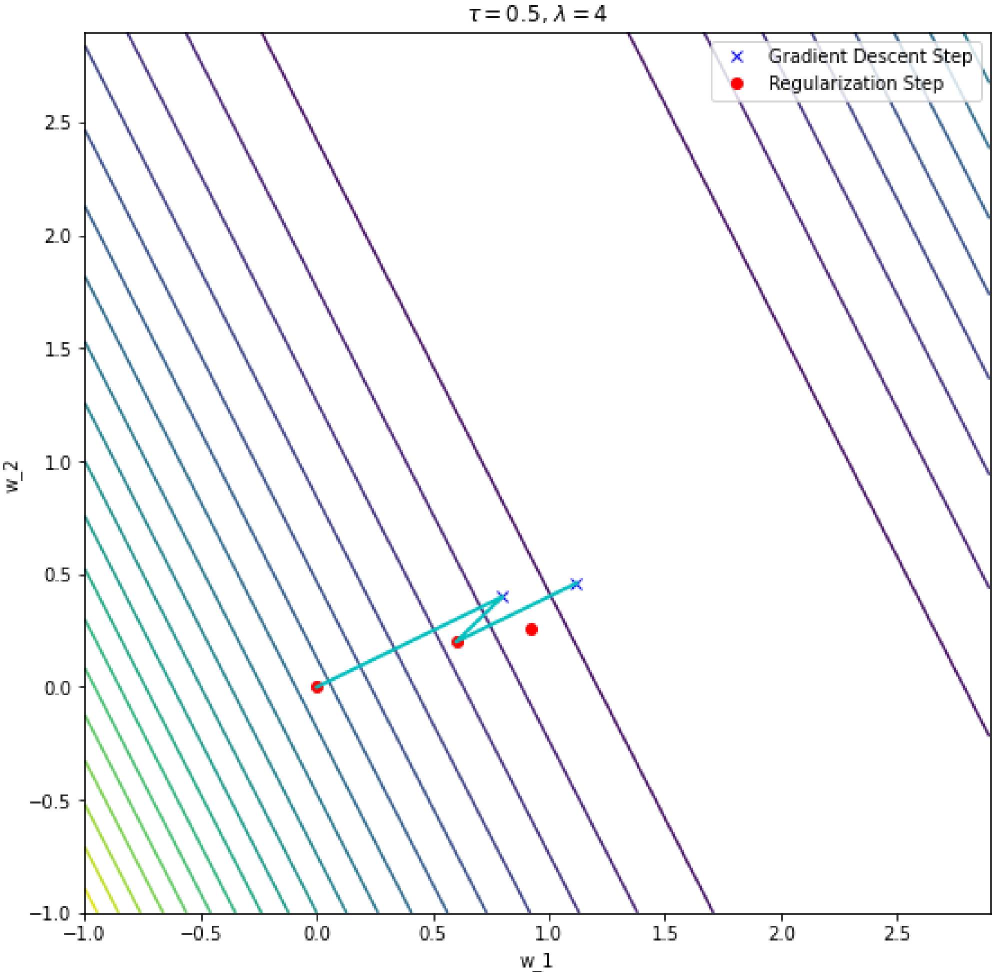
The weoghts converge to the minimum.

Question 4a)

Tau < 1/||X||op and ||X||op is equal to the first singular value in SVD of X. Thus Tau has a maximum value of 0.5 as sigma1 is 2.

Question 4b)

In [7]:
```
## Find and display weights generated by gradient descent

w_init = np.array([[0],[0]])
lam = 4;
it = 2
tau = 0.1
W,Z = prxgraddescent_l1(X,y,tau,lam,w_init,it)
# Concatenate gradient and regularization steps to display trajectory
G = np.zeros((2,0))
for i in range(it):
    G = np.hstack((G,np.hstack((W[:,[i]],Z[:,[i+1]]))))

plt.figure(figsize=(9,9))
plt.contour(w1,w2,fw,20)
plt.plot(Z[0,1::],Z[1,1:],'bx',linewidth=2, label="Gradient Descent Step")
plt.plot(W[0,:],W[1,:],'ro',linewidth=2, label="Regularization Step")
plt.plot(G[0,:],G[1,:],'-c',linewidth=2)
plt.legend()
plt.xlabel('w_1')
plt.ylabel('w_2')
plt.title('$\\tau = $'+str(.5)+', $\lambda = $'+str(lam));
```



In [ ]: