Activity 16

1. $f(w) = (w - w_{LS})^T X^T X (w - w_{LS}) + C$

a. • when $w = w_{LS}$

$f(w) = 0 \quad X^T X \quad 0 \quad + C$

$f(w) = C$

• when $w \neq w_{LS}$

$f(w) = Q\, X^T X\, Q > 0$

$f(w)$: positive definite > 0

b. $w_{LS} = V \Sigma^{-1} U^T y \qquad X = U \Sigma V^T \qquad V = I \qquad \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1/2 \end{bmatrix}$

$$= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \frac{1}{1/2} \begin{bmatrix} 1/2 & 0 \\ 0 & 1 \end{bmatrix} \underset{2\times 4}{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}} \underset{4\times 1}{\begin{bmatrix} 1 \\ 1/2 \\ 1 \\ 0 \end{bmatrix}}$$

$$= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \frac{1}{1/2} \begin{bmatrix} 1/2 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1/2 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} 2 \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$w_{LS} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$X = U \Sigma V^T$

$$= \underset{4\times 2}{\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}} \underset{2\times 2}{\begin{bmatrix} 1 & 0 \\ 0 & 1/2 \end{bmatrix}} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 \\ 0 & 1/2 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$X^TX = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1/2 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 \\ 0 & 1/4 \end{bmatrix}$$

$$f(w) = (w - w_{LS})^T X^TX (w - w_{LS}) + C$$

$$= \begin{bmatrix} w_1 - w_{01} \\ w_2 - w_{02} \end{bmatrix}^T \begin{bmatrix} 1 & 0 \\ 0 & 1/4 \end{bmatrix} \begin{bmatrix} w_1 - w_{01} \\ w_2 - w_{02} \end{bmatrix}$$

$$= \begin{bmatrix} w_1 - w_{01} & w_2 - w_{02} \end{bmatrix} \underset{2\times2}{\begin{bmatrix} 1 & 0 \\ 0 & 1/4 \end{bmatrix}} \underset{2\times1}{\begin{bmatrix} w_1 - w_{01} \\ w_2 - w_{02} \end{bmatrix}}$$

$$= \begin{bmatrix} w_1 - w_{01} & w_2 - w_{02} \end{bmatrix} \begin{bmatrix} w_1 - w_{01} \\ 1/4\, w_2 - w_{02} \end{bmatrix}$$

