ECE 580 Fun Work #2

Vineeth Ravi

February 17, 2018

PU ID : 0030019456
EMAIL ID : ravi24@purdue.edu

Answer 1

The MATLAB Code is shown after the Answer.

$f(x) = x^T Q x$

$$Q = \begin{bmatrix} 2 & 1 \\ 0 & 3 \end{bmatrix}$$

We bracket the minimizer starting with the initial guess of $x^{(0)} = [0.8, -0.25]^T$.
Taking $\epsilon = 0.075$, $d = \nabla f(x^{(0)} = [-2.95, 0.70]^T$ and implementing the MATLAB Code below we get :-

Iteration Number	$X^{(i)}$	$f(X^{(i)})$
$Start = 0$	$[0.8, -0.25]^T$	1.2675
1	$[0.5788, -0.1975]^T$	0.6726
2	$[0.1363, -0.0925]^T$	0.0502
3	$[-0.7487, 0.1175]^T$	1.0747

Since $f(X^{(3)}) \geqslant f(X^{(2)})$ and $f(X^{(2)}) \leqslant f(X^{(1)})$, i.e. The function increases between $f(X^{(2)})$ and $f(X^{(3)})$

$\implies$ The Minimizer is located in the interval $[X^{(1)}, X^{(3)}]$

The minimizer is located in the interval $a = [0.578750, -0.197500]^T$ and $b = [-0.748750, 0.117500]^T$

MATLAB CODE

```
%% FunWork Question 1
% To find the Uncertainity region
X=zeros(2,100);
f=zeros(1,100);

Q=[2 1;0 3];         % Q Matrix
x0=[0.8;-0.25];
epsilon=0.075;

X(:,1)=x0;
d=-(Q+Q')*X(:,1);
f(1)=X(:,1)'*Q*X(:,1);

for i=2:100
X(:,i)=X(:,(i-1))+(epsilon*d);
f(i)=X(:,i)'*Q*X(:,i);
epsilon=2*epsilon;
if((f(i-1)<f(i))&&(f(i-2)>f(i-1))&&(i>2))
    break
end
```

```
end

a=X(:,i-2);
b=X(:,i);

fprintf('The minimizer is located in the interval [a=[%f;%f] and b=[%f;%f]] \n',a(1),a(2),b(1)
```

Answer 2

The MATLAB Code is shown after the Answer.

Golden Section Search

We first determine the number of necessary iterations to reduce the width of the uncertainty interval from **1.3644** to **0.2** in MATLAB.

The initial or starting points and values to begin the iterations in MATLAB are :-
Initial Values
The initial width of the uncertainty interval is 1.364361
The Number of necessary iterations required is 4

Initial interval $[a = [0.578750, 0.197500]^T$ and $b = [-0.748750; 0.117500]^T]$
$a_0 = [0.578750, -0.197500]^T$
$b_0 = [-0.748750, 0.117500]^T$
$f(a_0) = 0.672619$
$f(b_0) = 1.074694$

Initial Value of $s = [0.071690, -0.077181]^T$
Initial Value of $t = [-0.241690, -0.002819]^T$
Initial Value of f(s)=0.022616
Initial Value of f(t)=0.117533

Showing all the MATLAB iterations in Tabular form:-

Iteration	a_k	b_k	$f(a_k)$	$f(b_k)$	New Uncertainty Interval	Width
1	$\begin{bmatrix} 0.265370 \\ -0.123139 \end{bmatrix}$	$\begin{bmatrix} 0.265370 \\ -0.123139 \end{bmatrix}$	0.153654	0.022616	$\left[a = \begin{bmatrix} 0.578750 \\ -0.197500 \end{bmatrix}, b = \begin{bmatrix} -0.241690 \\ -0.002819 \end{bmatrix}\right]$	0.843222
2	$\begin{bmatrix} 0.071690 \\ -0.077181 \end{bmatrix}$	$\begin{bmatrix} -0.048010 \\ -0.048777 \end{bmatrix}$	0.022616	0.014089	$\left[a = \begin{bmatrix} 0.265370 \\ -0.123139 \end{bmatrix}, b = \begin{bmatrix} -0.241690 \\ -0.002819 \end{bmatrix}\right]$	0.521140
3	$\begin{bmatrix} -0.048010 \\ -0.048777 \end{bmatrix}$	$\begin{bmatrix} -0.121990 \\ -0.031223 \end{bmatrix}$	0.014089	0.036496	$\left[a = \begin{bmatrix} 0.071690 \\ -0.077181 \end{bmatrix}, b = \begin{bmatrix} -0.241690 \\ -0.002819 \end{bmatrix}\right]$	0.322082
4	$\begin{bmatrix} -0.002289 \\ -0.059626 \end{bmatrix}$	$\begin{bmatrix} -0.048010 \\ -0.048777 \end{bmatrix}$	0.010813	0.014089	$\left[a = \begin{bmatrix} 0.071690 \\ -0.077181 \end{bmatrix}, b = \begin{bmatrix} -0.121990 \\ -0.031223 \end{bmatrix}\right]$	0.199058

The final width of the uncertainty interval is **0.199058**.

Hence the minimizer is located within the final uncertainty interval $\left[a = \begin{bmatrix} 0.071690 \\ -0.077181 \end{bmatrix}, b = \begin{bmatrix} -0.121990 \\ -0.031223 \end{bmatrix}\right]$

Hence the Width of the uncertainty interval is reduced to a value less than 0.2.

MATLAB CODE

```
%% FunWork Question 2
% Golden Section Search
% Run the MATLAB CODE from FunWork Question 1 before running this code, to find 'a' and 'b'
E=0.2;
Q=[2 1;0 3];        % Q Matrix
rho=(3-sqrt(5))/2;  % Golden Ratio
D=norm(a-b);
```

```
fprintf('The initial width of the uncertainty interval is %f \n',D);
N=ceil((log(E/D))/(log(1-rho)));
fprintf('The Number of necessary iterations required is %d \n\n',N);

fa=a'*Q*a;
fb=b'*Q*b;

s = a + rho*(b-a);
t = a + (1-rho)*(b-a);
ft=t'*Q*t;
fs=s'*Q*s;

fprintf('Initial Values\n');
fprintf('Initial interval [a=[%f;%f] and b=[%f;%f]] \n',a(1),a(2),b(1),b(2));
fprintf('a0=[%f,%f]\n',a(1),a(2));
fprintf('b0=[%f,%f]\n',b(1),b(2));
fprintf('f(a0)=[%f]\n',fa);
fprintf('f(b0)=[%f]\n',fb);
fprintf('Initial Value of s=[%f,%f]\n',s(1),s(2));
fprintf('Initial Value of t=[%f,%f]\n',t(1),t(2));
fprintf('Initial Value of f(s)=[%f]\n',fs);
fprintf('Initial Value of f(t)=[%f]\n',ft);
fprintf('\n');

for i=1:N
    if(fs<ft)
        b=t;
        t=s;
        s=a+rho*(b-a);
        ft=fs;
        fs=s'*Q*s;
    else
        a=s;
        s=t;
        t=a+(1-rho)*(b-a);
        fs=ft;
        ft=t'*Q*t;
    end
    fprintf('\n');
    fprintf('Iteration number i=%d\n',i);
    fprintf('a%d = [%f,%f]\n',i,s(1,1),s(2,1));
    fprintf('b%d = [%f,%f]\n',i,t(1,1),t(2,1));
    fprintf('f(a%d) = [%f]\n',i,fs);
    fprintf('f(b%d) = [%f]\n',i,ft);
    fprintf('New width interval [a=[%f;%f] and b=[%f;%f]] \n',a(1),a(2),b(1),b(2));
    fprintf('width=%f\n',norm(b-a));

end
af=a;
bf=b;
D=norm(af-bf);
fprintf('The final width of the uncertainty interval is %f \n',D);
```

Answer 3

The MATLAB Code is shown after the Answer.

3a) Fibonacci Search

For the Fibonacci Search, we first determine N which determines the Fibonacci Numbers, i.e. $F_1, F_2 ... F_{N+1}$

We can determine N by using the following formula

$\frac{(1+2\epsilon)*(InitialWidth)}{(FinalExpectedWidth)} \leqslant F_{N+1}$.

Taking the Value of $\epsilon = 0.05$ and Implementing the MATLAB Code below:-

We determine the number of necessary iterations to reduce the width of the uncertainty interval from **1.3644** to **0.2** in MATLAB.

The initial or starting points and values to begin the iterations in MATLAB are :-
Initial Values
The initial width of the uncertainty interval is 1.364361
$F_{N+1} \geqslant 7.5042$
$\implies F_{N+1} = 8$
$\implies$ The Number of necessary iterations required is $N = 4$

Initial interval $[a = [0.578750, 0.197500]^T$ and $b = [-0.748750; 0.117500]^T]$
$a_0 = [0.578750, -0.197500]^T$
$b_0 = [-0.748750, 0.117500]^T$
$f(a_0) = 0.672619$
$f(b_0) = 1.074694$

Initial Value of $s = [0.080938, -0.079375]^T$
Initial Value of $t = [-0.250938, -0.000625]^T$
Initial Value of f(s)=0.025579
Initial Value of f(t)=0.126097

Showing all the MATLAB iterations in Tabular form:-

Iteration	ρ_k	a_k	b_k	$f(a_k)$	$f(b_k)$	New Uncertainty Interval	Width
1	0.3750	$\begin{bmatrix} 0.246875 \\ -0.118750 \end{bmatrix}$	$\begin{bmatrix} 0.080938 \\ -0.079375 \end{bmatrix}$	0.134883	0.025579	$\left[a = \begin{bmatrix} 0.578750 \\ -0.197500 \end{bmatrix}, b = \begin{bmatrix} -0.250938 \\ -0.000625 \end{bmatrix}\right]$	0.852726
2	0.4000	$\begin{bmatrix} 0.080938 \\ -0.079375 \end{bmatrix}$	$\begin{bmatrix} -0.085000 \\ -0.040000 \end{bmatrix}$	0.025579	0.022650	$\left[a = \begin{bmatrix} 0.246875 \\ -0.118750 \end{bmatrix}, b = \begin{bmatrix} -0.250938 \\ -0.000625 \end{bmatrix}\right]$	0.511635
3	0.3333	$\begin{bmatrix} -0.085000 \\ -0.040000 \end{bmatrix}$	$\begin{bmatrix} -0.101594 \\ -0.036063 \end{bmatrix}$	0.022650	0.028208	$\left[a = \begin{bmatrix} 0.080938 \\ -0.079375 \end{bmatrix}, b = \begin{bmatrix} -0.250938 \\ -0.000625 \end{bmatrix}\right]$	0.341090
4	0.4500	$\begin{bmatrix} 0.080938 \\ -0.079375 \end{bmatrix}$	$\begin{bmatrix} -0.085000 \\ -0.040000 \end{bmatrix}$	0.025579	0.022650	$\left[a = \begin{bmatrix} 0.080938 \\ -0.079375 \end{bmatrix}, b = \begin{bmatrix} -0.101594 \\ -0.036063 \end{bmatrix}\right]$	0.187600

The final width of the uncertainty interval is **0.187600**.

Hence the minimizer is located within the final uncertainty interval $\left[a = \begin{bmatrix} 0.080938 \\ -0.079375 \end{bmatrix}, b = \begin{bmatrix} -0.101594 \\ -0.036063 \end{bmatrix}\right]$

Hence the Width of the uncertainty interval is reduced to a value less than 0.2.

MATLAB CODE

```
%% FunWork Question 3
% Fibonacci Search
% Run the MATLAB CODE from Question 1 before running this code, to find 'a' and 'b'
E=0.2;
epsilon=0.05;
Q=[2 1;0 3];        % Q Matrix

rho=zeros(1,100); % Ratio values
F=zeros(1,100); % Fibonacci Sequence.
F(1)=0;
F(2)=1;

D=norm(b-a);
fprintf('The initial width of the uncertainty interval is %f \n',D);
F_N1=(1+2*epsilon)*(D/E);

for i=3:100
```

```
    F(i)=F(i-1)+F(i-2);
    if(F(i)>=F_N1)
        break;
    end
end

N=i-3;

for i=1:N
    rho(i)=1-(F(N+3-i)/F(N+4-i));
end
rho(N)=rho(N)-epsilon;


fa=a'*Q*a;
fb=b'*Q*b;

s = a + rho(1)*(b-a);
t = a + (1-rho(1))*(b-a);
ft=t'*Q*t;
fs=s'*Q*s;

fprintf('Initial Values\n');
fprintf('Initial interval [a=[%f;%f] and b=[%f;%f]] \n',a(1),a(2),b(1),b(2));
fprintf('a0=[%f,%f]\n',a(1),a(2));
fprintf('b0=[%f,%f]\n',b(1),b(2));
fprintf('f(a0)=[%f]\n',fa);
fprintf('f(b0)=[%f]\n',fb);
fprintf('Initial Value of s=[%f,%f]\n',s(1),s(2));
fprintf('Initial Value of t=[%f,%f]\n',t(1),t(2));
fprintf('Initial Value of f(s)=[%f]\n',fs);
fprintf('Initial Value of f(t)=[%f]\n',ft);
fprintf('\n');

for i=1:N
    if(fs<ft)
        b=t;
        t=s;
        s=a+rho(i+1)*(b-a);
        ft=fs;
        fs=s'*Q*s;
    else
        a=s;
        s=t;
        t=a+(1-rho(i+1))*(b-a);
        fs=ft;
        ft=t'*Q*t;
    end
    fprintf('\n');
    fprintf('Iteration number i=%d\n',i);
    fprintf('a%d = [%f,%f]\n',i,s(1,1),s(2,1));
    fprintf('b%d = [%f,%f]\n',i,t(1,1),t(2,1));
    fprintf('f(a%d) = [%f]\n',i,fs);
    fprintf('f(b%d) = [%f]\n',i,ft);
    fprintf('New width interval [a=[%f;%f] and b=[%f;%f]] \n',a(1),a(2),b(1),b(2));
    fprintf('width=%f\n',norm(b-a));

end

af=a;
bf=b;
D=norm(af-bf);
fprintf('The final width of the uncertainty interval is %f \n',D);
```

The MATLAB Code is shown after the Answer.

3b) The Newton Method

Implementing Newton's method to find the Minimizer in the surface obtained by the intersection of the Function $f(x) = x^TQx$ and the line $x^{(0)} - \alpha\nabla f(x^{(0)})$.

Using Newton's iterative expression we find α at every iteration.
We use the MATLAB Code shown below to find α.
Since, this is only a Quadratic function in alpha, we obtain α in 1 step.
$\alpha^{(1)} = \alpha^{(0)} - \frac{f'(\alpha^{(0)})}{f''(\alpha^{(0)})}$
$\phi(\alpha) = f(x(\alpha)) = 3*((7\alpha)/10 - 1/4)^2 + 2*((59\alpha)/20 - 4/5)^2 - ((7\alpha)/10 - 1/4)*((59\alpha)/20 - 4/5)$
$f(\alpha) = 16.81\alpha^2 - 9.1925\alpha + 1.2675$

The value of $f'(\alpha^{(0)})$ is $\mathbf{-9.1925}$.
The value of $f"(\alpha^{(0)})$ is **33.6200**.
The value of $\alpha^{(1)}$ is **0.2734**.

$$\text{The Minimizer is } x^* = \begin{bmatrix} -0.0066 \\ -0.0586 \end{bmatrix}$$

The value of the function at the Minimizer x^* is **0.0108**.

MATLAB CODE

```
syms x y

%f = 3*(1-x).^2.*exp(-(x.^2) - (y+1).^2) - 10*(x/5 - x.^3 - y.^5).*exp(-x.^2-y.^2) - 1/3*exp(-

f=2*(x.^2)+(x.*y)+(3*y.^2);

%x0=[1;0];
x0=[0.8;-0.25];
syms alpha

grad_f=gradient(f);
l=subs(grad_f,{x,y},{x0(1),x0(2)});

xalpha=x0-alpha*l;

falpha=subs(f,{x,y},{xalpha(1),xalpha(2)});

al=[];

al(1)=0;

for i=2:10000

    df=diff(falpha,alpha);
    ddf=diff(df,alpha);
    dfalpha=subs(df,{alpha},{al(i-1)});
    ddfalpha=subs(ddf,{alpha},{al(i-1)});
    al(i)=al(i-1)-(dfalpha/ddfalpha);

    dfalphas=subs(df,{alpha},{al(i)});

   if(dfalphas <0.0000001)
       N=i;
       break
   end

end

xf=x0-al(N).*l;
disp('The value of f'' alpha is ');
```

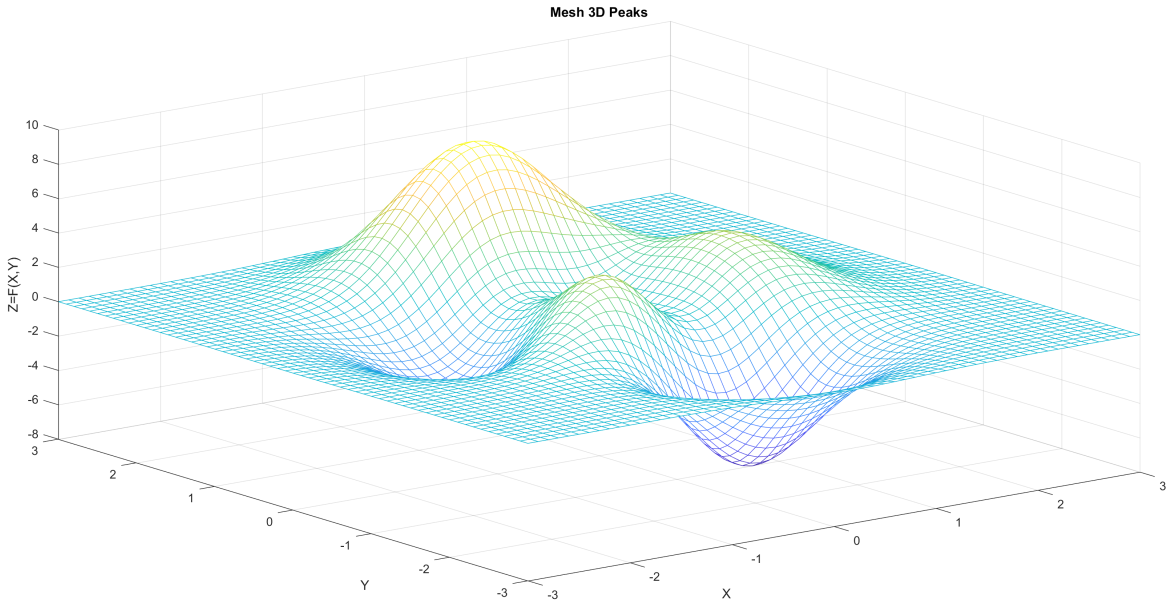
```
disp(double(dfalpha));
disp('The value of f" alpha is');
disp(double(ddfalpha));
disp('The value of alpha is');
disp(al(2));
disp('The Minimizer is');
disp(double(xf));

disp('The Value of the Function at the Minimizer is');
disp(double(subs(f,{x,y},{xf(1),xf(2)})));
```

Answer 4

The MATLAB Code is shown after the Answer.
The MATLAB commands in the code are use to plot the images, shown in the figures below.

a) 3D Plot using mesh

Figure 1: MESH 3D Plot.

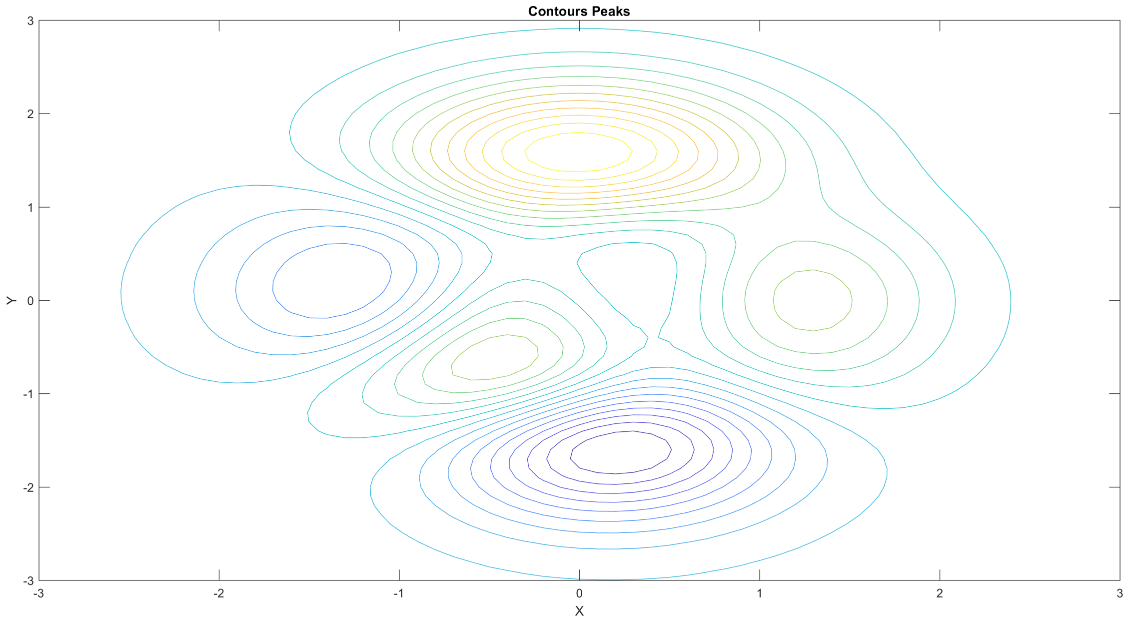


b) Contours Plot

MATLAB CODE

```
X=-3:0.1:3;
Y=X;
[x,y]=meshgrid(X);

z = 3*(1-x).^2.*exp(-(x.^2) - (y+1).^2) - 10*(x/5 - x.^3 - y.^5).*exp(-x.^2-y.^2) - 1/3*exp(-(
% Plot Surface for verification
%{
surf(x,y,z);

[X,Y,Z] = peaks;
mesh(X,Y,Z);
figure
contour(X,Y,Z,20);
%}

mesh(x,y,z);
```

Figure 2: Contours Plot.




```
xlabel('X');
ylabel('Y');
zlabel('Z=F(X,Y)');
title('Mesh 3D Peaks');

figure;

contour(x,y,z,20);
xlabel('X');
ylabel('Y');
title('Contours Peaks');
```

Answer 5

Gradient Descent

The MATLAB Code is shown after the Answer.

We take $\alpha = 0.02$ for implementing the Gradient Descent Algorithm. At every iteration starting from $x^{(0)}$ we move in the direction of the $-\nabla f(x^{(i)}$ at every iteration i, till we reach the minimizer x^*, based on a condition such that $||\nabla f(x^{(i)}|| \leqslant \epsilon$, where $\epsilon = 10^{-4}$.
We also use the function *quiver* to plot the arrows to indicate the progression of the optimization process, as shown in the Figures.
When we run the MATLAB code for:-

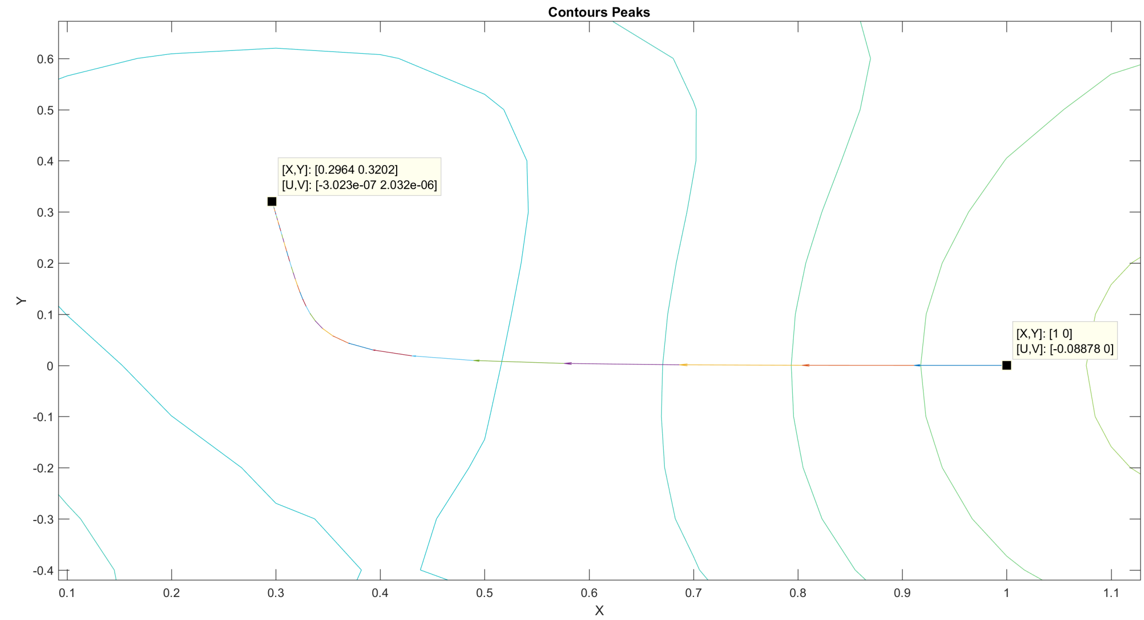
a) $x^{(0)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

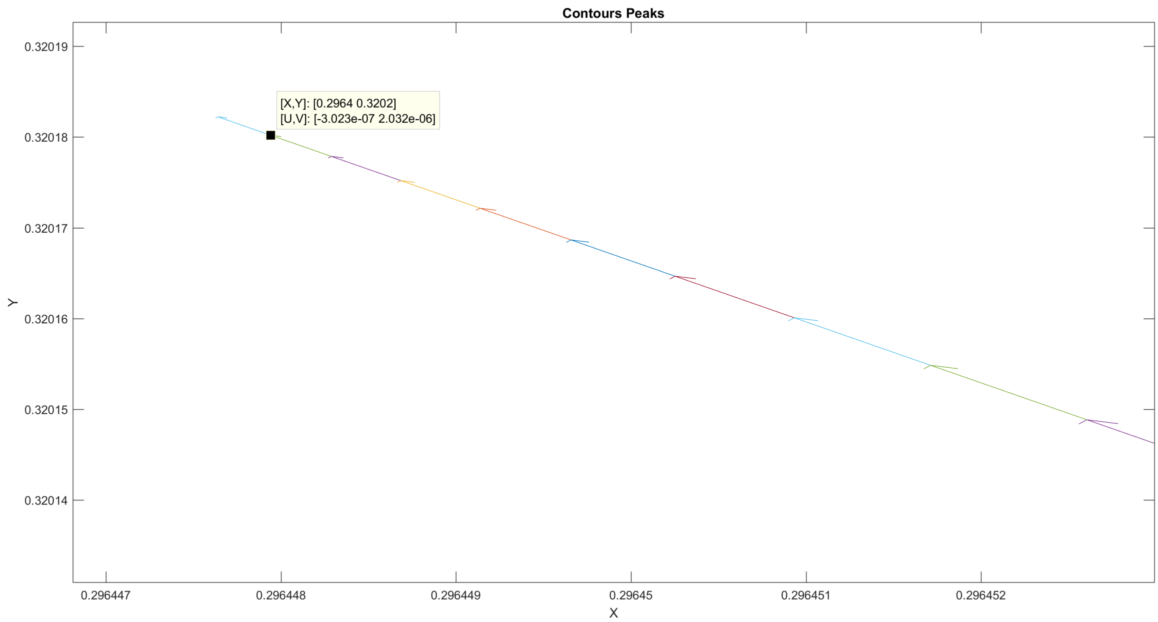
We get :-

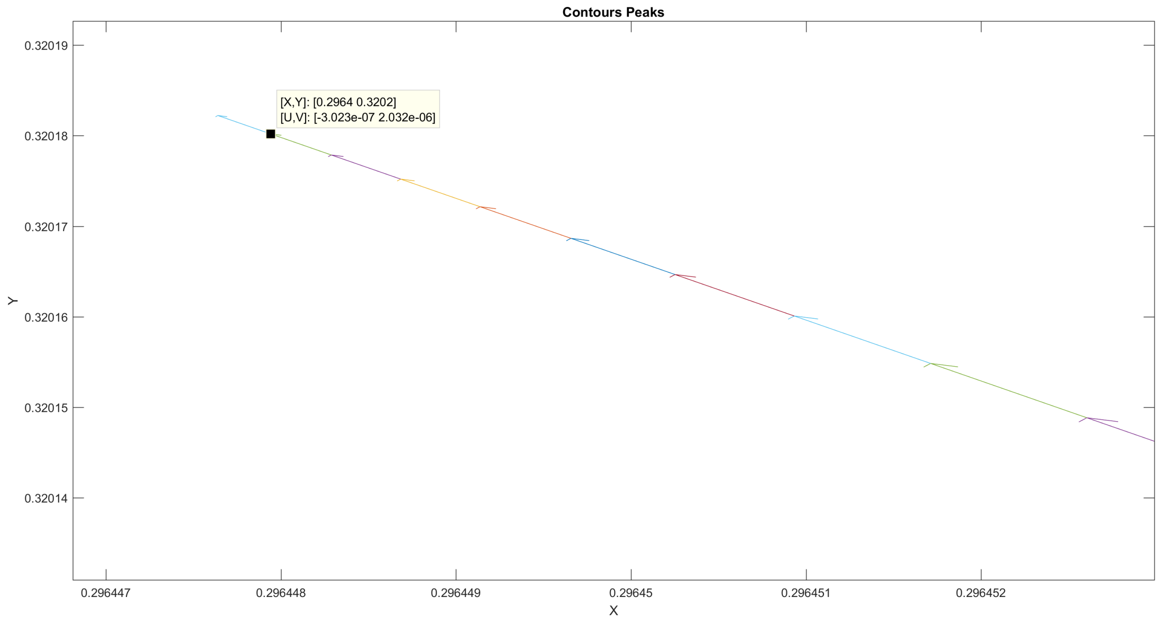
The number of iterations needed is 91.

$$\text{The local minimum is found at } X^* = \begin{bmatrix} 0.2964 \\ 0.3202 \end{bmatrix}$$

The value of the function at this minimum is -0.0649.


Contours Peaks
[X,Y]: [0.2964 0.3202]
[U,V]: [-3.023e-07 2.032e-06]
[X,Y]: [1 0]
[U,V]: [-0.08878 0]
Y
X


Contours Peaks
[X,Y]: [0.2964 0.3202]
[U,V]: [-3.023e-07 2.032e-06]
Y
X



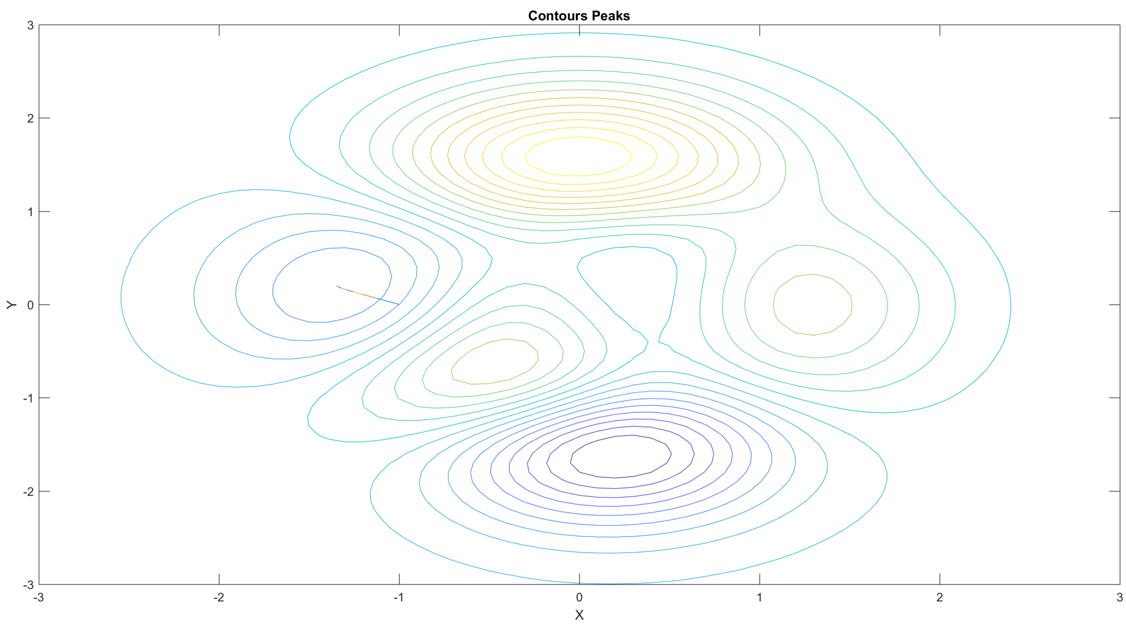
b) $x^{(0)} = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$

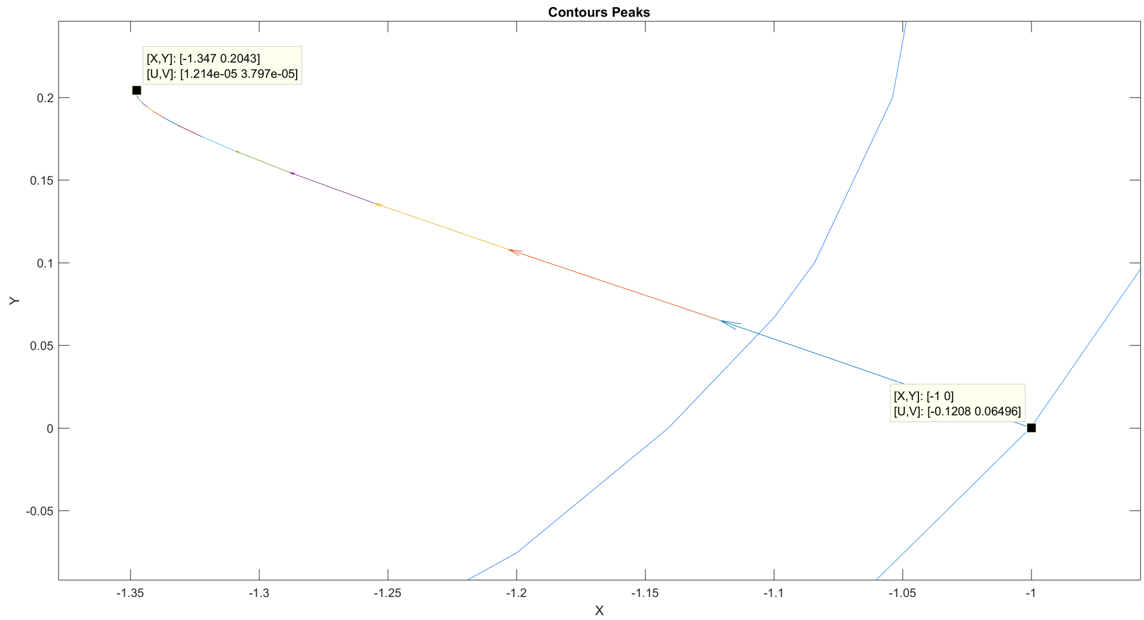
We get :-

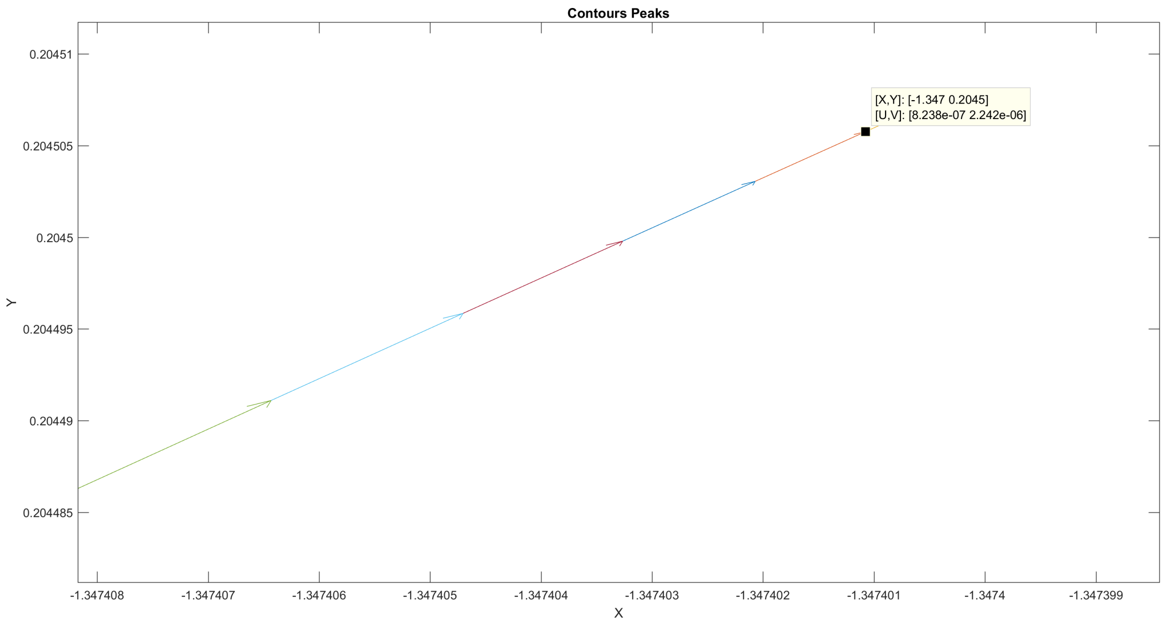
$$\text{The number of iterations needed is } 46.$$

$$\text{The local minimum is found at } X^* = \begin{bmatrix} -1.3474 \\ 0.2045 \end{bmatrix}$$

$$\text{The value of the function at this minimum is } -3.0498.$$







MATLAB CODE

```
%% Gradient Descent First Point
syms x y

f = 3*(1-x).^2.*exp(-(x.^2) - (y+1).^2) - 10*(x/5 - x.^3 - y.^5).*exp(-x.^2-y.^2) - 1/3*exp(-(

alpha = 0.02;
x0=[1;0];

X=[];
i=2;

grad_f=gradient(f);
X(:,1)=x0;

l=subs(grad_f,{x,y},{x0(1),x0(2)});

while norm(l)>=0.0001
  X(:,i)=X(:,i-1)-(alpha)*l;
```

```
  l=subs(grad_f,{x,y},{X(1,i),X(2,i)});
  i=i+1;
end
N=i-1;
disp('The points obtained by gradient descent method are:');
disp(X);
disp(' The number of iterations needed is ');
disp(N);
disp('The local minimum is found at X* = ');
disp(X(:,N));
disp('The value of the function at this minimum is ');
disp(double(subs(f,{x,y},{X(1,N),X(2,N)})));

%% Plot of the arrows
% Before Running this CODE :
% Run Any 1 of Gradient Descent or the Steepest Descent algorithm codes,
% for computing X(all points) and N(total points), to plot the points on contour with arrows
a=-3:0.1:3;
b=a;
[x,y]=meshgrid(a);

z = 3*(1-x).^2.*exp(-(x.^2) - (y+1).^2) - 10*(x/5 - x.^3 - y.^5).*exp(-x.^2-y.^2) - 1/3*exp(-(

figure;

contour(x,y,z,20);
xlabel('X');
ylabel('Y');
title('Contours Peaks');

hold on;

for i=1:N-1

p1 = X(:,i);                        % First Point
p2 = X(:,i+1);                        % Second Point
dp = p2-p1;                        % Difference
quiver(p1(1),p1(2),dp(1),dp(2),0);
%text(p1(1),p1(2), sprintf('(%f,%f)',p1));
%text(p2(1),p2(2), sprintf('(%f,%f)',p2));

end

%% Gradient Descent Second Point
syms x y

f = 3*(1-x).^2.*exp(-(x.^2) - (y+1).^2) - 10*(x/5 - x.^3 - y.^5).*exp(-x.^2-y.^2) - 1/3*exp(-(

alpha = 0.02;
x0=[-1;0];

X=[];
i=2;

grad_f=gradient(f);
X(:,1)=x0;

l=subs(grad_f,{x,y},{x0(1),x0(2)});

while norm(l)>=0.0001
  X(:,i)=X(:,i-1)-(alpha)*l;
  l=subs(grad_f,{x,y},{X(1,i),X(2,i)});
  i=i+1;
end
```

```
N=i -1;
disp('The points obtained by gradient descent method are:');
disp(X);
disp(' The number of iterations needed is ');
disp(N);
disp('The local minimum is found at X* = ');
disp(X(:,N));
disp('The value of the function at this minimum is ');
disp(double(subs(f,{x,y},{X(1,N),X(2,N)})));

%% Plot of the arrows
% Before Running this CODE :
% Run Any 1 of Gradient Descent or the Steepest Descent algorithm codes,
% for computing X(all points) and N(total points), to plot the points on contour with arrows
a=-3:0.1:3;
b=a;
[x,y]=meshgrid(a);

z = 3*(1-x).^2.*exp(-(x.^2) - (y+1).^2) - 10*(x/5 - x.^3 - y.^5).*exp(-x.^2-y.^2) - 1/3*exp(-(

figure;

contour(x,y,z,20);
xlabel('X');
ylabel('Y');
title('Contours Peaks');

hold on;

for i=1:N-1

p1 = X(:,i);                        % First Point
p2 = X(:,i+1);                        % Second Point
dp = p2-p1;                       % Difference
quiver(p1(1),p1(2),dp(1),dp(2),0);
%text(p1(1),p1(2), sprintf('(%f,%f)',p1));
%text(p2(1),p2(2), sprintf('(%f,%f)',p2));

end
```

Steepest Descent

The MATLAB Code is shown after the Answer.
I have used two different algorithms for computing α at every step. The first method is Newton's algorithm and the second method is using 'GOLDEN Section'. Both the functions give rise to the same answer and are shown in the end after the answer.
Newton Method

We use the *computealpha* function for finding α every iteration of the gradient descent algorithm. In this function, if the value of α at any iteration is $\leqslant 0$, then we replace it with $-\alpha$. This works because, at that iteration, we move in the opposite direction of the local maximizer, So eventually, we will move in the direction of the local minimizer. This happens, since we go in the direction opposite to the maximizer at every point/iteration when $\alpha \leqslant 0$ and When $\alpha \geqslant 0$, then we do not need to modify it, since we are already heading in the direction of the minimizer.

We find $\alpha^{(i)}$, such that its $f'(\alpha^{(i)}) \leqslant \epsilon$, where $\epsilon = 10^{-7}$.
We also use the function *quiver* to plot the arrows to indicate the progression of the optimization process, as shown in the Figures.

GOLDEN Section
In this algorithm we use a bracketing algorithm to bracket the function $f(x(\alpha))$, and find a range for α.
Then we use the Golden section search rule, to reduce the width of the interval to less than 0.01. We then compute the midpoint and use this value of alpha for the current iteration. We repeat this process for every iteration $x^{(i)}$, till we reach the minimizer based on the same condition of the Gradient Section Search mentioned previously.

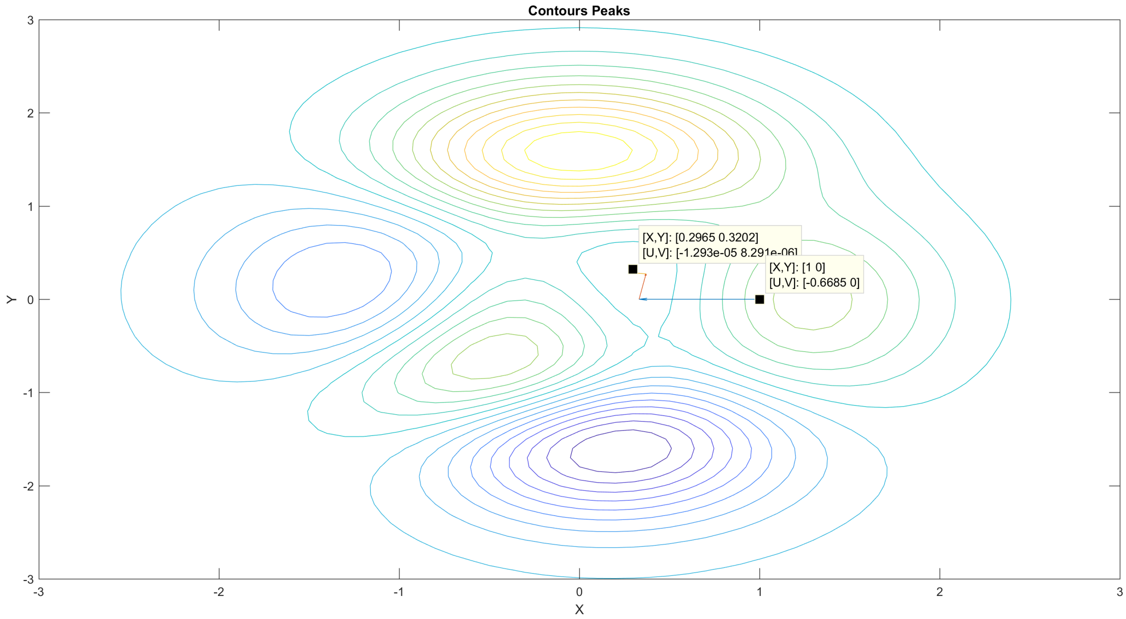
When we run the MATLAB code for:-

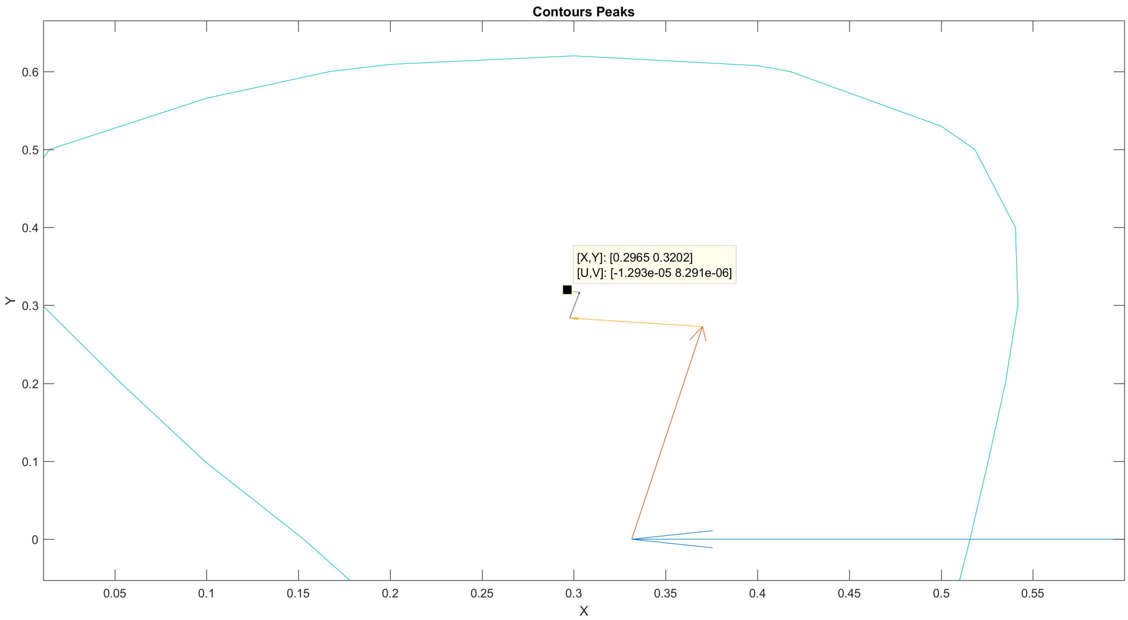
a) $x^{(0)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

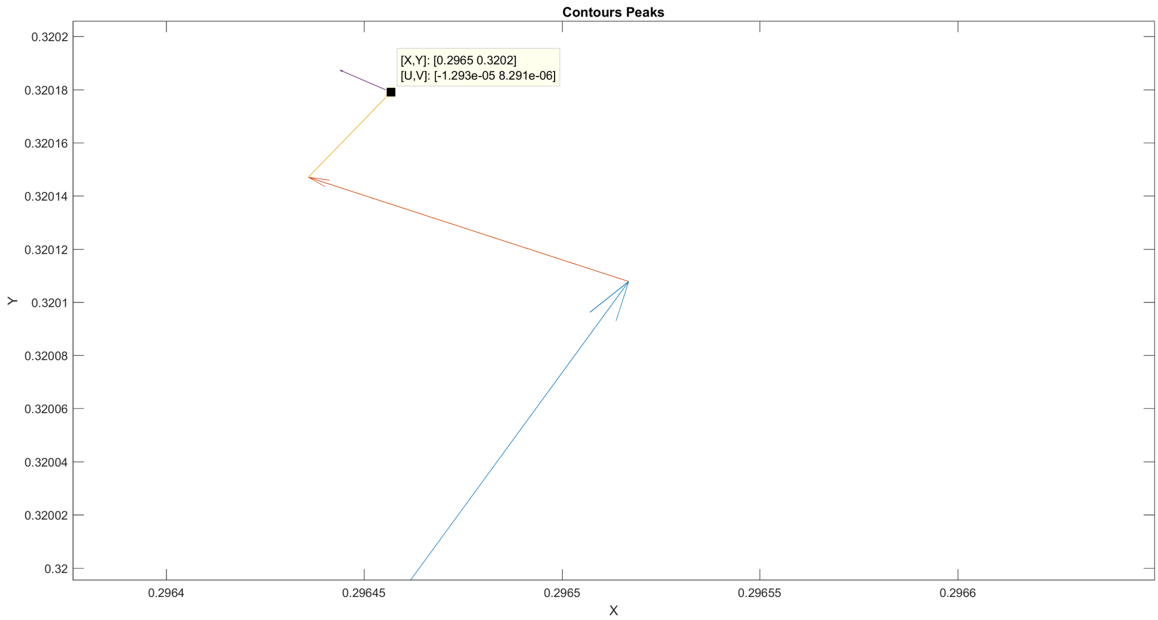
The number of iterations needed is 12.

$$\text{The local minimum is found at } X^* = \begin{bmatrix} 0.2964 \\ 0.3202 \end{bmatrix}$$

The value of the function at this minimum is -0.0649.





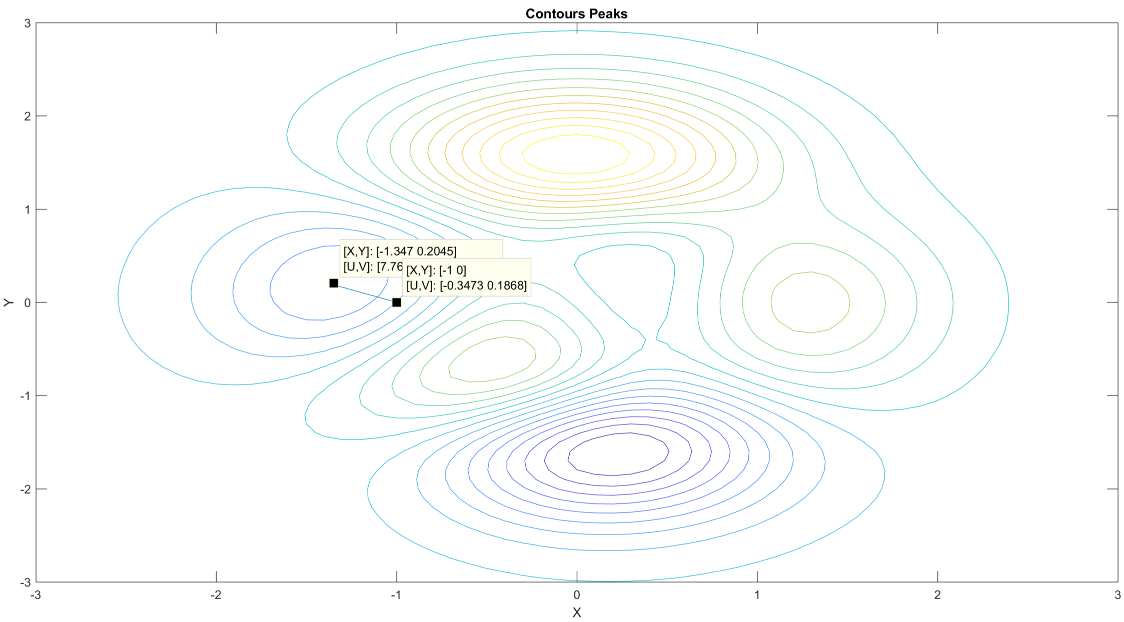


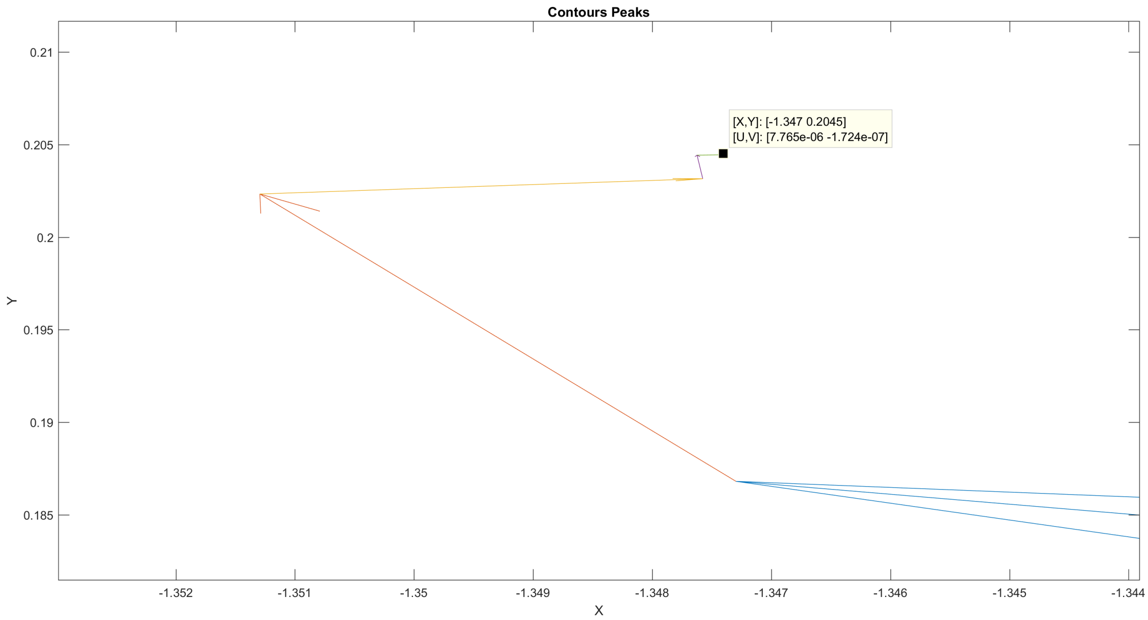
b) $x^{(0)} = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$

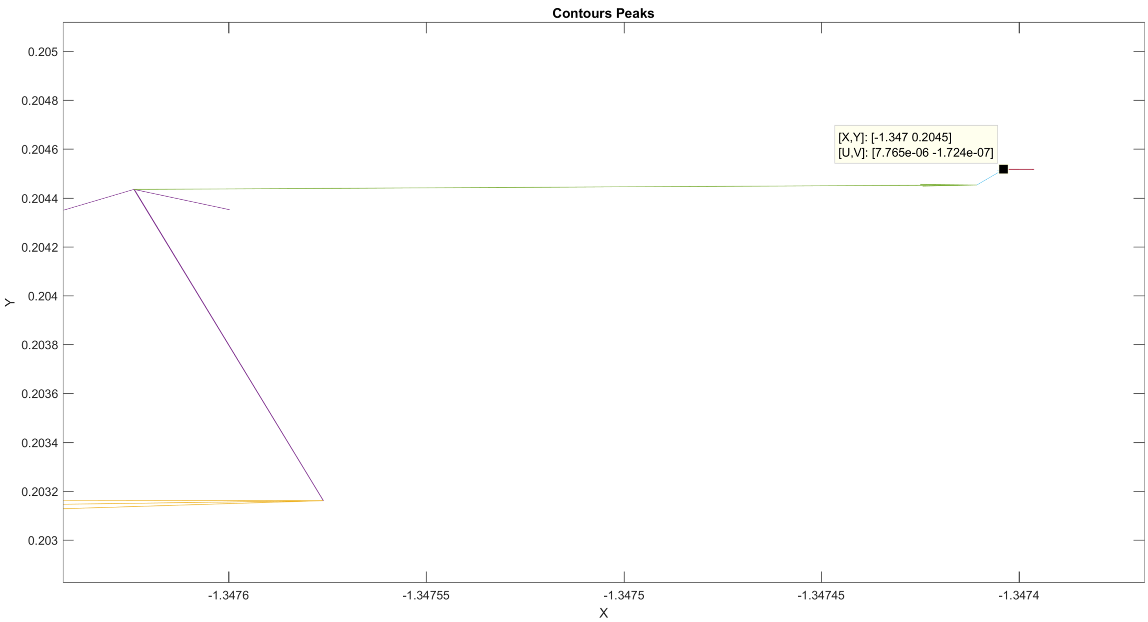
The number of iterations needed is 8.

The local minimum is found at $X^* = \begin{bmatrix} -1.3474 \\ 0.2045 \end{bmatrix}$

The value of the function at this minimum is -3.0498.







MATLAB CODE

```
%% Steepest Descent First Point

syms x y

f = 3*(1-x).^2.*exp(-(x.^2) - (y+1).^2) - 10*(x/5 - x.^3 - y.^5).*exp(-x.^2-y.^2) - 1/3*exp(-(

alphas = [];
x0=[1;0];

X=[];
i=2;

grad_f=gradient(f);
X(:,1)=x0;

l=subs(grad_f,{x,y},{x0(1),x0(2)});

while norm(l)>=0.0001
```

```
    alphas=AlphaTRY(X(:,i-1));
    disp(alphas);
   X(:,i)=X(:,i-1)-(alphas)*l;
    l=subs(grad_f,{x,y},{X(1,i),X(2,i)});
    i=i+1;
 end
N=i-1;

 disp('The points obtained by steepest descent method are:');
 disp(X);
 disp(' The number of iterations needed is ');
 disp(N);
 disp('The local minimum is found at X* = ');
 disp(X(:,N));
 disp('The value of the function at this minimum is ');
 disp(double(subs(f,{x,y},{X(1,N),X(2,N)})));

% THE BELOW FUNCTION Computes alpha using GOLDEN SECTION search method.

% Function for Computing ALPHA
 function [result]=AlphaTRY(H)

 syms c d al

 z = 3*(1-c).^2.*exp(-(c.^2) - (d+1).^2) - 10*(c/5 - c.^3 - d.^5).*exp(-c.^2-d.^2) - 1/3*exp(-(

 Y0=H;

 grad_F=gradient(z);

m=subs(grad_F,{c,d},{Y0(1),Y0(2)});

 Yalp=Y0-al*m;

 fYalp=subs(z,{c,d},{Yalp(1),Yalp(2)});

% Bracketing

W=zeros(1,1000);
 fval=zeros(1,1000);
 delta=0.00001;
W(1)=0;

 dal=-gradient(fYalp,[al]);
 dalval=subs(dal,{al},{W(1)});
 fval(1)=subs(fYalp,{al},{W(1)});

 for i=2:1000
W(i)=W(i-1)+(delta*dalval);
 fval(i)=subs(fYalp,{al},{W(i)});
 delta=2*delta;
 if((i>2)&&(fval(i-1)<fval(i))&&(fval(i-2)>fval(i-1)))
      break
 end
 end

 p=W(i-2);
 q=W(i);

% Golden Search

 E = 0.02;
 rho=(3-sqrt(5))/2;  % Golden Ratio
```

```
Dist=norm(q-p);

Niter=ceil((log(E/Dist))/(log(1-rho)));

s = p + rho*(q-p);
t = p + (1-rho)*(q-p);
ft=double(subs(fYalp,{al},{t}));
fs=double(subs(fYalp,{al},{s}));

for i=1:Niter
    if(fs<ft)
        q=t;
        t=s;
        s=p+rho*(q-p);
        ft=fs;
        fs=double(subs(fYalp,{al},{s}));
    else
        p=s;
        s=t;
        t=p+(1-rho)*(q-p);
        fs=ft;
        ft=double(subs(fYalp,{al},{t}));
    end
end

pfin=p;
qfin=q;

result=(pfin+qfin)/2;

end

%% Plot of the arrows
% Before Running this CODE :
% Run Any 1 of Gradient Descent or the Steepest Descent algorithm codes,
% for computing X(all points) and N(total points), to plot the points on contour with arrows
a=-3:0.1:3;
b=a;
[x,y]=meshgrid(a);

z = 3*(1-x).^2.*exp(-(x.^2) - (y+1).^2) - 10*(x/5 - x.^3 - y.^5).*exp(-x.^2-y.^2) - 1/3*exp(-(

figure;

contour(x,y,z,20);
xlabel('X');
ylabel('Y');
title('Contours Peaks');

hold on;

for i=1:N-1

p1 = X(:,i);                              % First Point
p2 = X(:,i+1);                              % Second Point
dp = p2-p1;                               % Difference
quiver(p1(1),p1(2),dp(1),dp(2),0);
%text(p1(1),p1(2), sprintf('(%f,%f)',p1));
%text(p2(1),p2(2), sprintf('(%f,%f)',p2));

end

%% Steepest Descent Second Point
```

```
syms x y

f = 3*(1-x).^2.*exp(-(x.^2) - (y+1).^2) - 10*(x/5 - x.^3 - y.^5).*exp(-x.^2-y.^2) - 1/3*exp(-(

alphas = [];
x0=[-1;0];

X=[];
i=2;

grad_f=gradient(f);
X(:,1)=x0;

l=subs(grad_f,{x,y},{x0(1),x0(2)});

while norm(l)>=0.0001
   alphas=AlphaTRY(X(:,i-1));
   X(:,i)=X(:,i-1)-(alphas)*l;
   l=subs(grad_f,{x,y},{X(1,i),X(2,i)});
   i=i+1;
end
N=i-1;

disp('The points obtained by steepest descent method are:');
disp(X);
disp(' The number of iterations needed is ');
disp(N);
disp('The local minimum is found at X* = ');
disp(X(:,N));
disp('The value of the function at this minimum is ');
disp(double(subs(f,{x,y},{X(1,N),X(2,N)})));

% THE BELOW FUNCTION Computes alpha using GOLDEN SECTION search method.

% Function for Computing ALPHA
function [result]=AlphaTRY(H)

syms c d al

z = 3*(1-c).^2.*exp(-(c.^2) - (d+1).^2) - 10*(c/5 - c.^3 - d.^5).*exp(-c.^2-d.^2) - 1/3*exp(-(

Y0=H;

grad_F=gradient(z);

m=subs(grad_F,{c,d},{Y0(1),Y0(2)});

Yalp=Y0-al*m;

fYalp=subs(z,{c,d},{Yalp(1),Yalp(2)});

% Bracketing

W=zeros(1,1000);
fval=zeros(1,1000);
delta=0.00001;
W(1)=0;

dal=-gradient(fYalp,[al]);
dalval=subs(dal,{al},{W(1)});
fval(1)=subs(fYalp,{al},{W(1)});

for i=2:1000
W(i)=W(i-1)+(delta*dalval);
```

```
fval(i)=subs(fYalp,{al},{W(i)});
delta=2*delta;
if((i>2)&&(fval(i-1)<fval(i))&&(fval(i-2)>fval(i-1)))
    break
end
end

p=W(i-2);
q=W(i);

% Golden Search

E = 0.02;
rho=(3-sqrt(5))/2;  % Golden Ratio

Dist=norm(q-p);

Niter=ceil((log(E/Dist))/(log(1-rho)));

s = p + rho*(q-p);
t = p + (1-rho)*(q-p);
ft=double(subs(fYalp,{al},{t}));
fs=double(subs(fYalp,{al},{s}));

for i=1:Niter
    if(fs<ft)
        q=t;
        t=s;
        s=p+rho*(q-p);
        ft=fs;
        fs=double(subs(fYalp,{al},{s}));
    else
        p=s;
        s=t;
        t=p+(1-rho)*(q-p);
        fs=ft;
        ft=double(subs(fYalp,{al},{t}));
    end
end

pfin=p;
qfin=q;

result=(pfin+qfin)/2;

end


%% Plot of the arrows
% Before Running this CODE :
% Run Any 1 of Gradient Descent or the Steepest Descent algorithm codes,
% for computing X(all points) and N(total points), to plot the points on contour with arrows
a=-3:0.1:3;
b=a;
[x,y]=meshgrid(a);

z = 3*(1-x).^2.*exp(-(x.^2) - (y+1).^2) - 10*(x/5 - x.^3 - y.^5).*exp(-x.^2-y.^2) - 1/3*exp(-(

figure;

contour(x,y,z,20);
xlabel('X');
ylabel('Y');
title('Contours Peaks');
```

```
hold on;

for i=1:N-1

p1 = X(:,i);                          % First Point
p2 = X(:,i+1);                          % Second Point
dp = p2-p1;                         % Difference
quiver(p1(1),p1(2),dp(1),dp(2),0);
%text(p1(1),p1(2), sprintf('(%f,%f)',p1));
%text(p2(1),p2(2), sprintf('(%f,%f)',p2));

end



% I have also included the code for finding alpha using NEWTON's method. BOTH the methods work
% TO USE Newton's method , replace AlphaTRY() with computealpha() function.

function [result]=computealpha(H)

syms x y

f = 3*(1-x).^2.*exp(-(x.^2) - (y+1).^2) - 10*(x/5 - x.^3 - y.^5).*exp(-x.^2-y.^2) - 1/3*exp(-(

%f=2*(x.^2)+(x.*y)+(3*y.^2);

x0=H;
syms alphab

grad_f=gradient(f);
l=subs(grad_f,{x,y},{x0(1),x0(2)});

xalpha=x0-alphab*l;

falpha=subs(f,{x,y},{xalpha(1),xalpha(2)});

al=[];

al(1)=0;

for i=2:10000

    df=diff(falpha,alphab);
    ddf=diff(df,alphab);
    dfalpha=subs(df,{alphab},{al(i-1)});
    ddfalpha=subs(ddf,{alphab},{al(i-1)});
    al(i)=al(i-1)-(dfalpha/ddfalpha);

    dfalphas=subs(df,{alphab},{al(i)});

     if(al(i)<0)
        al(i)=-al(i);
    end

    %disp(al(i));

   if(dfalphas<0.0000001)
       N=i;
       break
   end

end
```

```
xf=x0-al(N).*l;

result=al(N);

end
```

THE END