CS/ECE/ME532 Period 16 Activity

Estimated Time: 40 minutes for Q1 and 30 minutes for Q2.

1. The squared-error cost function $f(\boldsymbol{w}) = ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}||_2^2 = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})$ may be rewritten as a perfect square in the form

$$f(\boldsymbol{w}) = (\boldsymbol{w} - \boldsymbol{w}_{LS})^T \boldsymbol{X}^T \boldsymbol{X} (\boldsymbol{w} - \boldsymbol{w}_{LS}) + c$$

where $\boldsymbol{w}_{LS} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$ and $c = \boldsymbol{y}^T\boldsymbol{y} - \boldsymbol{y}^T\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$. This assumes the n-by-p ($p < n$) matrix $\boldsymbol{X}$ is full rank. $f(\boldsymbol{w})$ is called a "quadratic form" in $\boldsymbol{w}$ since it is a quadratic function of $\boldsymbol{w}$.

a) Prove that the minimum value of $f(\boldsymbol{w}) = c$ when $\boldsymbol{w} = \boldsymbol{w}_{LS}$ and all other $\boldsymbol{w}$ result in higher values of $f(\boldsymbol{w})$.

b) Suppose $\boldsymbol{y} = \begin{bmatrix} 1 \\ 1/2 \\ 1 \\ 0 \end{bmatrix}$ and the 4-by-2 $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ has singular value decomposition $\boldsymbol{U} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$, $\boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0 \\ 0 & 1/2 \end{bmatrix}$, and $\boldsymbol{V} = \boldsymbol{I}$. Sketch a contour plot of $f(\boldsymbol{w})$ in the w_1-w_2 plane.

c) Suppose $\boldsymbol{y} = \begin{bmatrix} 1 \\ 1/5 \\ 1 \\ 0 \end{bmatrix}$, $\boldsymbol{U} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$, $\boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0 \\ 0 & 1/5 \end{bmatrix}$, and $\boldsymbol{V} = \boldsymbol{I}$. Sketch a contour plot of $f(\boldsymbol{w})$ in the w_1-w_2 plane. How do the singular values of $\boldsymbol{X}$ affect the shape of the contours?

d) In this case assume $\boldsymbol{y} = \begin{bmatrix} \sqrt{2} \\ 0 \\ 1 \\ 0 \end{bmatrix}$, $\boldsymbol{U} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$, $\boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0 \\ 0 & 1/2 \end{bmatrix}$, and $\boldsymbol{V} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$. Sketch a contour plot of $f(\boldsymbol{w})$ in the w_1-w_2 plane. How do the right singular vectors of $\boldsymbol{X}$ affect the contours?

e) Sketch the gradient of $f(\boldsymbol{w})$ at $\boldsymbol{w} = \boldsymbol{w}_o$ on the contour plot for the previous case when

i. $\boldsymbol{w}_o = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$

ii. $\boldsymbol{w}_o = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$

iii. $\boldsymbol{w}_o = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$

SOLUTION:

a) Note that since $\boldsymbol{w}_{LS} - \boldsymbol{w}_{LS} = 0$, then $f(\boldsymbol{w}_{LS}) = c$. Furthermore, $\boldsymbol{X}^T\boldsymbol{X}$ is positive definite since $\boldsymbol{X}$ is full rank. This implies $\boldsymbol{g}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{g} > 0$ for all $\boldsymbol{g} \neq 0$, so $f(\boldsymbol{w}) > c$ for all $\boldsymbol{w} \neq \boldsymbol{w}_{LS}$.

b) Note that in all cases below $\boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{\Sigma}^2$ so the contours are ellipses with ratio of major to minor axis given by σ_1/σ_2. The ellipse is centered at $\boldsymbol{w}_{LS}$, which in all cases below is $\boldsymbol{w}_{LS} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

c) The contours become more eccentric as the ratio of singular values increases.

d) The columns of $\boldsymbol{V}$ define the orientation of the major and minor axes.

e) The gradients are always normal to the contours at the point.

2. The provided script will compute a specified number of iterations of the gradient descent algorithm and help you display the path taken by the weights in the gradient descent iteration superimposed on a contour plot. Assume $\boldsymbol{y} = \begin{bmatrix} \sqrt{2} \\ 0 \\ 1 \\ 0 \end{bmatrix}$, the 4-by-2 $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ has singular value decomposition $\boldsymbol{U} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$, $\boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0 \\ 0 & 1/2 \end{bmatrix}$, and $\boldsymbol{V} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$. Complete 20 iterations of gradient descent in each case specified below.

Include the plots you generate below with your submission.

a) What is the maximum value for the step size τ that will guarantee convergence?

b) Start gradient descent from the point $\boldsymbol{w} = \begin{bmatrix} 1.5 \\ -0.5 \end{bmatrix}$ and use a step size of $\tau = 0.5$. How do you explain the trajectory the weights take toward the optimum, e.g., why is it shaped this way? What would the trajectory be if you started from the point $\boldsymbol{w} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$? From $\boldsymbol{w} = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$?

c) Start gradient descent from the point $\boldsymbol{w} = \begin{bmatrix} 1.5 \\ -0.5 \end{bmatrix}$ and use a step size of $\tau = 2.5$. Complete 20 iterations. What happens and why?

d) Now change $\boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0 \\ 0 & 1/4 \end{bmatrix}$, start from $\boldsymbol{w} = \begin{bmatrix} 1.5 \\ -0.5 \end{bmatrix}$ and use $\tau = 0.5$. What happens to the cost function? How does the change in the cost function lead to a change in the trajectory of the gradient descent weights? What would happen if you further decreased the smaller singular value?

e) Discuss how changing the ratio of the singular values changes the shape of the cost function and how that might affect the number of iterations it takes for gradient descent to get close to the optimum.

SOLUTION:

a) Maximum step size is $\tau < 2/\sigma_1^2 = 2$. Depending on how the gradient descent step is defined (if the factor of two is included with step size or not), we can have $\tau < 1/\sigma_1^2 = 1$

b) Trajectory always follows gradient. If one starts on the major or minor axis, the iterations proceed directly to the solution.

c) The step size is too large and the steps cause us to diverge. Each iteration is higher up the side of the bowl than the previous.

d) The surface becomes more eccentric. The trajectories proceed in the steepest direction (minor axis) first, and then will approach the solution along the major axis. This effect becomes more dramatic as the smaller singular value further decreases.

e) The cost function becomes steeper along the minor axis and (relatively) shallow along the major axis as the ratio of largest to smallest singular values increases.

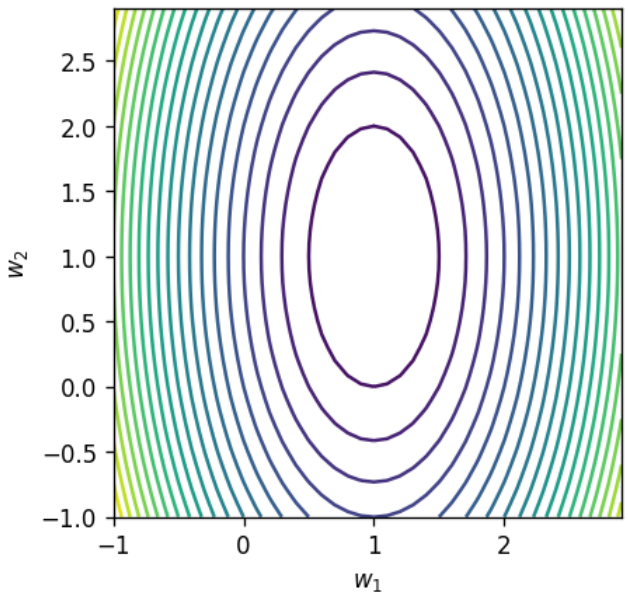
It can take many iterations to converge along the major axis direction because it is so shallow.

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:
```
## 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)

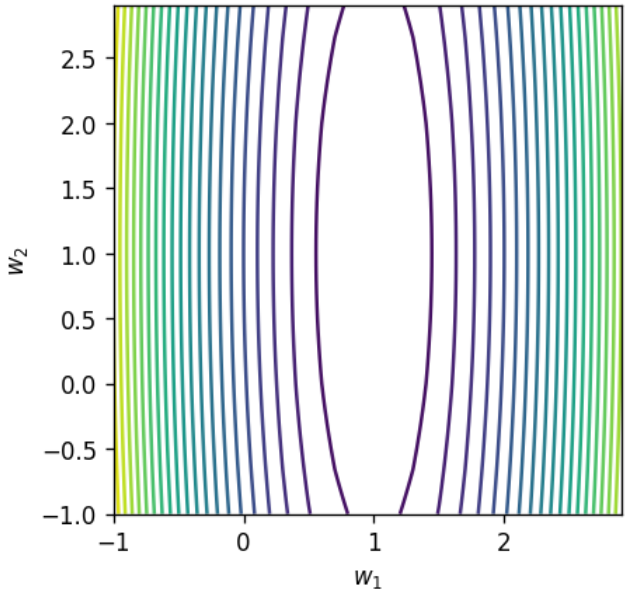
In [4]:
```
U = np.array([[1, 0], [0, 1], [0, 0], [0, 0]])
S = np.array([[1, 0], [0, 0.5]])
Sinv = np.linalg.inv(S)
V = np.eye(2)
X = U @ S @ V.T
y = np.array([[1], [0.5], [1], [0]])

### 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');

```



Question 1c)

In [5]:

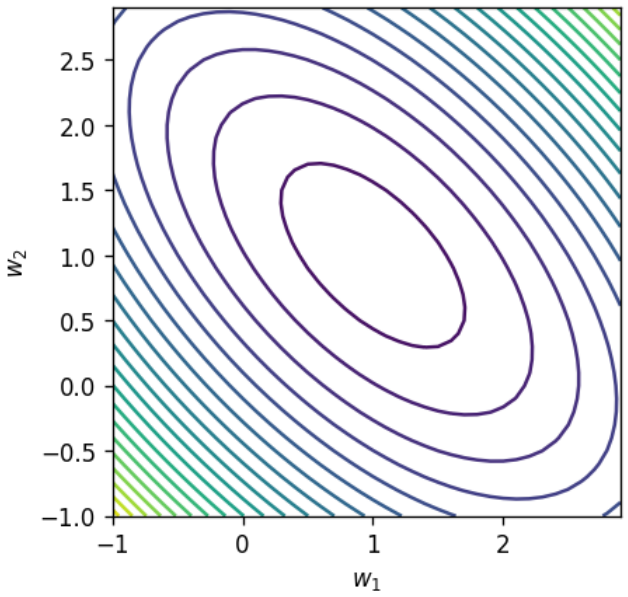
```
U = np.array([[1, 0], [0, 1], [0, 0], [0, 0]])
S = np.array([[1, 0], [0, 0.2]])
Sinv = np.linalg.inv(S)
V = np.eye(2)
X = U @ S @ V.T
y = np.array([[1], [0.2], [1], [0]])

### 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');

```



Question 1d)

In [8]:

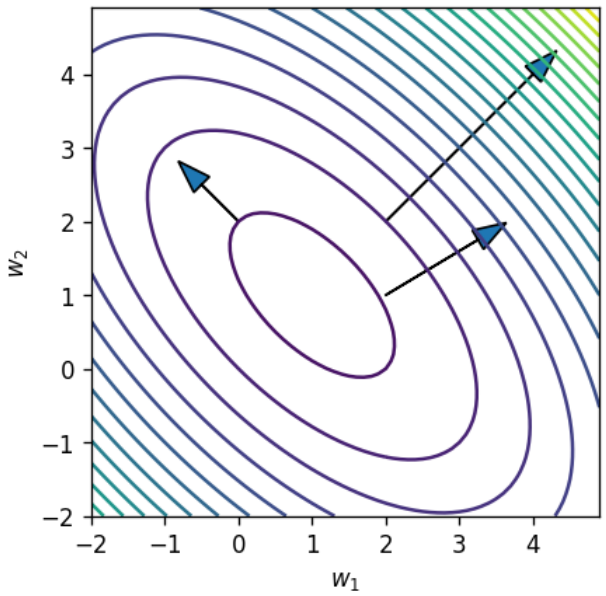
```
U = np.array([[1, 0], [0, 1], [0, 0], [0, 0]])
S = np.array([[1, 0], [0, 0.5]])
Sinv = np.linalg.inv(S)
V = 1/np.sqrt(2)*np.array([[1,1],[1,-1]])
X = U @ S @ V.T
y = np.array([[np.sqrt(2)], [0], [1], [0]])

### 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(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


w01 = np.array([[2],[2]])
grad1 = 2*X.T @ (X @ w01 - y)

w02 = np.array([[2],[2]])
grad2 = 2*X.T @ (X @ w02 - y)

w03 = np.array([[2],[2]])
grad3 = 2*X.T @ (X @ w03 - y)

### Plot the countours
plt.figure(num=None, figsize=(4, 4), dpi=120)
plt.contour(w1,w2,fw,20)
plt.xlim([-4,7])
plt.ylim([-4,7])
plt.xlabel('$w_1$')
plt.ylabel('$w_2$')
plt.axis('square');
```



Question 1e)

In [20]:

```
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(-2,5,.1)
w2 = np.arange(-2,5,.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)

### compute and plot the gradients
w0 = [np.array([[2],[2]]),np.array([[0],[2]]),np.array([[2],[1]])]
for w in w0:
    grad = 2*X.T @ (X @ w - y)
    plt.arrow(w[0,0], w[1,0], grad[0,0], grad[1,0], head_width=.3)

plt.xlim([-4,7])
plt.ylim([-4,7])
plt.xlabel('$w_1$')
plt.ylabel('$w_2$')
plt.axis('square');
```



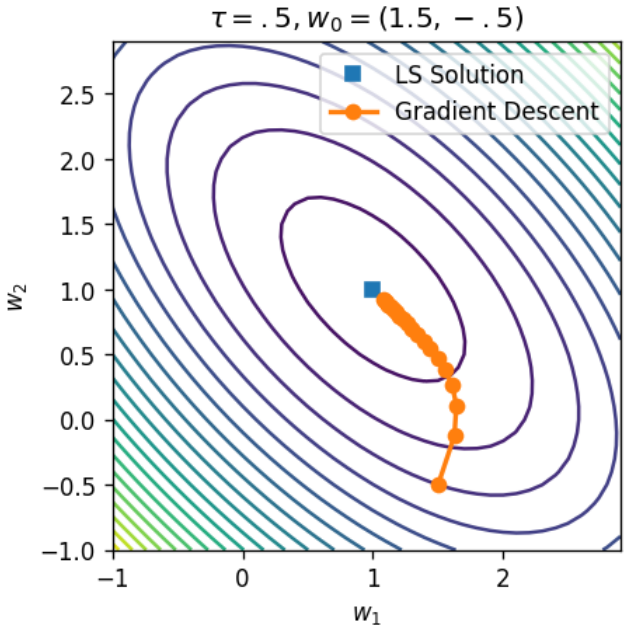
Question 2b)

In [55]:

```
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
```

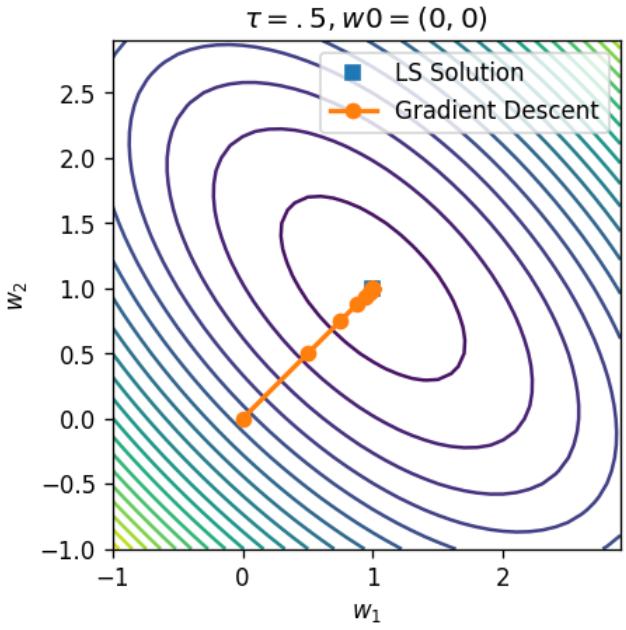
In [62]:

```
w_init = np.array([[1.5],[-.5]])  # complete this line with a 2x1 numpy array for the values specified in the
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'$\tau = .5, w_0 = (1.5, -.5)$');
plt.axis('square');
```



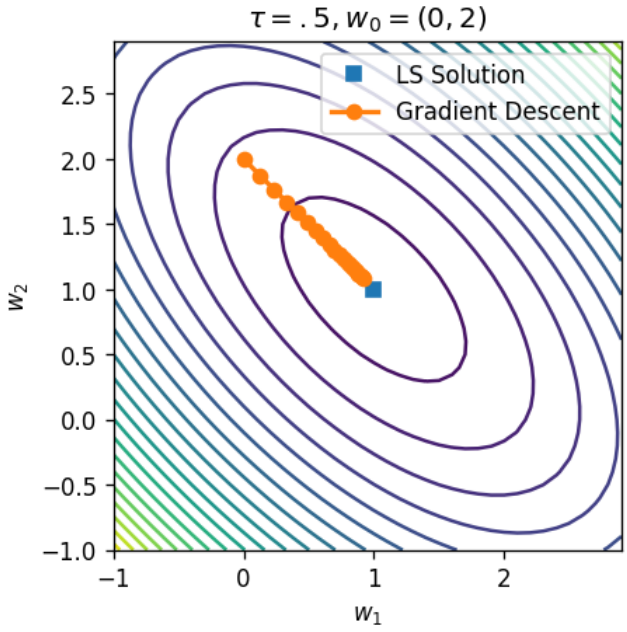
In [63]:

```
w_init = np.array([[0],[0]])  # complete this line with a 2x1 numpy array for the values specified in the act
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'$\tau = .5, w0 = (0,0)$');
plt.axis('square');
```



In [64]:

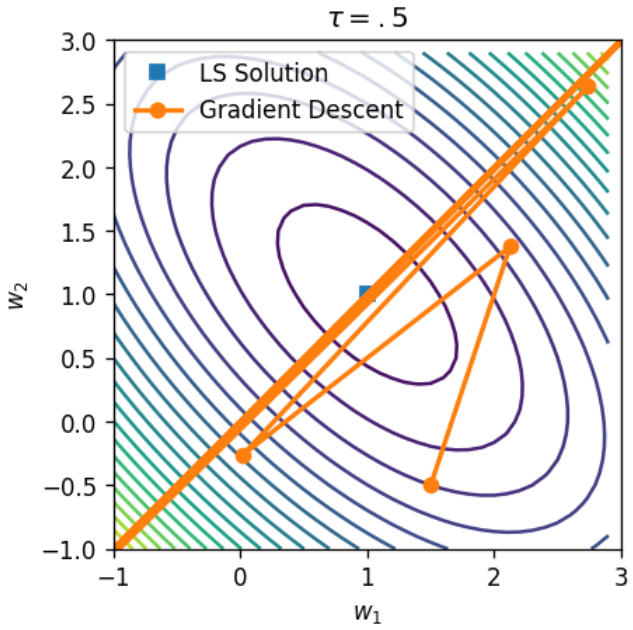
```
w_init = np.array([[0],[2]])  # complete this line with a 2x1 numpy array for the values specified in the act
it = 20
tau = .5
W = graddescent(X,y,tau,w_init,it);


### 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'$\tau = .5, w_0 = (0,2)$');
plt.axis('square');
```



Question 2c)

In [66]:

```
w_init = np.array([[1.5],[-.5]])  # complete this line with a 2x1 numpy array for the values specified in the
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.axis('square');
plt.xlim([-1,3])
plt.xlabel('$w_1$')
plt.ylim([-1,3])
plt.ylabel('$w_2$')
plt.title(r'$\tau = .5$');
```



Question 2d)

In [67]:

```
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
```

In [68]:

```
w_init = np.array([[1.5],[-.5]])  # complete this line with a 2x1 numpy array for the values specified in the
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'$\tau = .5, w_0 = (1.5, -.5)$');
plt.axis('square');
```