$$= (w_1 - w_{01})^2 + \frac{1}{4}(w_2 - w_{02})^2 + C$$

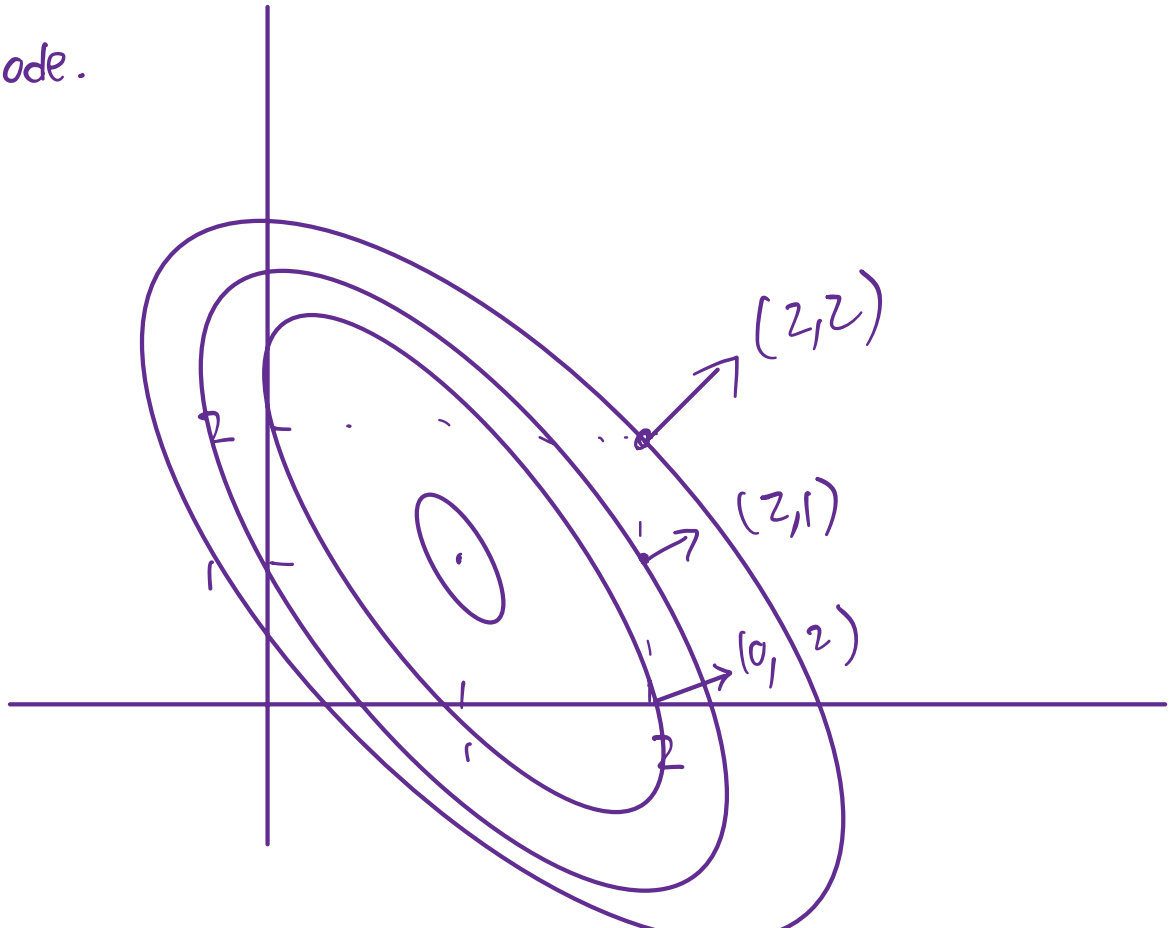
$$f(w) = (w_1 - 1)^2 + \frac{1}{4}(w_2 - 1)^2 + C$$

b. Plot → code.

c. code

d. code

e.



2. a. max value for the step τ

$$0 < \tau < \frac{2}{\sigma_i^2}$$

$$0 < \tau < \frac{2}{1}$$

$$0 < \tau < 2$$

period_16_activity_starter

March 21, 2024

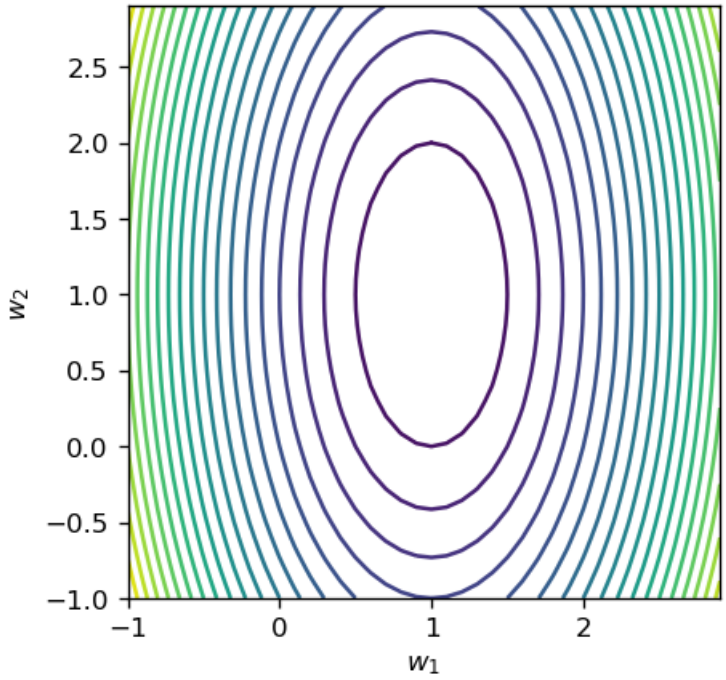
[1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```
## DO NOT Change
def graddescent(X,y,tau,w_init,it):
    """
    compute 10 iterations of gradient descent starting at w1
    w_{k+1}= w_k - tau*X'*(X*w_k - y)
    """
    W = np.zeros((w_init.shape[0],it))
    W[:,[0]] = w_init
    for k in range(it-1):
        W[:,[k+1]] = W[:,[k]] - tau * X.T @ (X @ W[:,[k]] - y)
    return W
```

1 Question 1b)

[3]:
```
U = np.array([[1, 0], [0, 1], [0, 0], [0, 0]])
S = np.array([[1, 0], [0, 0.5]])
Sinv = np.linalg.inv(S)
V = np.eye(2)
X = U @ S @ V.T
y = np.array([[1], [0.5], [1], [0]])

### Find Least Squares Solution
w_ls = V @ Sinv @ U.T @ y
c = y.T @ y - y.T @ X @ w_ls

### 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] = (w-w_ls).T @ X.T @ X @ (w-w_ls) + c
```

```
### Plot the countours
plt.figure(num=None, figsize=(4, 4), dpi=120)
plt.contour(w1,w2,fw,20)
plt.xlim([-1,3])
plt.ylim([-1,3])
plt.xlabel('$w_1$')
plt.ylabel('$w_2$')
plt.axis('square');
```

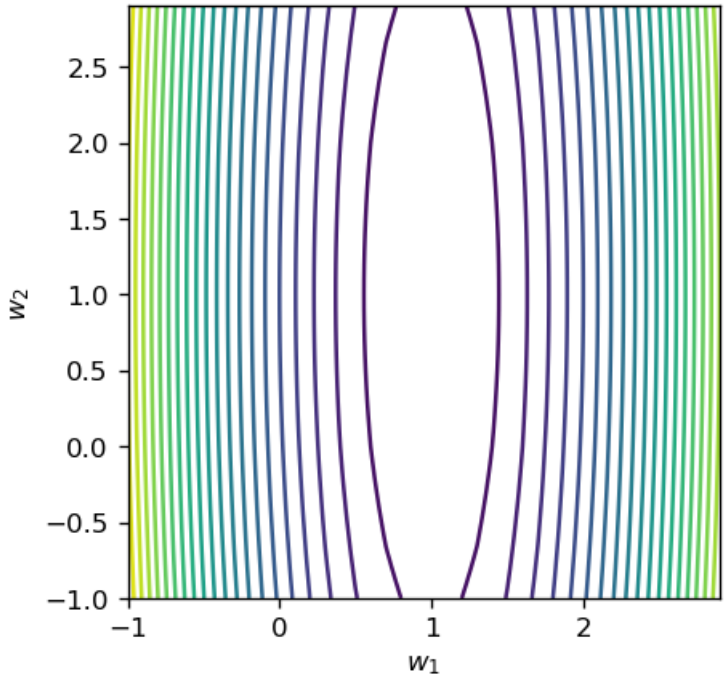


2 Question 1c)

The sigma will affect the shape of elipse (major and minor)

[5]:
```
## Copy and paste code from 1b
U = np.array([[1, 0], [0, 1], [0, 0], [0, 0]])
S = np.array([[1, 0], [0, 0.2]])
Sinv = np.linalg.inv(S)
V = np.eye(2)
X = U @ S @ V.T
y = np.array([[1], [0.2], [1], [0]])

### Find Least Squares Solution
w_ls = V @ Sinv @ U.T @ y
```

```
c = y.T @ y - y.T @ X @ w_ls

### 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] = (w-w_ls).T @ X.T @ X @ (w-w_ls) + c

### Plot the countours
plt.figure(num=None, figsize=(4, 4), dpi=120)
plt.contour(w1,w2,fw,20)
plt.xlim([-1,3])
plt.ylim([-1,3])
plt.xlabel('$w_1$')
plt.ylabel('$w_2$')
plt.axis('square');
```

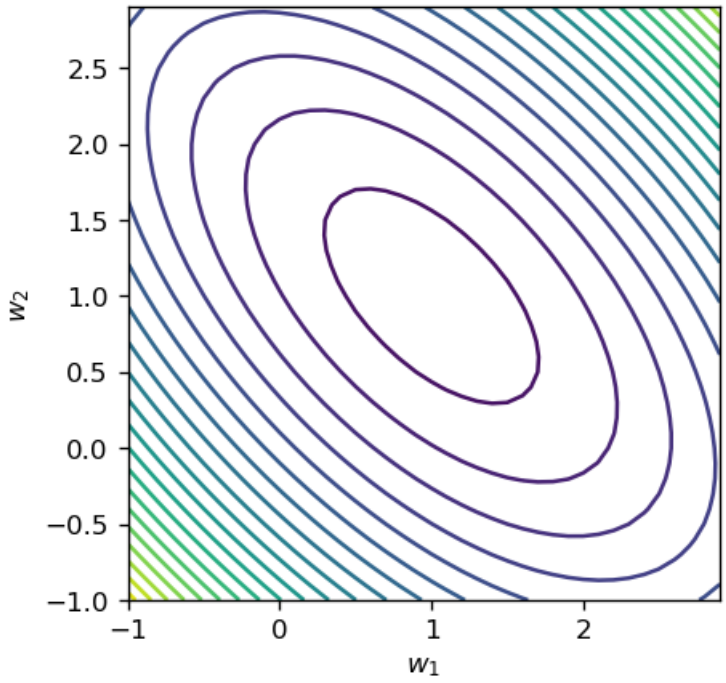


3 Question 1d)

The V matrix will rotate the elipse

[15]:
```
## Copy and paste code from 1b
## Copy and paste code from 1b
import math
U = np.array([[1, 0], [0, 1], [0, 0], [0, 0]])
S = np.array([[1, 0], [0, 0.5]])
Sinv = np.linalg.inv(S)
# V = np.eye(2)
V = np.array([[1/np.sqrt(2),1/np.sqrt(2)], [1/np.sqrt(2),-1/np.sqrt(2)]])
print(V.shape)
X = U @ S @ V.T
y = np.array([[np.sqrt(2)], [0], [1], [0]])

### Find Least Squares Solution
w_ls = V @ Sinv @ U.T @ y
c = y.T @ y - y.T @ X @ w_ls

### 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] = (w-w_ls).T @ X.T @ X @ (w-w_ls) + c

### Plot the countours
plt.figure(num=None, figsize=(4, 4), dpi=120)
plt.contour(w1,w2,fw,20)
plt.xlim([-1,3])
plt.ylim([-1,3])
plt.xlabel('$w_1$')
plt.ylabel('$w_2$')
plt.axis('square');
```

```
(2, 2)
```

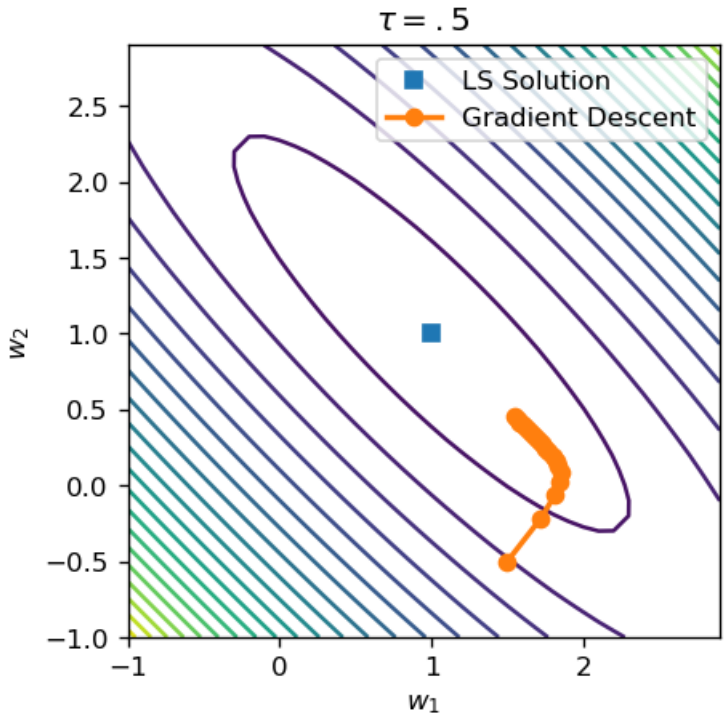


4 Question 2b)

If start from (0, 0), it is faster to convergence.

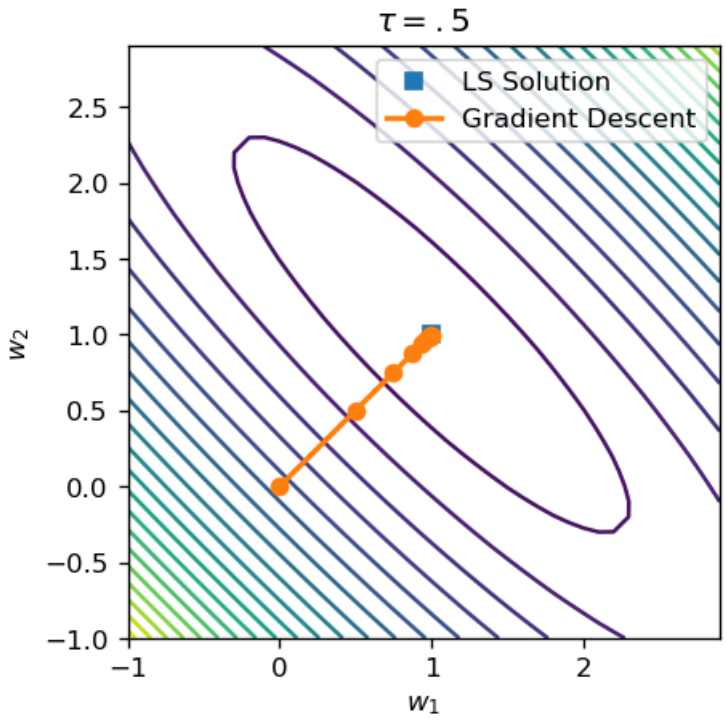
[16]:
```
U = np.array([[1, 0], [0, 1], [0, 0], [0, 0]])
S = np.array([[1, 0], [0, 0.5]])
Sinv = np.linalg.inv(S)
V = 1/np.sqrt(2)*np.array([[1, 1], [1, -1]])
X = U @ S @ V.T
y = np.array([[np.sqrt(2)], [0], [1], [0]])

### Find Least Squares Solution
w_ls = V @ Sinv @ U.T @ y
c = y.T @ y - y.T @ X @ w_ls

### 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(w1)):
    for j in range(len(w2)):
        w = np.array([ [w1[i]], [w2[j]] ])
        fw[i,j] = (w-w_ls).T @ X.T @ X @ (w-w_ls) + c
```

[32]:
```
w_init = np.array([[1.5], [-0.5]])# complete this line with a 2x1 numpy array␣
 ↪for the values specified in the activity
it = 20
tau = .5
W = graddescent(X,y,tau,w_init,it);


### Create plot
plt.figure(num=None, figsize=(4, 4), dpi=120)
plt.contour(w1,w2,fw,20)
plt.plot(w_ls[0],w_ls[1],"s", label="LS Solution")
plt.plot(W[0,:],W[1,:],'o-',linewidth=2, label="Gradient Descent")
plt.legend()
plt.xlim([-1,3])
plt.xlabel('$w_1$')
plt.ylim([-1,3])
plt.ylabel('$w_2$')
plt.title(r'$\tau = .5$');
plt.axis('square');
```



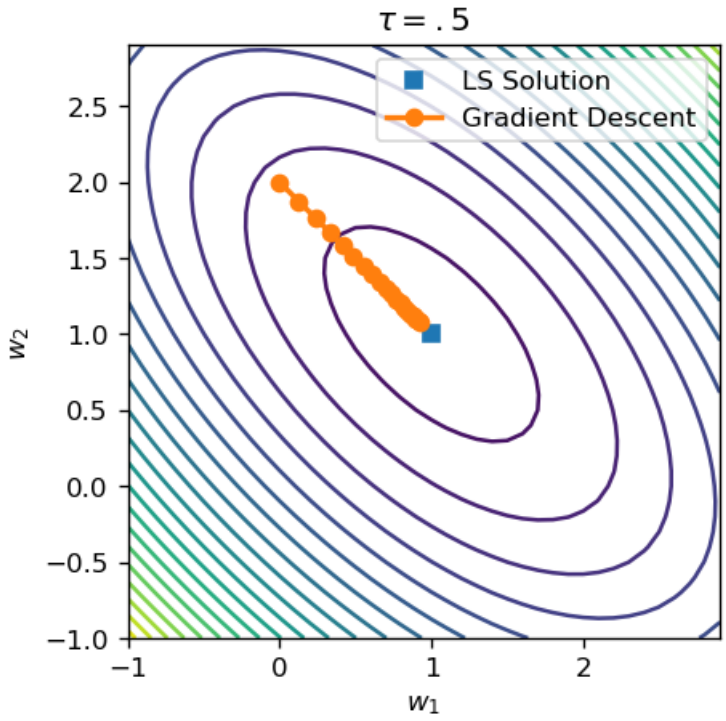
[33]:
```
# copy and paste code from above
w_init = np.array([[0], [0]])# complete this line with a 2x1 numpy array for␣
 ↪the values specified in the activity
```

