In [3]:
```
import numpy as np
import matplotlib.pyplot as plt
```

Question 1-a

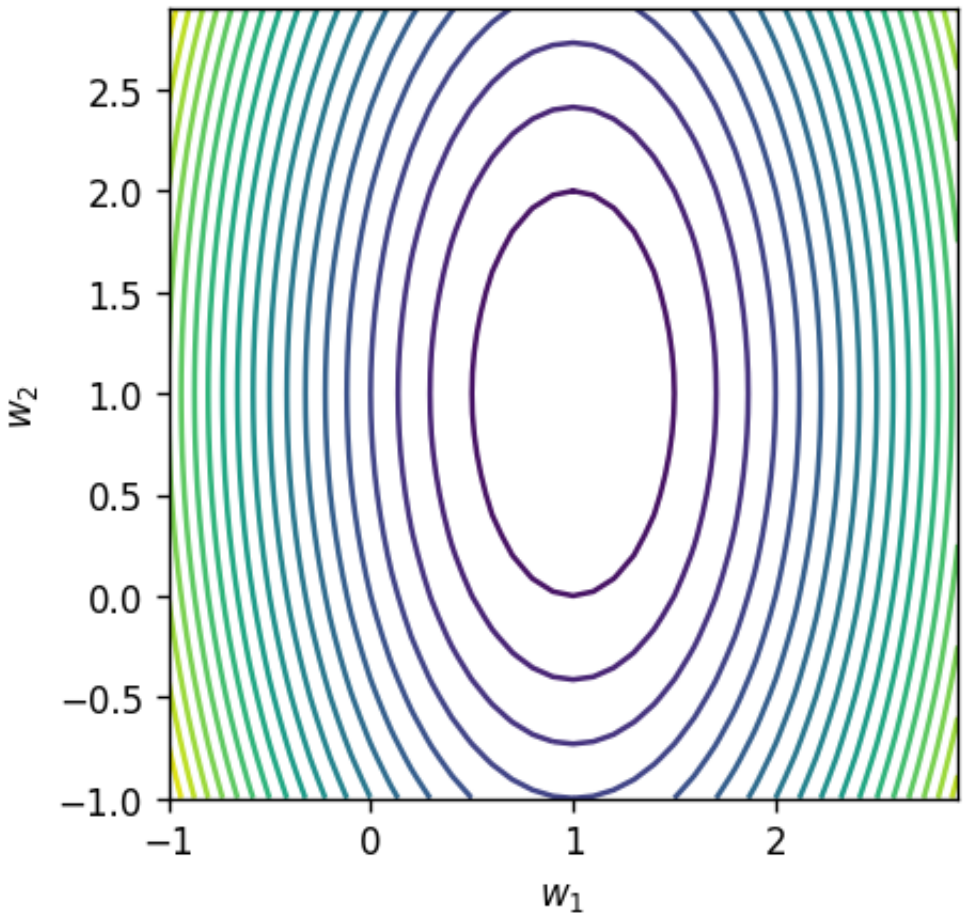
When $w = w_{LS}$ $w - w_{LS} = 0$ so the first part of $f(w)$ becomes 0 and $f(w) = c$

In [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
```

Question 1b)

In [4]:
```
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');
```

```
/var/folders/tn/v9tpvrrs4qgdbw0xd1q0l8qh0000gn/T/ipykernel_82298/146905903.p
y:19: DeprecationWarning: Conversion of an array with ndim > 0 to a scalar i
s deprecated, and will error in future. Ensure you extract a single element
from your array before performing this operation. (Deprecated NumPy 1.25.)
  fw[i,j] = (w-w_ls).T @ X.T @ X @ (w-w_ls) + c
```

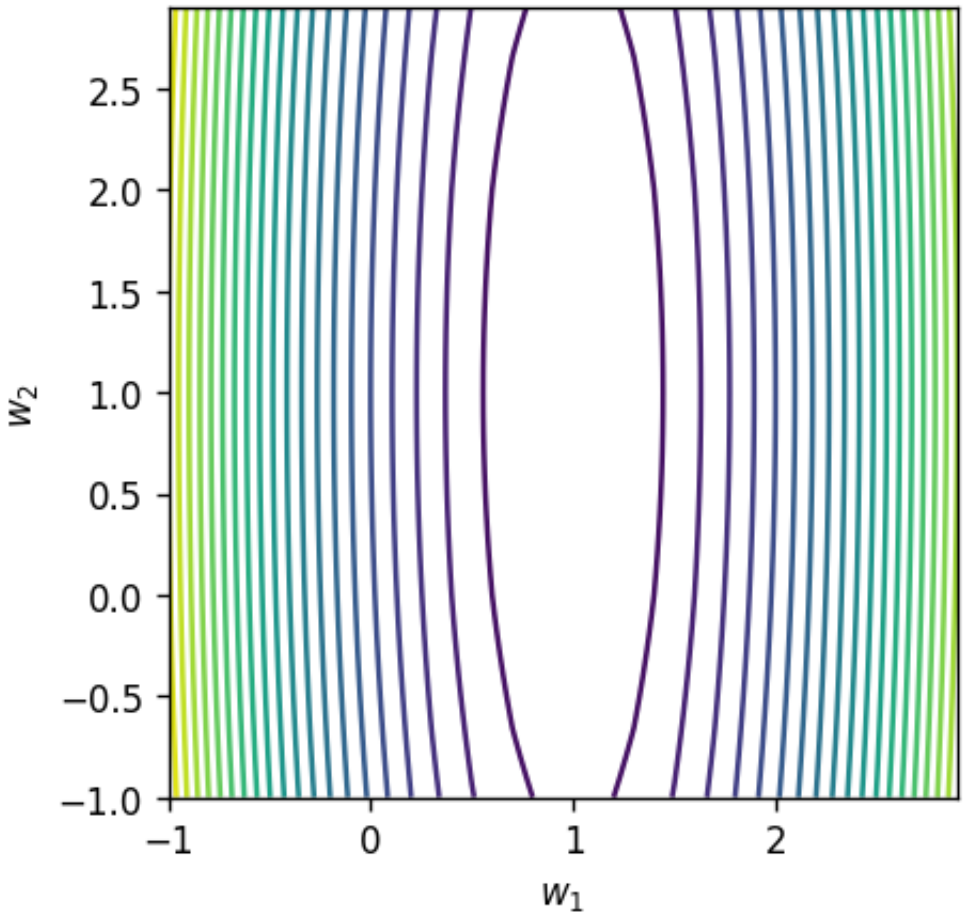


Question 1c)

In [5]:
```
## Copy and paste code from 1b
U = np.array([[1, 0], [0, 1], [0, 0], [0, 0]])
S = np.array([[1, 0], [0, 1/5]])
Sinv = np.linalg.inv(S)
V = np.eye(2)
X = U @ S @ V.T
y = np.array([[1], [1/5], [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');
```

```
/var/folders/tn/v9tpvrrs4qgdbw0xd1q0l8qh0000gn/T/ipykernel_82298/2403673536.
py:20: DeprecationWarning: Conversion of an array with ndim > 0 to a scalar
is deprecated, and will error in future. Ensure you extract a single element
from your array before performing this operation. (Deprecated NumPy 1.25.)
  fw[i,j] = (w-w_ls).T @ X.T @ X @ (w-w_ls) + c
```

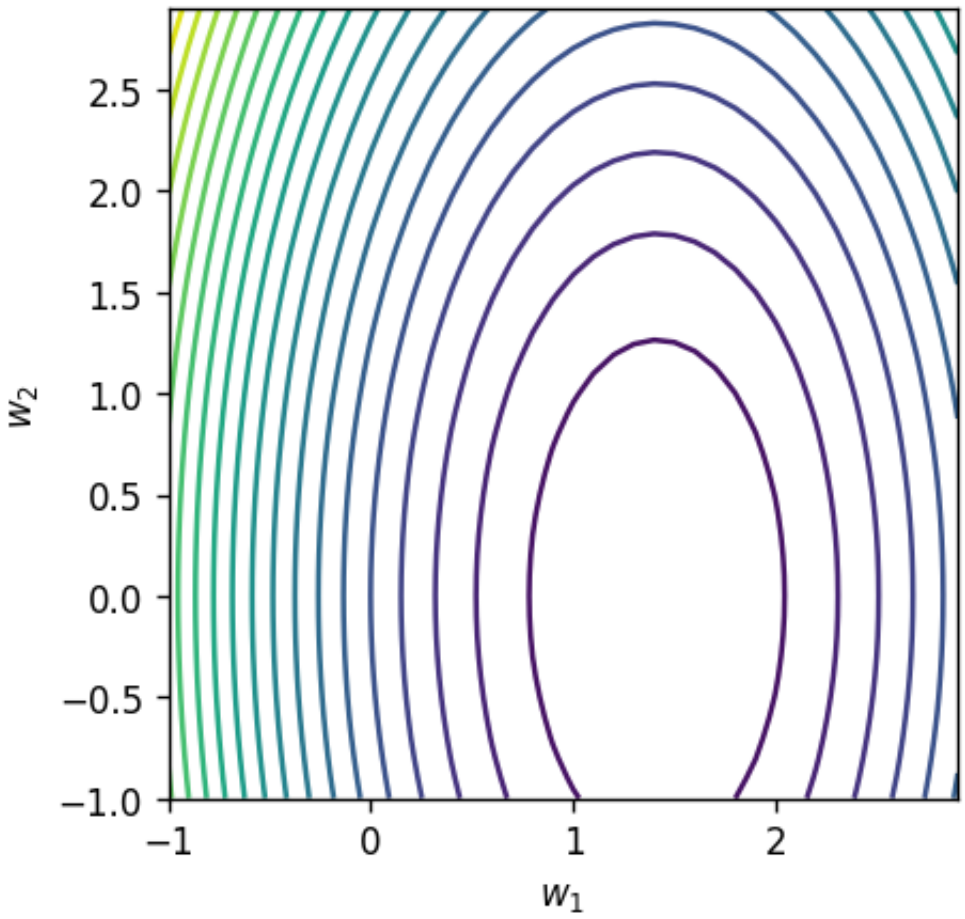


Question 1d)

In [6]:
```
## Copy and paste code from 1b
## Copy and paste code from 1b
U = np.array([[1, 0], [0, 1], [0, 0], [0, 0]])
```

```
S = np.array([[1, 0], [0, 1/2]])
Sinv = np.linalg.inv(S)
V = np.eye(2)
X = U @ S @ V.T
y = np.array([[np.sqrt(2)], [0], [0], [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');
```

```
/var/folders/tn/v9tpvrrs4qgdbw0xd1q0l8qh0000gn/T/ipykernel_82298/3114011275.
py:21: DeprecationWarning: Conversion of an array with ndim > 0 to a scalar
is deprecated, and will error in future. Ensure you extract a single element
from your array before performing this operation. (Deprecated NumPy 1.25.)
  fw[i,j] = (w-w_ls).T @ X.T @ X @ (w-w_ls) + c
```



Question 1-e

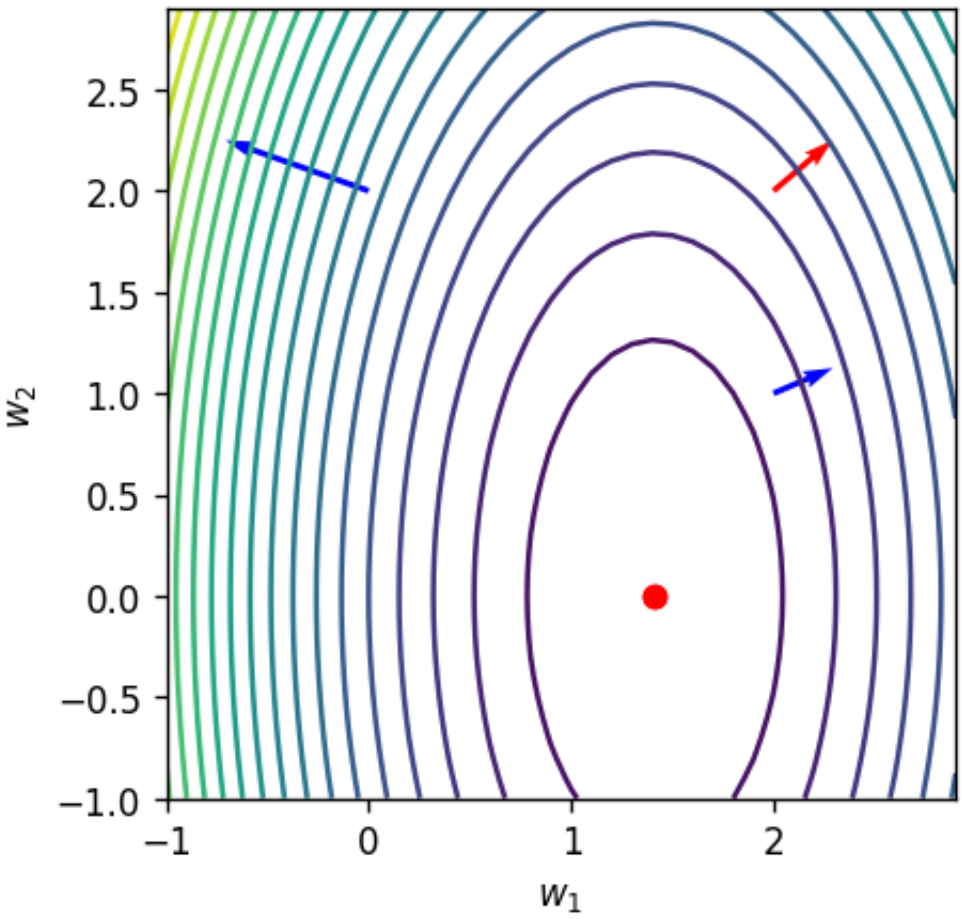
In [24]:

```
def gradient(A, d, W):
    return .5 *  A.T @ (A @ W - d)

## Copy and paste code from 1b
## Copy and paste code from 1b
U = np.array([[1, 0], [0, 1], [0, 0], [0, 0]])
S = np.array([[1, 0], [0, 1/2]])
Sinv = np.linalg.inv(S)
V = np.eye(2)
X = U @ S @ V.T
y = np.array([[np.sqrt(2)], [0], [0], [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.plot(w_ls[0],w_ls[1],'ro')
w_0i = np.array([[2], [2]])
w_0ii = np.array([[0], [2]])
w_0iii = np.array([[2], [1]])

grad_1 = gradient(X, y, w_0i)
grad_2 = gradient(X, y, w_0ii)
grad_3 = gradient(X, y, w_0iii)

plt.quiver(w_0i[0], w_0i[1], grad_1[0], grad_1[1], angles='xy', scale_units=
plt.quiver(w_0ii[0], w_0ii[1], grad_2[0], grad_2[1], angles='xy', scale_unit
plt.quiver(w_0iii[0], w_0iii[1], grad_3[0], grad_3[1], angles='xy', scale_ur

plt.plot(w_ls[0],w_ls[1],'ro')

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')
;
```

```
/var/folders/tn/v9tpvrrs4qgdbw0xd1q0l8qh0000gn/T/ipykernel_82298/1277983476.
py:24: DeprecationWarning: Conversion of an array with ndim > 0 to a scalar
is deprecated, and will error in future. Ensure you extract a single element
from your array before performing this operation. (Deprecated NumPy 1.25.)
  fw[i,j] = (w-w_ls).T @ X.T @ X @ (w-w_ls) + c
```

Out[24]: ''



Question 2-a

The max size is $\tau < \frac{2}{||A||_{op}^2}$

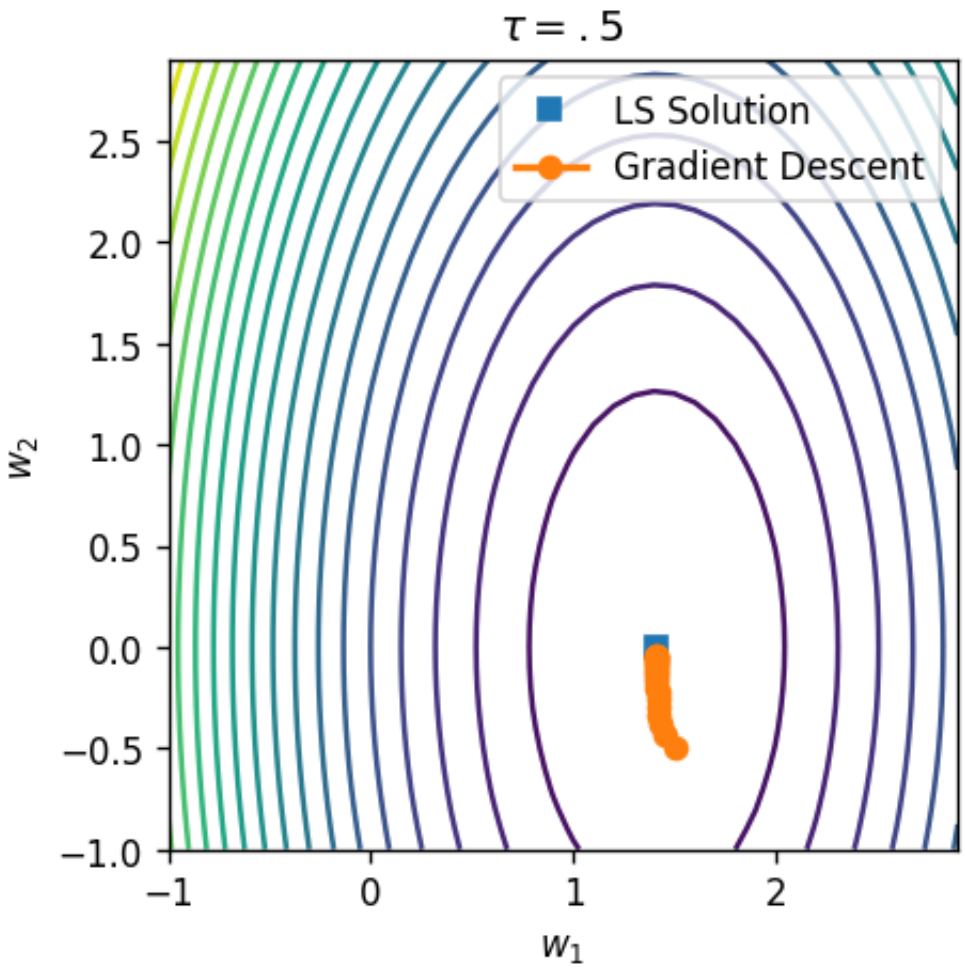
Question 2b)

In [22]:
```
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
```

```
/var/folders/tn/v9tpvrrs4qgdbw0xd1q0l8qh0000gn/T/ipykernel_82298/3366260239.
py:19: DeprecationWarning: Conversion of an array with ndim > 0 to a scalar
is deprecated, and will error in future. Ensure you extract a single element
from your array before performing this operation. (Deprecated NumPy 1.25.)
  fw[i,j] = (w-w_ls).T @ X.T @ X @ (w-w_ls) + c
```

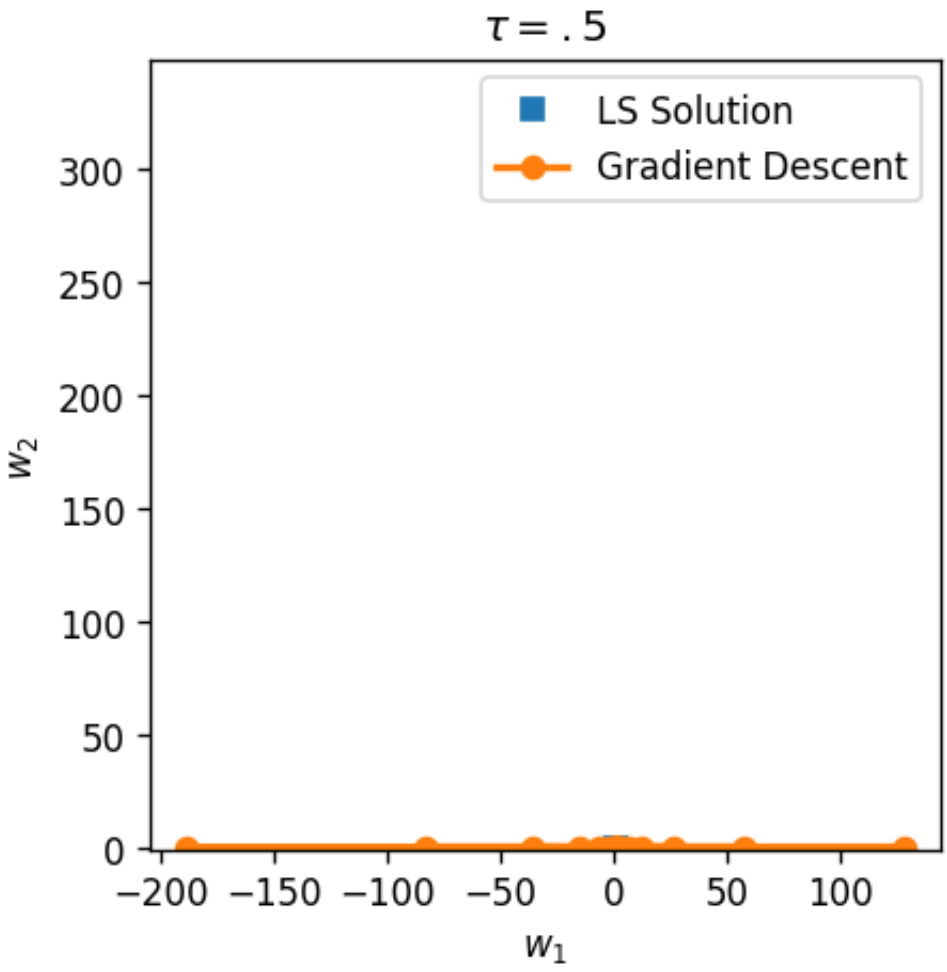
In [25]:
```
w_init = np.array([[1.5], [-0.5]])  # complete this line with a 2x1 numpy ar
it = 20
tau = .5
W = graddescent(X,y,tau,w_init,it);


### 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');
```



Question 2c)

In [ ]:

```
# copy and paste code from above
w_init = np.array([[1.5], [-0.5]])  # complete this line with a 2x1 numpy ar
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'$\tau = 2.5$');
plt.axis('square');
```

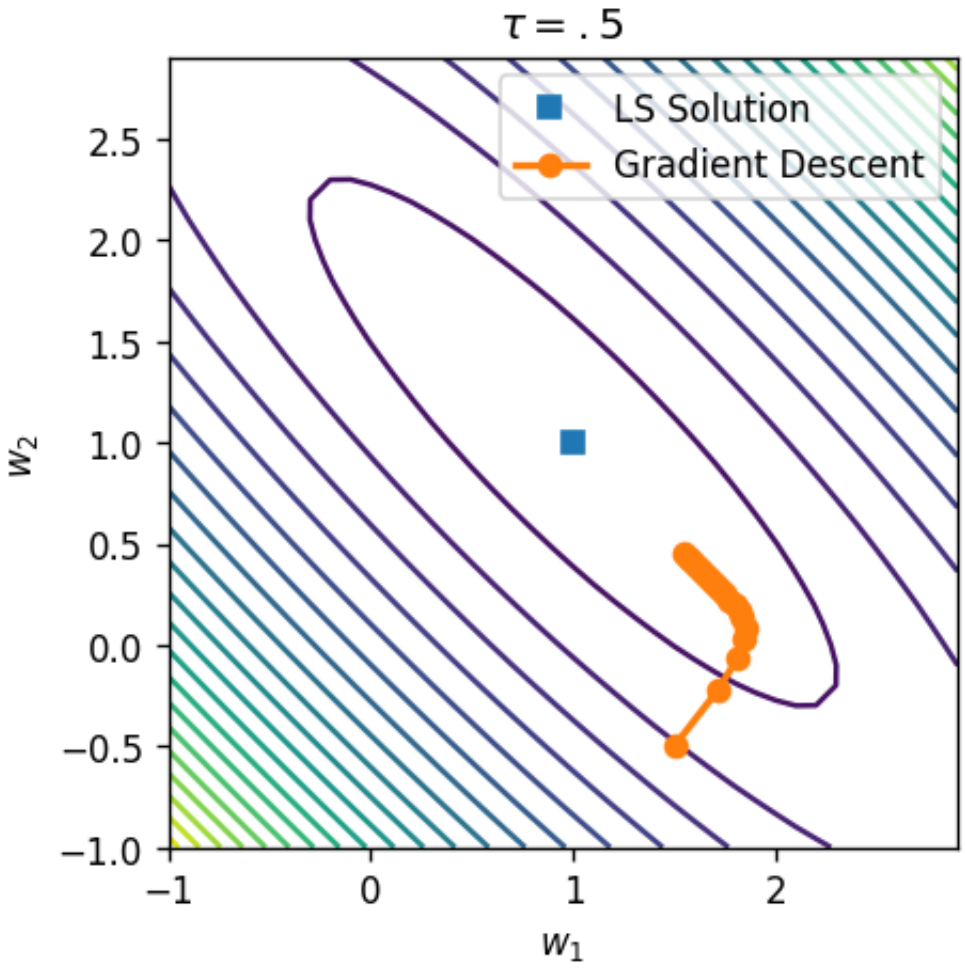


Question 2d)

In [29]:
```
## Copy and paste code from above
U = np.array([[1, 0], [0, 1], [0, 0], [0, 0]])
S = np.array([[1, 0], [0, 1/4]])
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
```

```
w_init = np.array([[1.5], [-0.5]])  # complete this line with a 2x1 numpy ar
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');
```

```
/var/folders/tn/v9tpvrrs4qgdbw0xd1q0l8qh0000gn/T/ipykernel_82298/1085480139.
py:20: DeprecationWarning: Conversion of an array with ndim > 0 to a scalar
is deprecated, and will error in future. Ensure you extract a single element
from your array before performing this operation. (Deprecated NumPy 1.25.)
  fw[i,j] = (w-w_ls).T @ X.T @ X @ (w-w_ls) + c
```



The cost function is rotated

Questoin 2-e

For fast gradient descent, a lower singular value ratio is ideal. When the ratio is high, gradient descent struggles and is slow