Math 4650/MSSC 5650 - Homework 6

Henrique Medeiros Dos Reis

Spring 2023

Problem 1 (5 pts)**.** Let $f : \mathbb{R}^2 \to \mathbb{R}$ be the function $f(x, y) = x^2 + xy + y^2$. Find *two different* descent directions for f at the point $(x, y) = (1, -1)$.

Solution 1.

$$\nabla f(x, y) = \begin{bmatrix} 2x + y \\ x + 2y \end{bmatrix}$$

$$\nabla f(1, -1) = \begin{bmatrix} 2 + (-1) \\ 1 + 2(-1) \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

For it to be a descent direction, we need $\nabla f(x, y)^T d < 0$

$$\begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} = d_1 - d_2$$

So $d = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$ and $d = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ are descent directions since $\begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \end{bmatrix} = -2 < 0$ and $\begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = -1 < 0$

Problem 2 (5 pts)**.** Let $f : \mathbb{R}^2 \to \mathbb{R}$ be the function $f(x, y) = \frac{1}{2}x^2 + y$. Suppose we take $(x_0, y_0) = (1, -1)$ to be our initial iterate, and $(d_x, d_y) = -\nabla f(x_0, y_0) = (-1, -1)$ to be our descent direction. Find the next iterate (x_1, y_1) computed using the descent direction method with exact line search. Verify that $f(x_1, y_1) < f(x_0, y_0)$ by direct computation.

Solution 2.

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + t_0 d_0 = \begin{bmatrix} 1 \\ -1 \end{bmatrix} + t_0 \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

$$t_0 = argmin_{t \geq 0} f((x_0, y_0) + t_0 d_0)$$

$$\Rightarrow min_{t \geq 0} f\left(\begin{bmatrix} 1 - 1t \\ -1 - t \end{bmatrix}\right) = min_{t \geq 0}\left(\frac{1}{2}(1 - t)^2 + (-1 - t)\right)$$

And lets call the function inside the min $g(x)$ and set it equal to 0 in order to find the minimizer

$$g'(x) = (1 - t)(-1) - 1 = t - 2 = 0 \Rightarrow t = t_0 = 2$$

Then

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} + 2\begin{bmatrix} -1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1-2 \\ -1-2 \end{bmatrix} = \begin{bmatrix} -1 \\ -3 \end{bmatrix}$$

Finally, we can verify $f(x_1, y_1) < f(x_0, y_0)$

$$f(-1,-3) = \frac{1}{2}(-1)^2 - 3 = \frac{-5}{2} < f(1,-1) = \frac{1}{2}(1)^2 - 1 = \frac{-1}{2}$$

Problem 3 (5 pts)**.** Let $f : \mathbb{R}^n \to \mathbb{R}$ be a general quadratic function $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A}\mathbf{x} + 2\mathbf{b}^\top \mathbf{x} + c$ where $\mathbf{A} \in \mathbb{R}^{n\times n}$ is positive definite, and $\mathbf{b} \in \mathbb{R}^n$ and $c \in \mathbb{R}$ are arbitrary. Fix an $\mathbf{x} \in \mathbb{R}^n$ and suppose $\mathbf{d} \in \mathbb{R}^d$ is a descent direction of f at $\mathbf{x}$, i.e., $\mathbf{d}^\top \nabla f(\mathbf{x}) < 0$. Show that the exact line search step-size t^*, defined by

$$t^* = \operatorname*{arg\,min}_{t\geq 0} f(\mathbf{x} + t\mathbf{d}),$$

has the explicit form

$$t^* = -\frac{\mathbf{d}^\top(\mathbf{A}\mathbf{x} + \mathbf{b})}{\mathbf{d}^\top \mathbf{A}\mathbf{d}}.$$

In your derivation, where is the assumption that $\mathbf{A}$ is positive definite being used? (Hint: If you get stuck, this is worked out in Example 4.4 in Beck, Ch. 4; but please put the proof in your own words).

Solution 3. Optional

Problem 4 (MATLAB, 5 pts)**.** Consider the quadratic function $f : \mathbb{R}^2 \to \mathbb{R}$ defined by $f(\mathbf{x}) = \frac{1}{2}(\gamma x_1^2 + x_2^2)$ with $\gamma = 0.05$, which has a unique global minimum at $(0,0)$. Using the provided MATLAB script `gd.m` as a starting point[1], implement the following two versions of the decent direction method to minimize this function:

(a) descent direction $\mathbf{d}_k = -\nabla f(\mathbf{x}_k)$ (steepest descent), constant stepsize $t_k = 0.05$

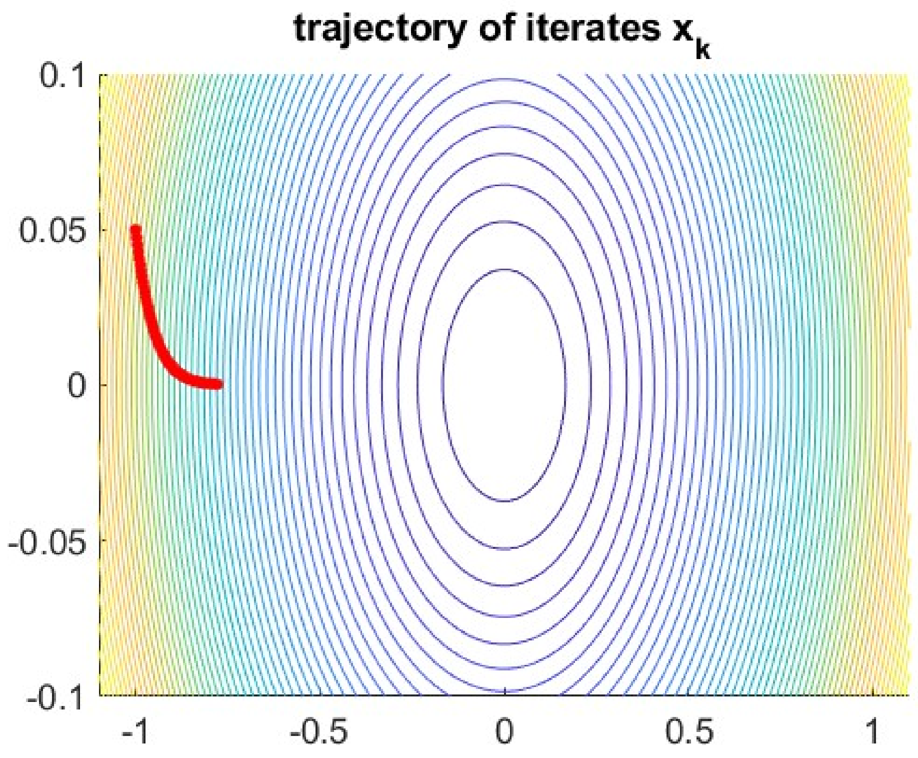
(b) descent direction $\mathbf{d}_k = -\begin{bmatrix} 1/\gamma & 0 \\ 0 & 1 \end{bmatrix} \nabla f(\mathbf{x}_k)$, constant stepsize $t_k = 0.05$.

In both cases, start with the initial guess $\mathbf{x}_0 = \begin{bmatrix} -1 \\ 0.05 \end{bmatrix}$ and run 100 iterations. What do you observe? How well did each method do in finding the minimizer? How do the trajectories of the iterates differ for the two choices of descent directions? In your write-up, include a print-out of your code and plots of the trajectories of the iterates in both cases.

[1]Note: you will need to modify the definition of `f` and `grad` in the script to match this new function.

Solution 4. (a) .

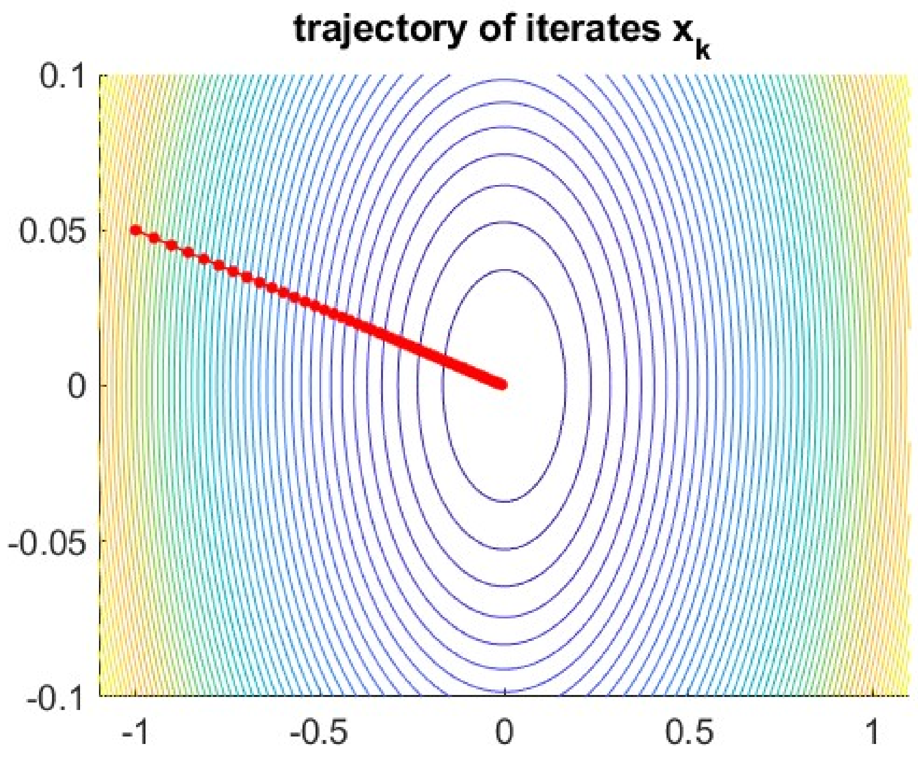
```
gamma = 0.05;
f = @(x) 0.5*(gamma*x(1)^2 + x(2)^2); %define f(x)
grad = @(x) [gamma*x(1);x(2)];        %define grad f(x)

t0 = 0.05;       %constant stepsize
x = [-1;0.05];   %initial point x0
cost = f(x);     %initialize cost array
xar = x;         %initialize array of iterates (for plotting)
for k=1:100
    if norm(grad(x)) < 1e-16
     break;
    end
    d = -grad(x); %steepest descent
    t = t0;
    x = x + t*d;
    cost = [cost,f(x)];
    xar = [xar,x];
end

% plot trajectory of iterates
figure(1);cla;
[X,Y] = meshgrid(linspace(-1.1,1.1,100),linspace
   (-0.1,0.1,100));
Z = zeros(size(X));
for i=1:length(X(:))
    Z(i) = f([X(i),Y(i)]);
end
hold on
contour(X,Y,Z,50);
plot(xar(1,:),xar(2,:),'.-r','markersize',15,'linewidth',1);
hold off;
set(gca,'fontsize',14);
title('trajectory of iterates x_k');
```



(b) .

```
gamma = 0.05;
f = @(x) 0.5*(gamma*x(1)^2 + x(2)^2); %define f(x)
grad = @(x) [gamma*x(1);x(2)];        %define grad f(x)

t0 = 0.05;       %constant stepsize
x = [-1;0.05];   %initial point x0
cost = f(x);     %initialize cost array
xar = x;         %initialize array of iterates (for plotting)
for k=1:100
    if norm(grad(x)) < 1e-16
     break;
    end
    d = -[1/gamma, 0; 0,1]*grad(x);
    t = t0;
    x = x + t*d;
    cost = [cost,f(x)];
    xar = [xar,x];
end

% plot trajectory of iterates
```

```
figure(1);cla;
[X,Y] = meshgrid(linspace(-1.1,1.1,100),linspace
   (-0.1,0.1,100));
Z = zeros(size(X));
for i=1:length(X(:))
    Z(i) = f([X(i),Y(i)]);
end
hold on
contour(X,Y,Z,50);
plot(xar(1,:),xar(2,:),'.-r','markersize',15,'linewidth',1);
hold off;
set(gca,'fontsize',14);
title('trajectory of iterates x_k');
```



From the first plot, it is possible to see that the number of iterates was not enough to reach the minimum value of the function, since the method used was a constant stepsize, the number of iterates would have to be much bigger in order to converge. Also, the direction of the descent is correct and it seems like in a larger number of iterates it would converge to the minimum.

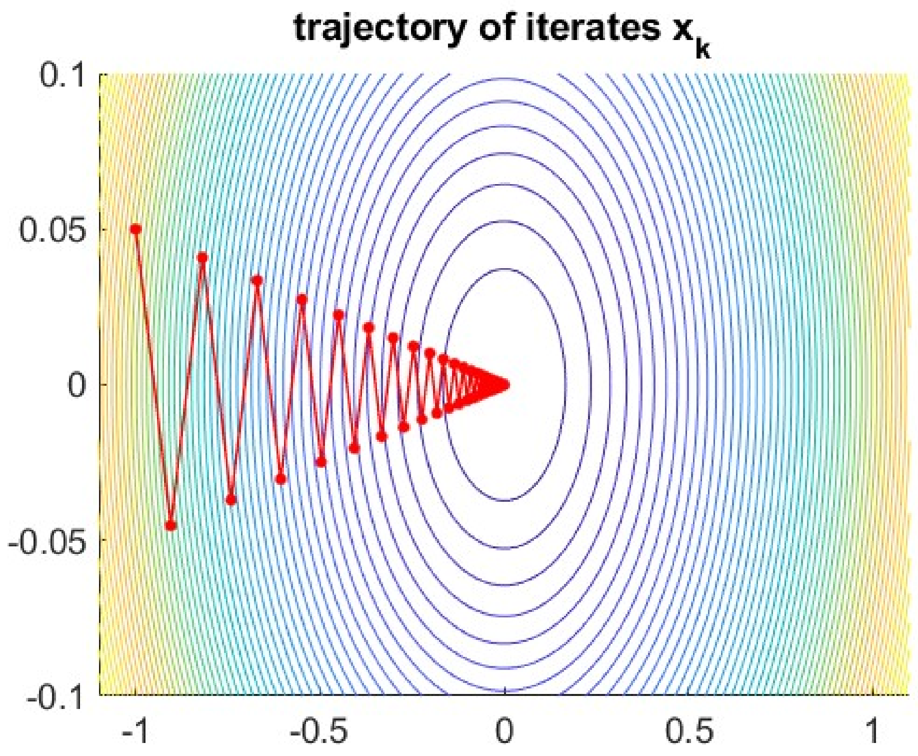
On the second plot, the direction is better from the start, since it is heading to the minimum since the start, due to this fact, we were able to reach the minimum value even with only 100 iterations and a constant stepsize. This direction was a better than the negative gradient, but much more specific for this function.

Problem 5 (MATLAB, 5 pts)**.** Repeat Problem 4 but instead of using a constant stepsize, use *exact line search* to update the stepsize t_k for both versions of the descent direction method. What do you observe? How well did the two versions of the descent direction method do in finding the minimizer? How do the trajectories of the iterates differ for the two versions? In your write-up, include a print-out of your code and plots of the trajectories of the iterates in both cases.
(Hint: Note that $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A}\mathbf{x}$ where $\mathbf{A} = \begin{bmatrix} \gamma/2 & 0 \\ 0 & 1/2 \end{bmatrix}$; now use the result in Problem 3 to compute the exact line search stepsize t_k)

Solution 5. (a) .

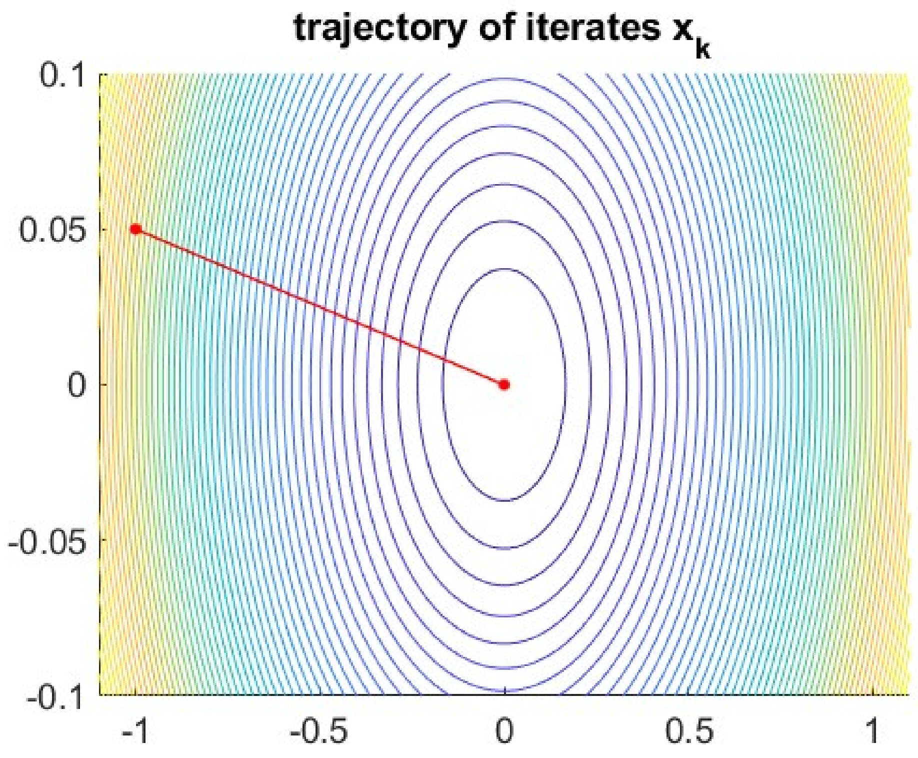
```
gamma = 0.05;
f = @(x) 0.5*(gamma*x(1)^2 + x(2)^2); %define f(x)
grad = @(x) [gamma*x(1);x(2)];        %define grad f(x)

A = [gamma/2, 0; 0, 1/2];
x = [-1;0.05];  %initial point x0
cost = f(x);    %initialize cost array
xar = x;        %initialize array of iterates (for plotting)
for k=1:100
    if norm(grad(x)) < 1e-16
     break;
    end
    d = -grad(x); %steepest descent
    t = -(d'*grad(x))/(2*d'*A*d);  %exact line search for
       quadratic
    x = x + t*d;
    cost = [cost,f(x)];
    xar = [xar,x];
end

% plot trajectory of iterates
figure(1);cla;
[X,Y] = meshgrid(linspace(-1.1,1.1,100),linspace
   (-0.1,0.1,100));
Z = zeros(size(X));
for i=1:length(X(:))
    Z(i) = f([X(i),Y(i)]);
end
hold on
contour(X,Y,Z,50);
```

```
plot(xar(1,:),xar(2,:),'.-r','markersize',15,'linewidth',1);
hold off;
set(gca,'fontsize',14);
title('trajectory of iterates x_k');
```



(b) .

```
gamma = 0.05;
f = @(x) 0.5*(gamma*x(1)^2 + x(2)^2); %define f(x)
grad = @(x) [gamma*x(1);x(2)];         %define grad f(x)

A = [gamma/2, 0; 0, 1/2];
x = [-1;0.05];  %initial point x0
cost = f(x);    %initialize cost array
xar = x;        %initialize array of iterates (for plotting)
for k=1:100
    if norm(grad(x)) < 1e-16
     break;
    end
    d = -[1/gamma, 0; 0,1]*grad(x);
    t = -(d'*grad(x))/(2*d'*A*d);  %exact line search for
       quadratic
    x = x + t*d;
    cost = [cost,f(x)];
    xar = [xar,x];
end
% plot trajectory of iterates
figure(1);cla;
[X,Y] = meshgrid(linspace(-1.1,1.1,100),linspace
   (-0.1,0.1,100));
Z = zeros(size(X));
for i=1:length(X(:))
    Z(i) = f([X(i),Y(i)]);
end
hold on
contour(X,Y,Z,50);
plot(xar(1,:),xar(2,:),'.-r','markersize',15,'linewidth',1);
hold off;
set(gca,'fontsize',14);
title('trajectory of iterates x_k');
```



From the first plot, we were able to converge, however it took all the iterations. This happened because of the “zig zag” while searching for the minimum. From the start the directions that the iterations go are always decreasing, but it is not going exactly to the minimum every time, so it used travels a little more than necessary.

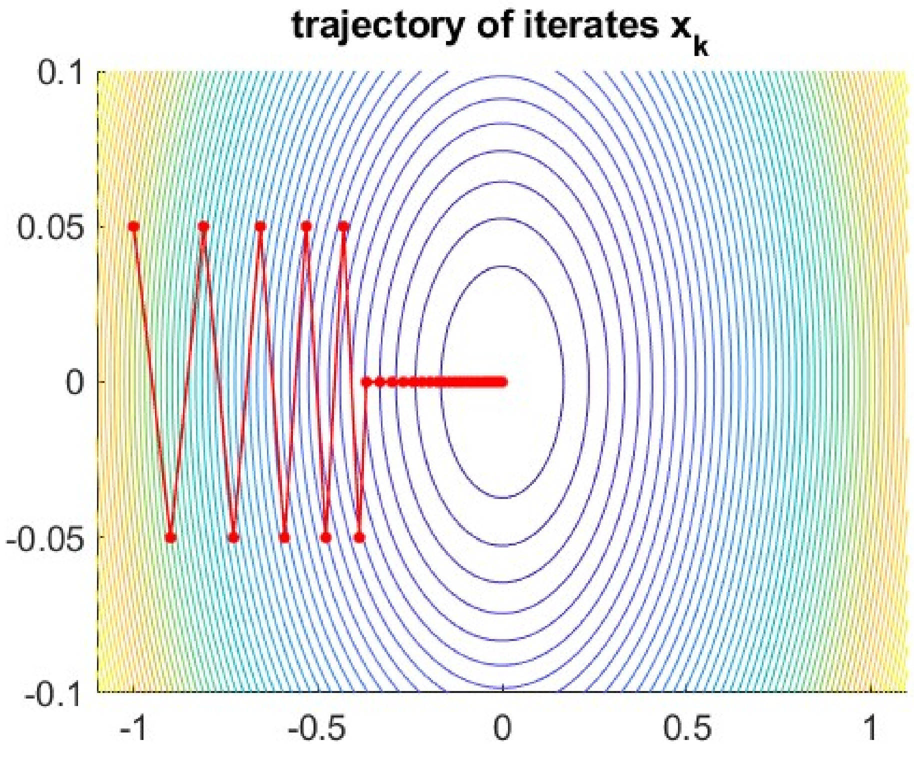
On the second plot, the direction is better from the start, since it is heading to the minimum since the start, due to this fact, we were able to reach the minimum value with only 2 iterations. Since we started looking at the correct direction, the exact line search took care to find the minimum with only one step from there. This direction was a better than the negative gradient, but much more specific for this function.

Problem 6 (**MSSC**, MATLAB, 5 pts)**.** Repeat Problem 4 but instead of using a constant stepsize, use *backtracking line search* to update the stepsize t_k for both versions of the descent direction method. Use the parameters $s = 2$, $\alpha = 0.25$, $\beta = 0.5$. What do you observe? How well did the two versions of the descent direction method do in finding the minimizer? How do the trajectories of the iterates differ for the two versions? In your write-up, include a print-out of your code and plots of the trajectories of the iterates in both cases. (Hint: See Example 4.9 in Beck Ch. 4 for an example implementation of backtracking line search)

Solution 6. (a) .

```
gamma = 0.05;
f = @(x) 0.5*(gamma*x(1)^2 + x(2)^2); %define f(x)
grad = @(x) [gamma*x(1);x(2)];         %define grad f(x)

A = [gamma/2, 0; 0, 1/2];
x = [-1;0.05];  %initial point x0
cost = f(x);    %initialize cost array
xar = x;        %initialize array of iterates (for plotting)
% Set initial parameters
s=2;
alpha=0.25;
beta=0.5;
for k=1:100
    if norm(grad(x)) < 1e-16
     break;
    end
    d = -grad(x); %steepest descent
    t=s;
    while (f(x)-f(x+t*d)<alpha*norm(grad(x))^2)
        t=beta*t;
    end

    x = x + t*d;
    cost = [cost,f(x)];
    xar = [xar,x];
```

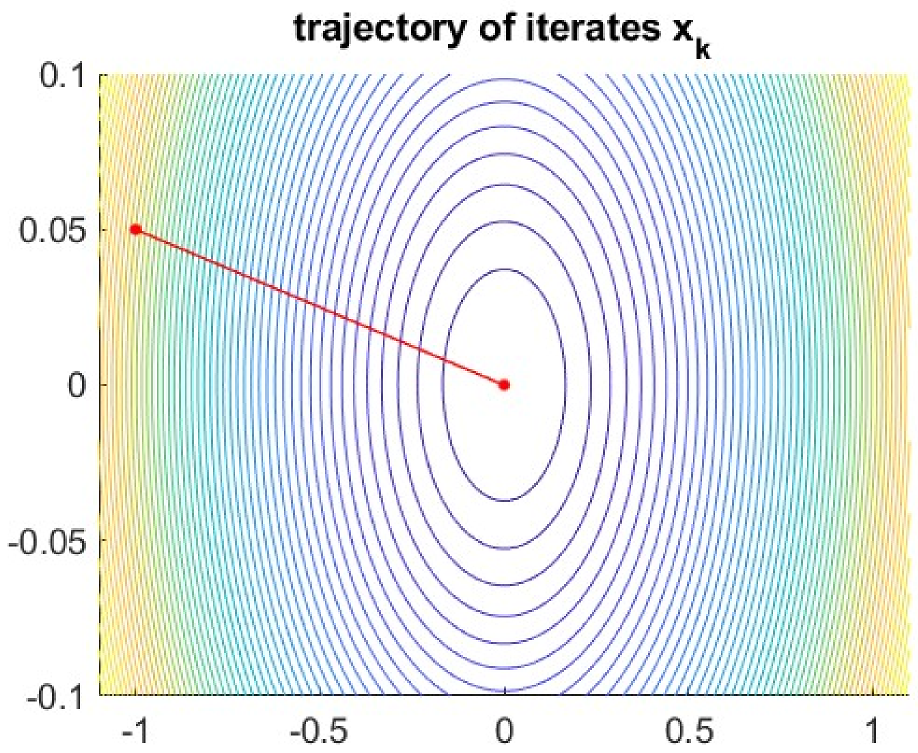
```
end

% plot trajectory of iterates
figure(1);cla;
[X,Y] = meshgrid(linspace(-1.1,1.1,100),linspace
   (-0.1,0.1,100));
Z = zeros(size(X));
for i=1:length(X(:))
    Z(i) = f([X(i),Y(i)]);
end
hold on
contour(X,Y,Z,50);
plot(xar(1,:),xar(2,:),'.-r','markersize',15,'linewidth',1);
hold off;
set(gca,'fontsize',14);
title('trajectory of iterates x_k');
```



(b) .

```
gamma = 0.05;
f = @(x) 0.5*(gamma*x(1)^2 + x(2)^2); %define f(x)
grad = @(x) [gamma*x(1);x(2)];         %define grad f(x)
```

```
A = [gamma/2, 0; 0, 1/2];
x = [-1;0.05];  %initial point x0
cost = f(x);    %initialize cost array
xar = x;        %initialize array of iterates (for plotting)
% Set initial parameters
s=2;
alpha=0.25;
beta=0.5;
for k=1:100
    if norm(grad(x)) < 1e-16
     break;
    end
    d = -[1/gamma, 0; 0,1]*grad(x);
    t=s;
    while (f(x)-f(x+t*d)<alpha*norm(grad(x))^2)
        t=beta*t;
    end
    x = x + t*d;
    cost = [cost,f(x)];
    xar = [xar,x];
end
% plot trajectory of iterates
figure(1);cla;
[X,Y] = meshgrid(linspace(-1.1,1.1,100),linspace
   (-0.1,0.1,100));
Z = zeros(size(X));
for i=1:length(X(:))
    Z(i) = f([X(i),Y(i)]);
end
hold on
contour(X,Y,Z,50);
plot(xar(1,:),xar(2,:),'.-r','markersize',15,'linewidth',1);
hold off;
set(gca,'fontsize',14);
title('trajectory of iterates x_k');
```



From the first plot, we were able to converge, however, once again it took all the iterations. This happened because of the "zig zag" while searching for the minimum. From the start the directions that the iterations go are always decreasing, but it is not going exactly to the minimum every time, so it used travels a little more than necessary. In fact, the first few iterations just bounced back and forth between 0.05 and -0.05, until it reached a point where it started to decrease values from x and y to reach the minimum value at $0, 0$.

On the second plot, the direction is better from the start, since it is heading to the minimum since the start, due to this fact, we were able to reach the minimum value with only 2 iterations. Since we started looking at the correct direction, the backtracking line search took care to find the minimum with only one step from there. This direction was a better than the negative gradient, but much more specific for this function.