```
it = 10
tau = .5
W = graddescent(X,y,tau,w_init,it);


### Create plot
plt.figure(num=None, figsize=(4, 4), dpi=120)
plt.contour(w1,w2,fw,20)
plt.plot(w_ls[0],w_ls[1],"s", label="LS Solution")
plt.plot(W[0,:],W[1,:],'o-',linewidth=2, label="Gradient Descent")
plt.legend()
plt.xlim([-1,3])
plt.xlabel('$w_1$')
plt.ylim([-1,3])
plt.ylabel('$w_2$')
plt.title(r'$\tau = .5$');
plt.axis('square');
```



[22]:
```
# copy and paste code from above
w_init = np.array([[0], [2]])# complete this line with a 2x1 numpy array for␣
 ↪the values specified in the activity
it = 20
tau = .5
```

```
W = graddescent(X,y,tau,w_init,it);


### Create plot
plt.figure(num=None, figsize=(4, 4), dpi=120)
plt.contour(w1,w2,fw,20)
plt.plot(w_ls[0],w_ls[1],"s", label="LS Solution")
plt.plot(W[0,:],W[1,:],'o-',linewidth=2, label="Gradient Descent")
plt.legend()
plt.xlim([-1,3])
plt.xlabel('$w_1$')
plt.ylim([-1,3])
plt.ylabel('$w_2$')
plt.title(r'$\tau = .5$');
plt.axis('square');
```

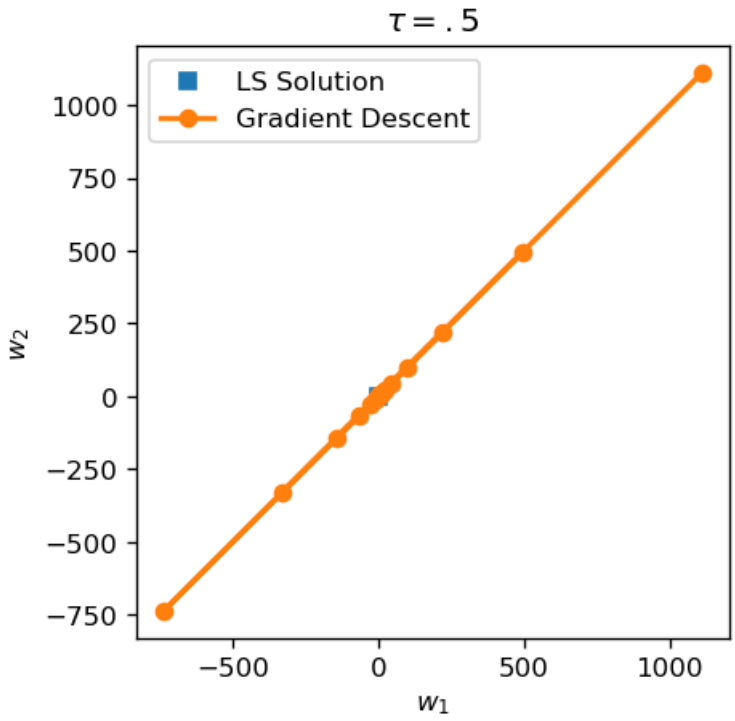


5 Question 2c)

It doesn't converge because tau is too big.

[25]:
```
w_init = np.array([[1.5], [-0.5]])# complete this line with a 2x1 numpy array␣
↪for the values specified in the activity
it = 20
```

