In the name of God

Blind Source Separation

HW #2

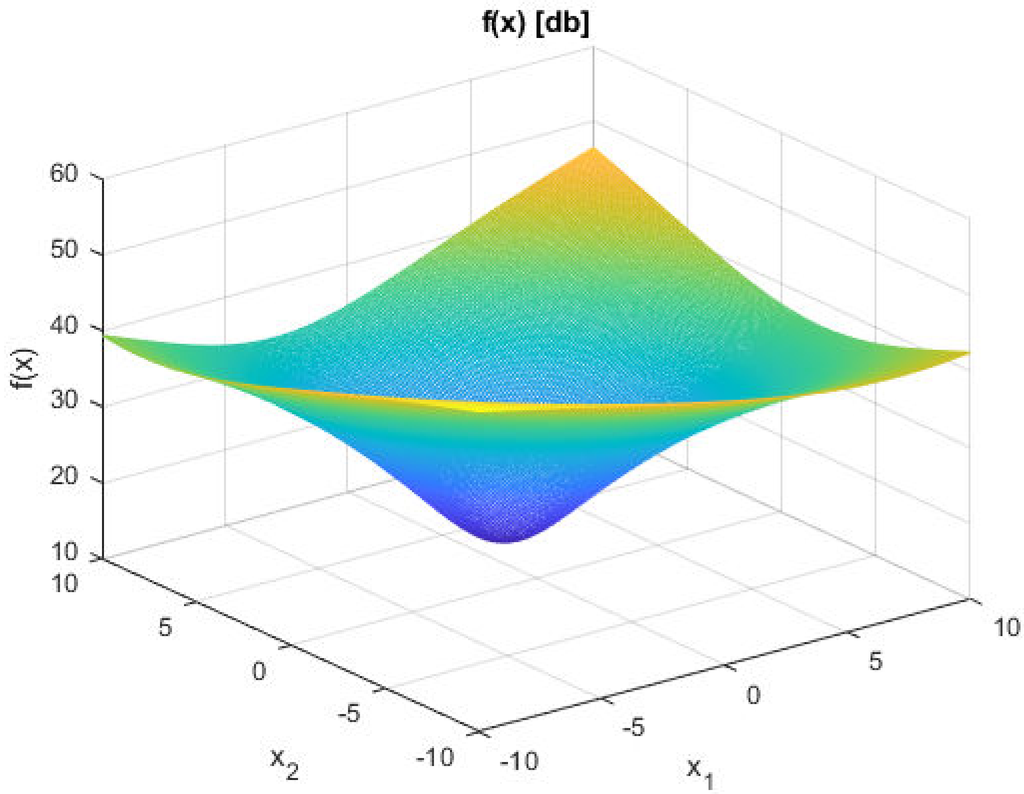
Maryam Riazi

810197518

Table of Contents

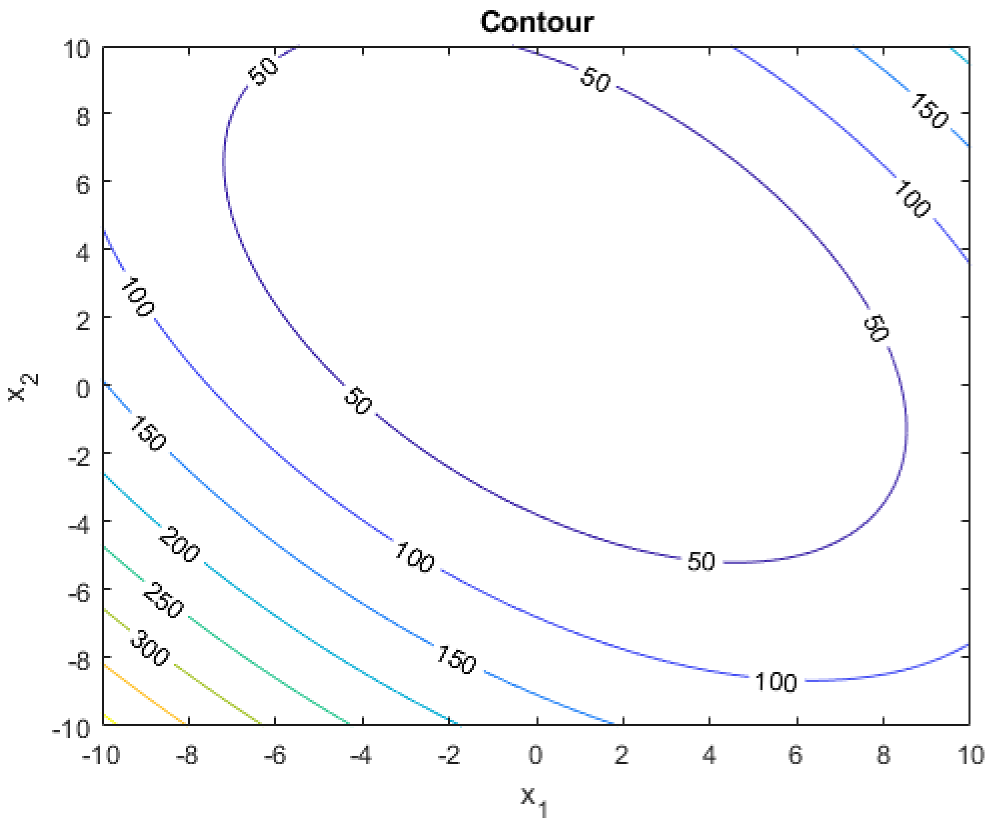
Question1.

```
[x1,x2] = meshgrid(-10:0.1:10);
f = x1.*x1 + x2.*x2- 4*x1 -6*x2 + 13 + x1.*x2;
fdb =  20*log10(f);
figure
mesh(x1,x2,fdb)
title('f(x) [db]')
xlabel('x_1')
ylabel('x_2')
zlabel('f(x)')
```



Question2.

```
figure
contour(x1,x2,f,'ShowText','on')
title('Contours of f(x)')
xlabel('x_1')
ylabel('x_2')
```


Contour
50
50
150
100
100
50
50
150
50
200
100
250
300
150
100
x2
x1
10
8
6
4
2
0
-2
-4
-6
-8
-10
-10
-8
-6
-4
-2
0
2
4
6
8
10

Question3.

$$f(x_1, x_2) = x_1^2 + x_2^2 - 4x_1 - 6x_2 + 13 + x_1 x_2$$

$$\nabla f = \begin{pmatrix} 2x_1 - 4 + x_2 \\ 2x_2 - 6 + x_1 \end{pmatrix}$$

$$H = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$$

$$\underset{1\times2}{x^T} \; \underset{2\times2}{H} \; \underset{2\times1}{x} = \underset{1\times2}{(x_1 \;\; x_2)} \underset{2\times2}{\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

$$= \begin{pmatrix} 2x_1 + x_2 & x_1 + 2x_2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

$$= 2x_1^2 + x_1 x_2 + x_1 x_2 + 2x_2^2$$

$$= \underbrace{(x_1^2 + x_2^2)}_{>0} + \underbrace{(x_1 + x_2)^2}_{>0} > 0$$

Question4.

$$\nabla f = \emptyset \Longrightarrow \nabla f = \begin{pmatrix} 2x_1 - 4 + x_2 \\ 2x_2 - 6 + x_1 \end{pmatrix} = \emptyset$$

$$\begin{cases} 2x_1 + x_2 = 4 \\ 2x_2 + x_1 = 6 \end{cases} \Longrightarrow 3x_1 = 2 \Longrightarrow \begin{cases} x_1 = 2/3 \\ x_2 = 4 - \frac{4}{3} \\ \quad = \frac{8}{3} \end{cases}$$

Question5.

Steepest decent Algorithm is as below (my "SD" function act exactly as below):

$$\begin{bmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \end{bmatrix} = \begin{bmatrix} x_1^{(k)} \\ x_2^{(k)} \end{bmatrix} - \mu \nabla_{x^{(k)}} f$$

```
X = [6;6];
mio1 = 0.01;
mio2 = 0.1;
[OptimalX1,fvalues1,iterations1]=SD(X,mio1)
```

```
OptimalX1 = 2×1
    0.6667
    2.6666
fvalues1 = 1×951
   57.6508   54.4988   51.5322   48.7403   46.1126   43.6395   41.3119   39.1211 ···
iterations1 = 951
```

```
[OptimalX2,fvalues2,iterations2]=SD(X,mio2)
```

```
OptimalX2 = 2×1
    0.6667
    2.6666
fvalues2 = 1×91
   32.0800   17.8484   10.8257    7.3446    5.6066    4.7288    4.2775    4.0392 ···
iterations2 = 91
```

A)

Yes! In both cases the function converges but in different paces.

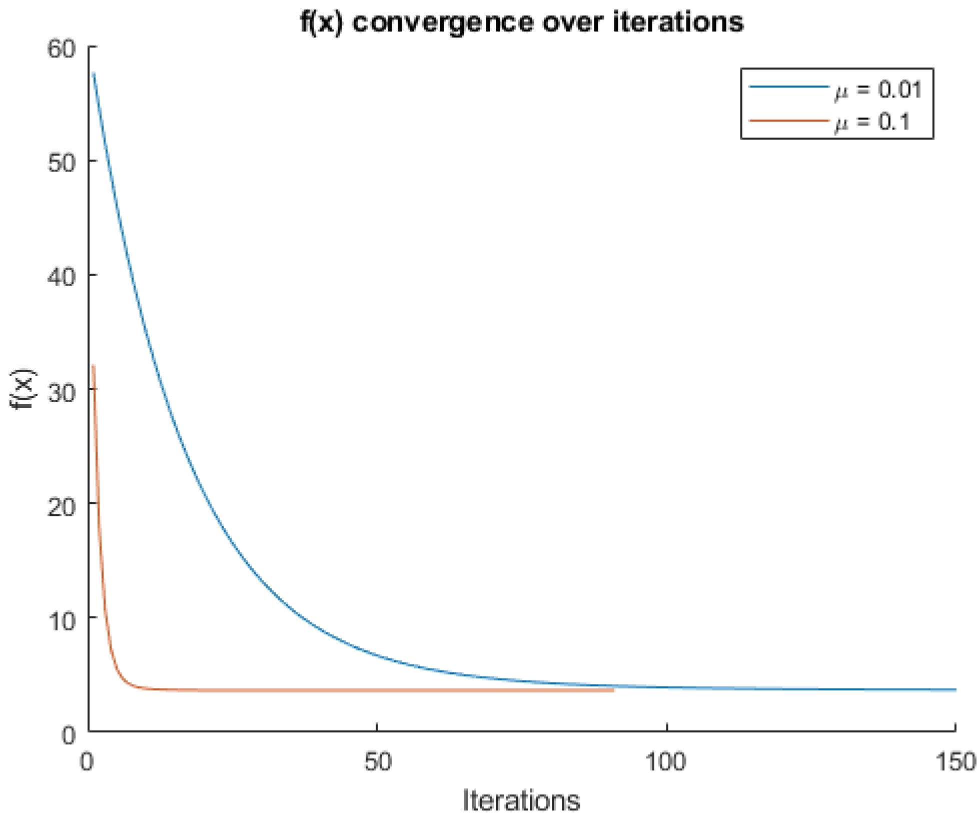
B)

As it is shown in above results for $\mu = 0.01$, 951 iterations are needed but for $\mu = 0.1$, only 91 iterartions are needed.

C)

```
figure
hold on
plot (1:150,fvalues1(1:150));
```

```
plot (1:iterations2,fvalues2);
title('f(x) convergence over iterations')
xlabel('Iterations')
ylabel('f(x)')
legend('\mu = 0.01','\mu = 0.1','Location','northeast')
hold off
```



Question6.

A)

Newton Algorithm is as below (my "Newton" function is exactly as below):

$$\begin{bmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \end{bmatrix} = \begin{bmatrix} x_1^{(k)} \\ x_2^{(k)} \end{bmatrix} - H_{x^{(k)}}^{-1} \nabla_{x^{(k)}} f$$

```
[OptimalNewtonX,fvaluesNewton,iterationsNewton] = Newton(X)
```

```
OptimalNewtonX = 2×1
    0.6667
    2.6667
fvaluesNewton = 3.6667
iterationsNewton = 1
```

B)

At each iteration, Newton's method attempts to fit a paraboloid on f(x) at $x^{(k)}$, find the minimum of the paraboloid and then find the projection of that point on f(x). This process will be done exactly over and over again.

In our case, f is a quadratic function, so the exact extremum is found in one step!

Question7.

In Alternation Minimization algorithm all we do is exactly as in S.D algorithm with a little difference!

The difference is as that in this algorithm we do gradient descent one time assuming x1 is fixed and the other time x2 is fixed.

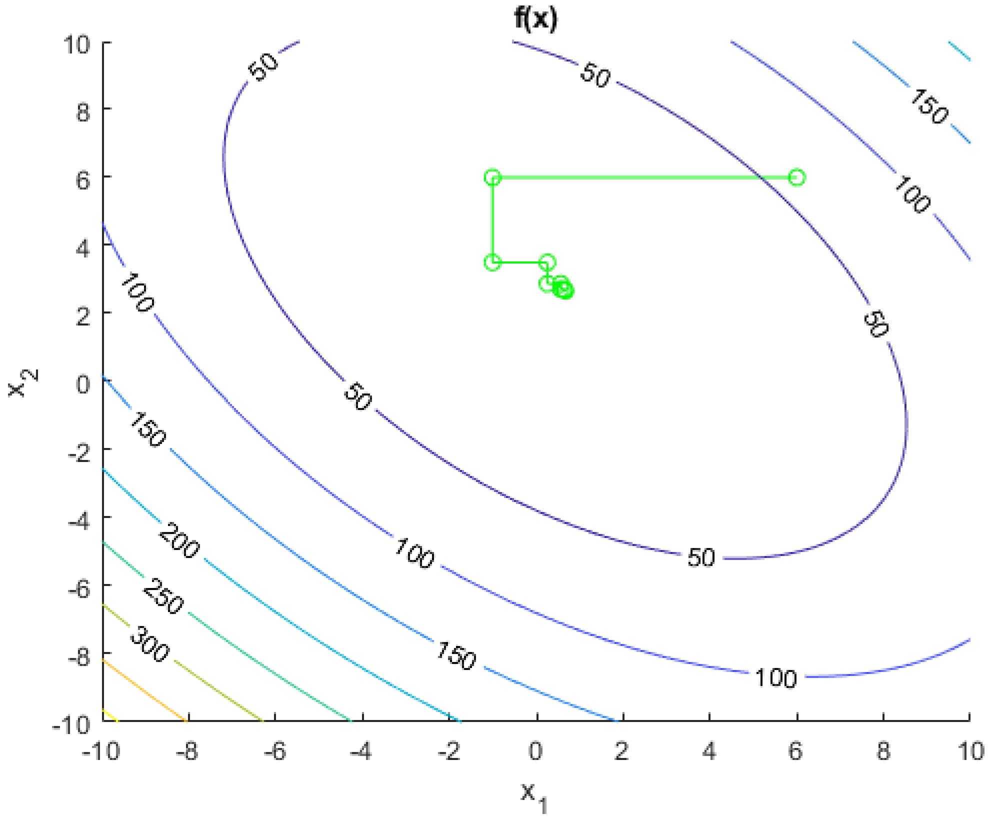
This process should be iterated as much as the function converges!

$x_1^{(1)} = x_2^{(1)} = 6$

(2) $x_2^{(2)}: \nabla f(x_1) = 2x_1 - 4 + x_2 = 0 \Rightarrow x_1^{(2)} = \dfrac{4 - x_2}{2} = -1$

(3) $x_1^{(3)} = -1: \nabla f(x_2) = 2x_2 - 6 + x_1 = 0 \Rightarrow x_2^{(3)} = \dfrac{6 - x_1}{2} = \dfrac{7}{2}$

```
[OptimalAMX_1,OptimalAMX_2]=AM(X);
figure
hold on
plot(OptimalAMX_1,OptimalAMX_2,'-o','Color','green')
contour(x1,x2,f,'ShowText','on')
xlim([-10 10])
ylim([-10 10])
title('f(x)')
xlabel('x_1')
ylabel('x_2')
hold off
```



Question8.

$f(x) = x_1^2 + x_2^2 - 4x_1 - 6x_2 + 13 + x_1 x_2$

s.t $\begin{cases} -10 < x_1, x_2 < 10 \\ \\ x_1^2 + x_2^2 = 1 \end{cases}$

For any point $P = (x_1, x_2)$ in 2D plane, the projection of it on the unit circle would be:

$\theta = \mathrm{tg}^{-1}\left(\frac{x_2}{x_1}\right)$

$P' = (\cos(\theta), \sin(\theta))$

```
X = [6;6];
mio = 0.1;
[OptimalX,min_fvalue] = Gradient_projection(X,mio)
```

```
X = 2×1
    0.4943
    0.8693
min_fvalue = 7.2367
```

Question9.

$$f(x_1,x_2) = x_1^2 + x_2^2 - 4x_1 - 6x_2 + 13 + x_1x_2$$

$$\begin{cases} -10 < x_1, x_2 < 10 \\ x_1^2 + x_2^2 = 1 \end{cases}$$

$$\mathcal{L} = x_1^2 + x_2^2 - 4x_1 - 6x_2 + 13 + x_1x_2 + \lambda\left(x_1^2 + x_2^2 - 1\right)$$

$$\Rightarrow \begin{cases} \dfrac{\partial \mathcal{L}}{\partial x_1} = 2x_1 - 4 + x_2 + 2\lambda x_1 = \phi \quad (1) \\ \dfrac{\partial \mathcal{L}}{\partial x_2} = 2x_2 - 6 + x_1 + 2\lambda x_2 = \phi \quad (2) \\ x_1^2 + x_2^2 = 1 \end{cases}$$

$$\Longrightarrow \begin{cases} x_1(2+2\lambda) + x_2 - 4 = 0 \\ x_1 + x_2(2+2\lambda) - 6 = 0 \end{cases}$$

$$\Longrightarrow x_1 + (2+2\lambda)\left(4 - x_1(2+2\lambda)\right) - 6 = 0$$

$$\Longrightarrow x_1\left(1 - (2+2\lambda)^2\right) = 6 - 4(2+2\lambda)$$

$$\longrightarrow x_1 = \frac{6 - 4(2+2\lambda)}{1 - (2+2\lambda)^2}$$

$$\longrightarrow x_2 = \frac{4 - 6(2+2\lambda)}{1 - (2+2\lambda)^2}$$

$$x_1^2 + x_2^2 = 1 = \frac{40 + (2+2\lambda)^2(40) - 2 \times 48}{\left(1 - (2+2\lambda)^2\right)^2}$$

$$\Longrightarrow \text{verified } \lambda \simeq 2.08$$

$$\Longrightarrow \begin{cases} x_1 \simeq 0.5039 \\ x_2 \simeq 0.8637 \\ \text{optimal value of } f \simeq 7.23 \end{cases}$$

As we could see the Lagrangian method proved the Gradient Projection method.

Functions

```
function [OptimalX,fvalues,iterations] = SD(X,mio)
fvalues = [];
gradf = [2*X(1)-4+X(2);2*X(2)-6+X(1)];
norm(gradf);
iterations = 0;
while (norm(gradf)>0.0001)
    X = X - mio*gradf;
    iterations = iterations + 1;
    fvalues(iterations) = X(1)^2+X(2)^2-4*X(1)-6*X(2)+13+X(1)*X(2);
    gradf = [2*X(1)-4+X(2);2*X(2)-6+X(1)];
end
OptimalX = X;
end
%%%%%%%%%%%
function [OptimalX,fvalues,iterations] = Newton(X)
fvalues = [];
Hessianf = [2 1
    1 2];
gradf = [2*X(1)-4+X(2);2*X(2)-6+X(1)];
iterations = 0;
while (norm(gradf)>0.0001)
    X = X - inv(Hessianf)*gradf;
    iterations = iterations + 1;
    fvalues(iterations) = X(1)^2+X(2)^2-4*X(1)-6*X(2)+13+X(1)*X(2);
    gradf = [2*X(1)-4+X(2);2*X(2)-6+X(1)];
end
OptimalX = X;
end
%%%%%%%%%%%%
function [Xvalues_1,Xvalues_2] = AM(X)
Xvalues_1=[];
Xvalues_2=[];
iteration = 1;
while true
    Xvalues_1 = [Xvalues_1 X(1)];
    Xvalues_2 = [Xvalues_2 X(2)];
    if (length(Xvalues_1) >= 100)
        break;
    end
    if mod(iteration,2) == 1
        X(1) = (4-X(2))/2;
    else
        X(2) = (6-X(1))/2;
    end
    iteration = iteration + 1;
end
end
%%%%%%%%%%%%
function [OptimalX,fvalues] = Gradient_projection(X,mio)
iterations = 0;
```

```
while iterations<100
    fvalues = X(1)^2+X(2)^2-4*X(1)-6*X(2)+13+X(1)*X(2);
    gradf = [2*X(1)-4+X(2)
        2*X(2)-6+X(1)];

    X(1) = X(1)-mio*gradf(1);
    X(2) = X(2)-mio*gradf(2);

    theta = atan(X(2)/X(1));
    X(1) = cos(theta); %mitooni inja biyay va
    X(2) = sin(theta);
    iterations = iterations + 1;
end
OptimalX = X;
end
```