```
tau = 2.5
W = graddescent(X,y,tau,w_init,it);


### Create plot
plt.figure(num=None, figsize=(4, 4), dpi=120)
plt.contour(w1,w2,fw,20)
plt.plot(w_ls[0],w_ls[1],"s", label="LS Solution")
plt.plot(W[0,:],W[1,:],'o-',linewidth=2, label="Gradient Descent")
plt.legend()
plt.xlim([-1,3])
plt.xlabel('$w_1$')
plt.ylim([-1,3])
plt.ylabel('$w_2$')
plt.title(r'$\tau = .5$');
plt.axis('square');
```

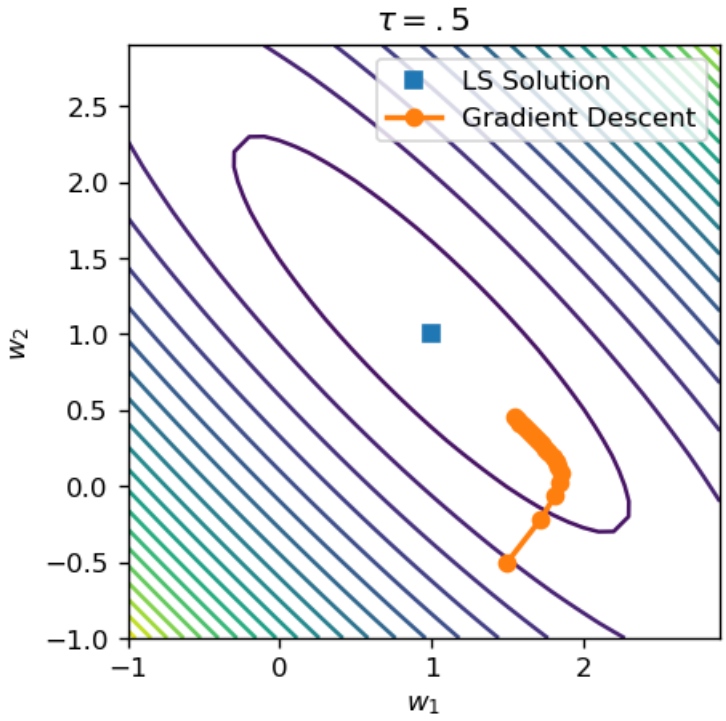


6 Question 2d)

As the elipse major axis is bigger, it takes longer to converge. The sigma will affect the shape of elipse.

[34]:
```
## Copy and paste code from above
U = np.array([[1, 0], [0, 1], [0, 0], [0, 0]])
S = np.array([[1, 0], [0, 0.25]])
Sinv = np.linalg.inv(S)
V = 1/np.sqrt(2)*np.array([[1, 1], [1, -1]])
X = U @ S @ V.T
y = np.array([[np.sqrt(2)], [0], [1], [0]])

### Find Least Squares Solution
w_ls = V @ Sinv @ U.T @ y
c = y.T @ y - y.T @ X @ w_ls

### 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(w1)):
    for j in range(len(w2)):
        w = np.array([ [w1[i]], [w2[j]] ])
        fw[i,j] = (w-w_ls).T @ X.T @ X @ (w-w_ls) + c
```

[30]:
```
w_init = np.array([[1.5], [-0.5]])# complete this line with a 2x1 numpy array␣
 ↪for the values specified in the activity
it = 20
tau = .5
W = graddescent(X,y,tau,w_init,it);


### Create plot
plt.figure(num=None, figsize=(4, 4), dpi=120)
plt.contour(w1,w2,fw,20)
plt.plot(w_ls[0],w_ls[1],"s", label="LS Solution")
plt.plot(W[0,:],W[1,:],'o-',linewidth=2, label="Gradient Descent")
plt.legend()
plt.xlim([-1,3])
plt.xlabel('$w_1$')
plt.ylim([-1,3])
plt.ylabel('$w_2$')
plt.title(r'$\tau = .5$');
plt.axis('square');
```



7 Question 2e)

Sigma affects the shape of elipse. If the ratio between the major and the minor axis of elipse is huge. It takes longer to converge.

[ ]: