ECE 580 Fun Work #3

Vineeth Ravi

March 7, 2018

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

Answer 1

The Fibonacci Line Search Function for computing Alphas, is common to all the below algorithms, So it is included in the end after all the answers.

The Main MATLAB Code is shown after the Answer.

The objective is to minimize the Rastrigin's function using the **Steepest Descent Algorithm.**

$f(x_1, x_2) = 20 + (\frac{x_1}{10})^2 + (\frac{x_2}{10})^2 - 10(cos(\frac{2\pi x_1}{10}) + cos(\frac{2\pi x_2}{10}))$

When we run the MATLAB code for:-

a) $x^{(0)} = \begin{bmatrix} 7.5 \\ 9 \end{bmatrix}$

We get :-

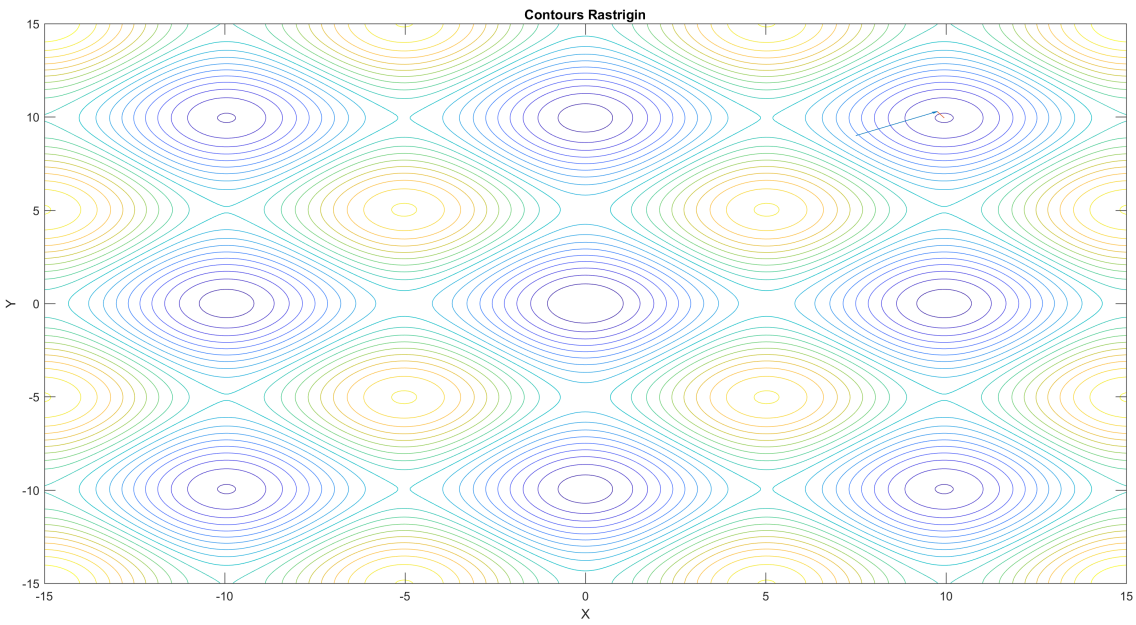
The points obtained by steepest descent method are:

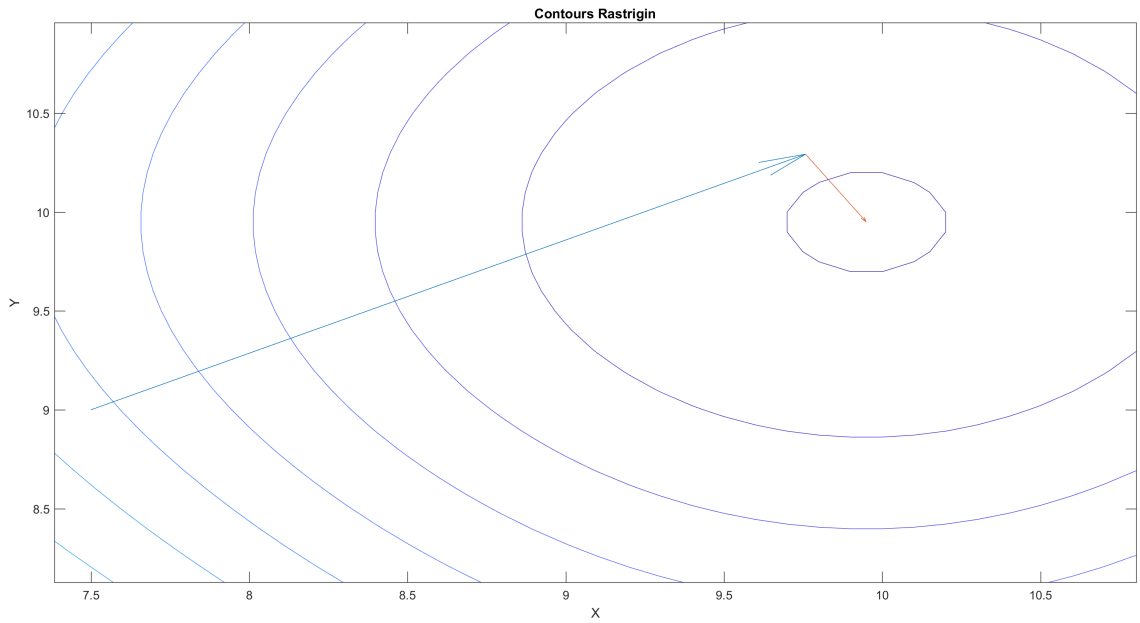
$$\begin{bmatrix} 7.5000 \\ 9.0000 \end{bmatrix} \begin{bmatrix} 9.7574 \\ 10.2931 \end{bmatrix} \begin{bmatrix} 9.9484 \\ 9.9517 \end{bmatrix} \begin{bmatrix} 9.9496 \\ 9.9496 \end{bmatrix}$$

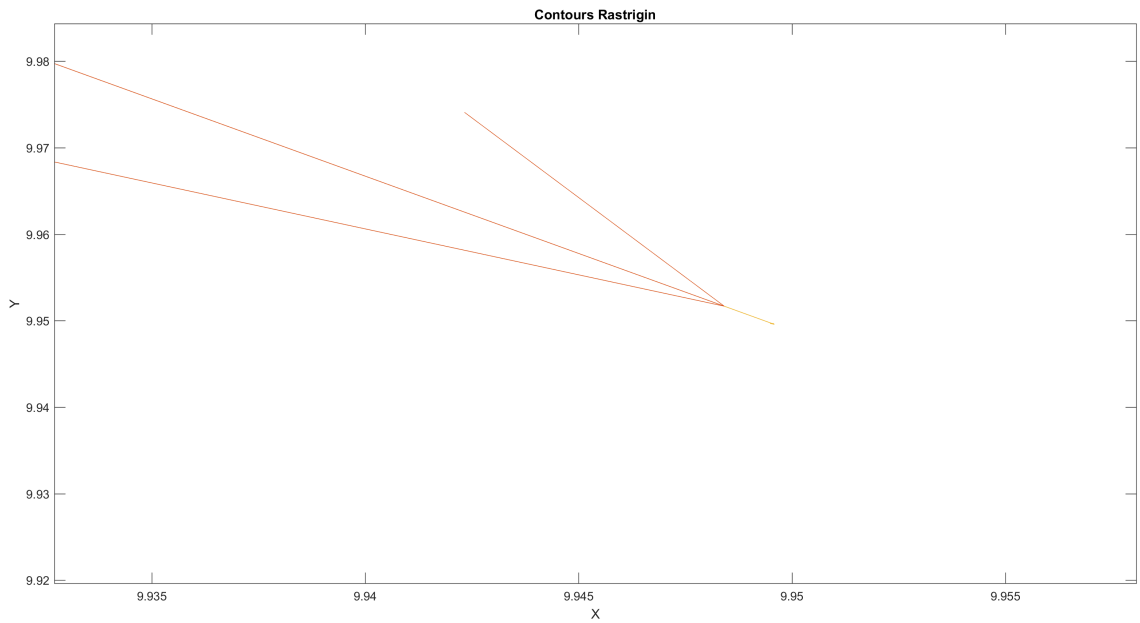
The number of iterations needed is 4.

The local minimum is found at $X^* = \begin{bmatrix} 9.9496 \\ 9.9496 \end{bmatrix}$

The value of the function at this minimum is 1.9899.







b) $x^{(0)} = \begin{bmatrix} -7 \\ -7.5 \end{bmatrix}$

We get :-

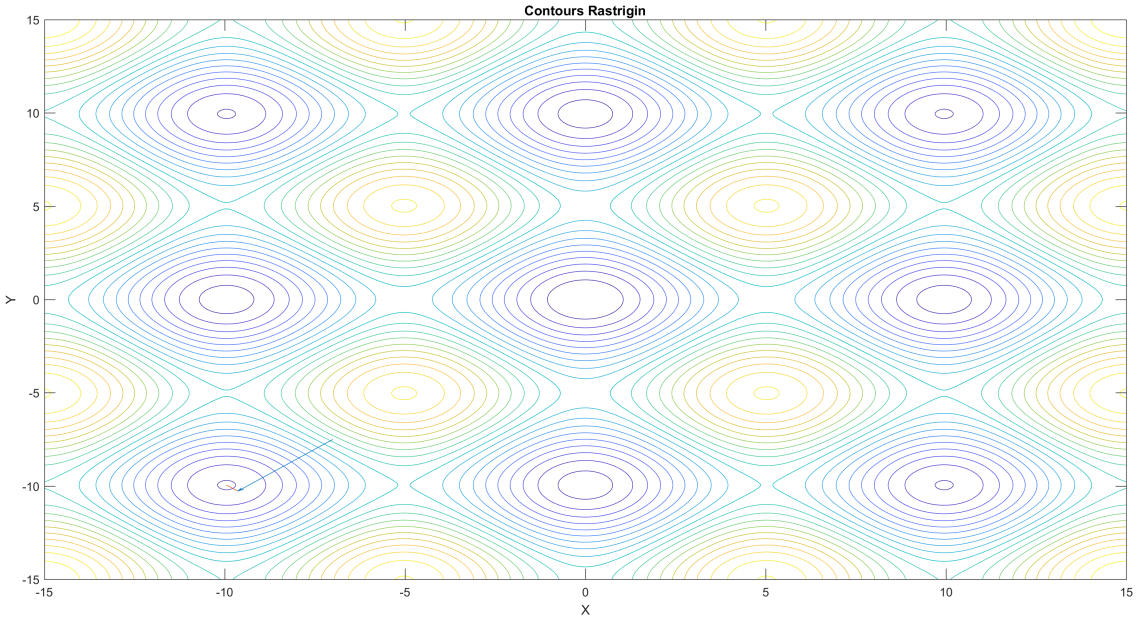
The points obtained by steepest descent method are:

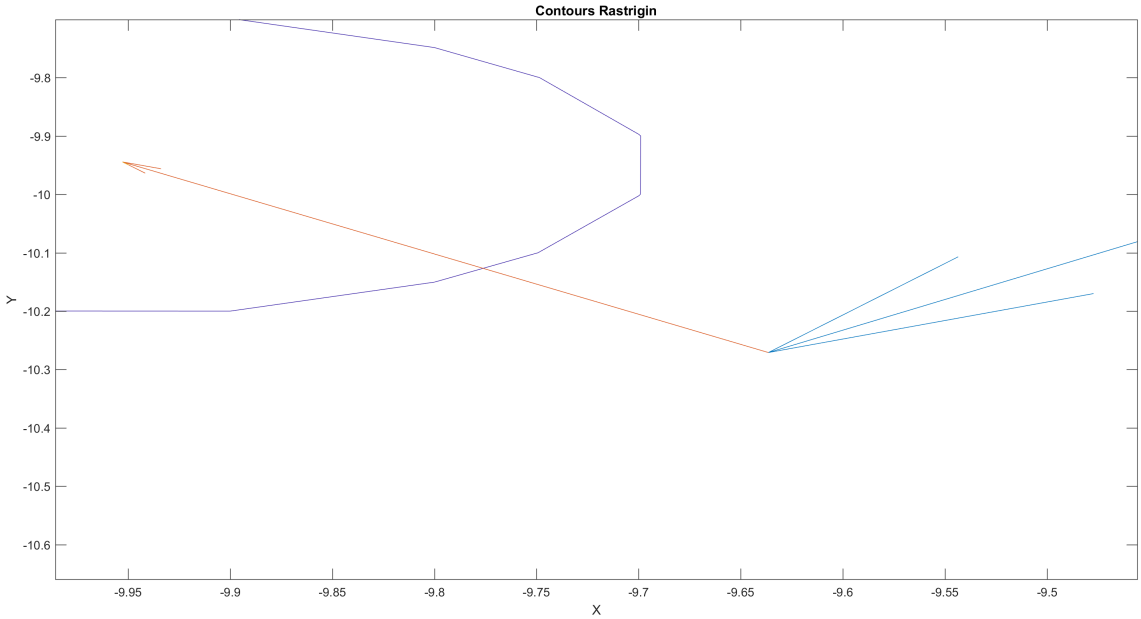
$$\begin{bmatrix} -7.0000 \\ -7.5000 \end{bmatrix} \begin{bmatrix} -9.6365 \\ -10.2709 \end{bmatrix} \begin{bmatrix} -9.9527 \\ -9.9444 \end{bmatrix} \begin{bmatrix} -9.9496 \\ -9.9496 \end{bmatrix}$$

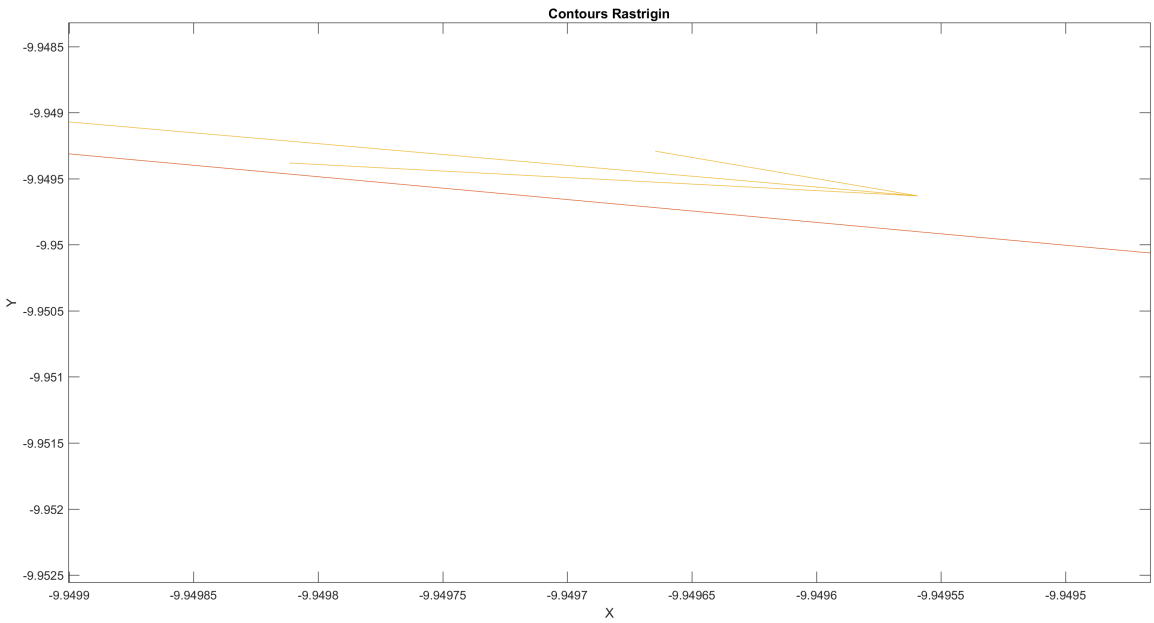
The number of iterations needed is 4.

The local minimum is found at $X^* = \begin{bmatrix} -9.9496 \\ -9.9496 \end{bmatrix}$

The value of the function at this minimum is 1.9899.


Contours Rastrigin
15
10
5
0
-5
-10
-15
Y
-15
-10
-5
0
5
10
15
X


Contours Rastrigin
-9.8
-9.9
-10
-10.1
-10.2
-10.3
-10.4
-10.5
-10.6
Y
-9.95
-9.9
-9.85
-9.8
-9.75
-9.7
-9.65
-9.6
-9.55
-9.5
X



MATLAB CODE

```
%% Steepest Descent First Point

syms x y

f= 20 + (0.01)*(x.^2) +  (0.01)*(y.^2) - 10*(cos(0.2*pi*x)+cos(0.2*pi*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=[7.5;9];

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=computealphaFibonacci(X(:,i-1));
  %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)})));


%% Steepest Descent Second Point
```

```
syms x y

f= 20 + (0.01)*(x.^2) +  (0.01)*(y.^2) - 10*(cos(0.2*pi*x)+cos(0.2*pi*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=[-7;-7.5];

X=[];
i=2;

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

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

while norm(l)>=0.001
  alphas=computealphaFibonacci(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)})));


%% Arrows Plot
a=-15:0.1:15;
b=a;
[x,y]=meshgrid(a);

z= 20 + (0.01)*(x.^2) +  (0.01)*(y.^2) - 10*(cos(0.2*pi*x)+cos(0.2*pi*y));
%z = 3*(1-x).^2.*exp(-(y.^2) - (x+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 Rastrigin');

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
```

Answer 2

The MATLAB Code is shown after the Answer.

The objective is to minimize the Rastrigin's function using the **Powell Conjugate Gradient Algorithm.**

$f(x_1, x_2) = 20 + (\frac{x_1}{10})^2 + (\frac{x_2}{10})^2 - 10(cos(\frac{2\pi x_1}{10}) + cos(\frac{2\pi x_2}{10}))$

When we run the MATLAB code for:-

a) $x^{(0)} = \begin{bmatrix} 7.5 \\ 9 \end{bmatrix}$

We get :-

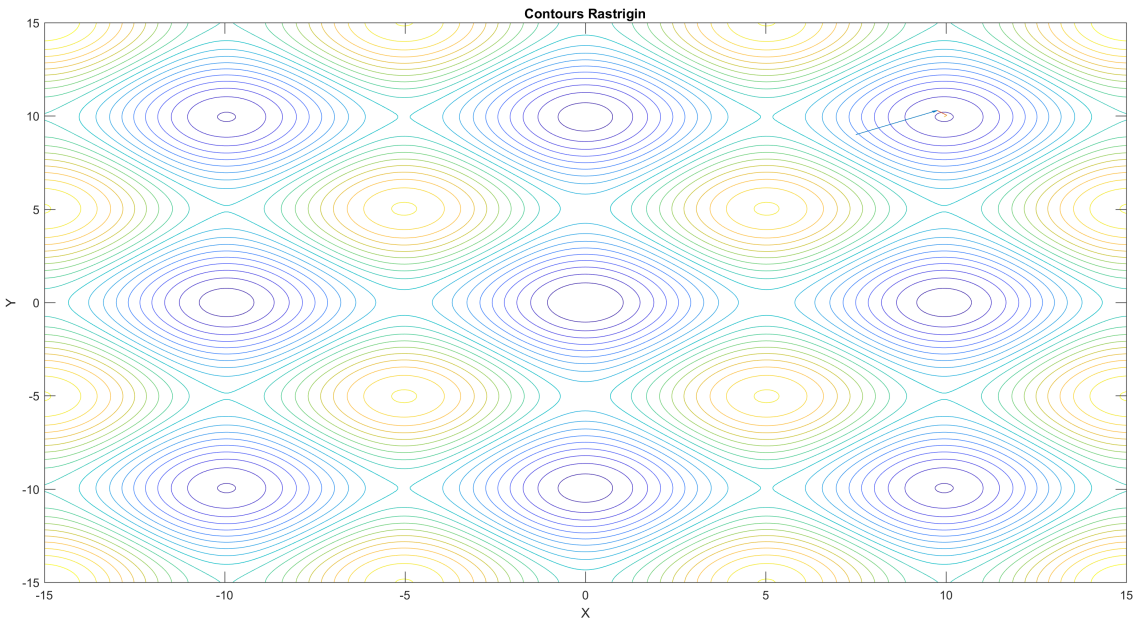
The points obtained by Powell Conjugate Gradient Algorithm are:

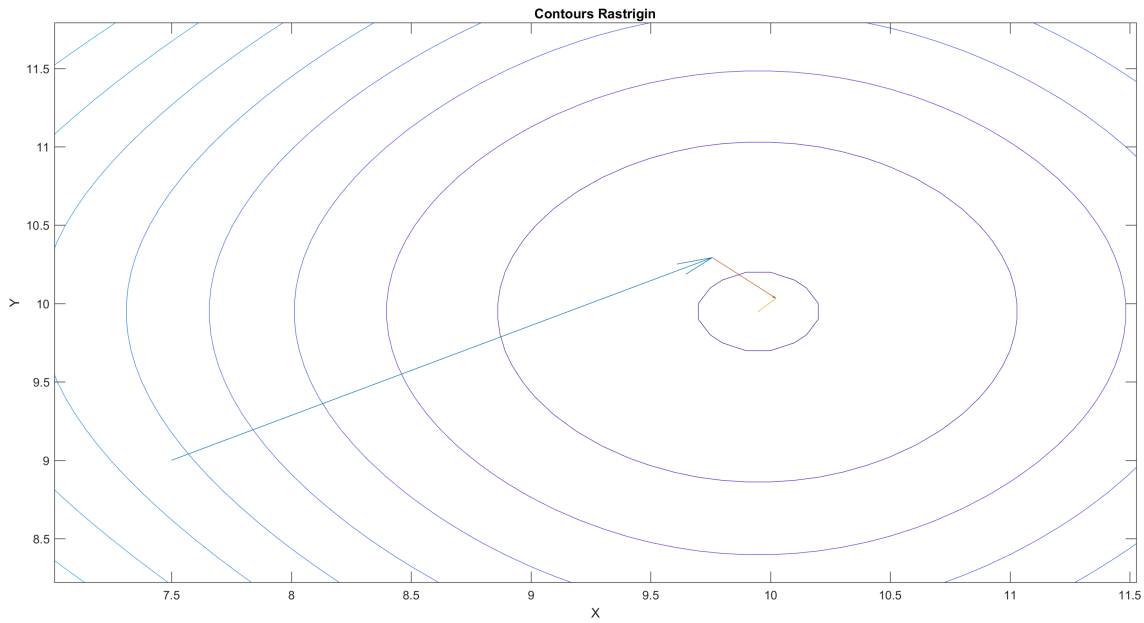
$$\begin{bmatrix} 7.5000 \\ 9.0000 \end{bmatrix} \begin{bmatrix} 9.7574 \\ 10.2931 \end{bmatrix} \begin{bmatrix} 10.0231 \\ 10.0331 \end{bmatrix} \begin{bmatrix} 9.9503 \\ 9.9504 \end{bmatrix} \begin{bmatrix} 9.9496 \\ 9.9496 \end{bmatrix}$$

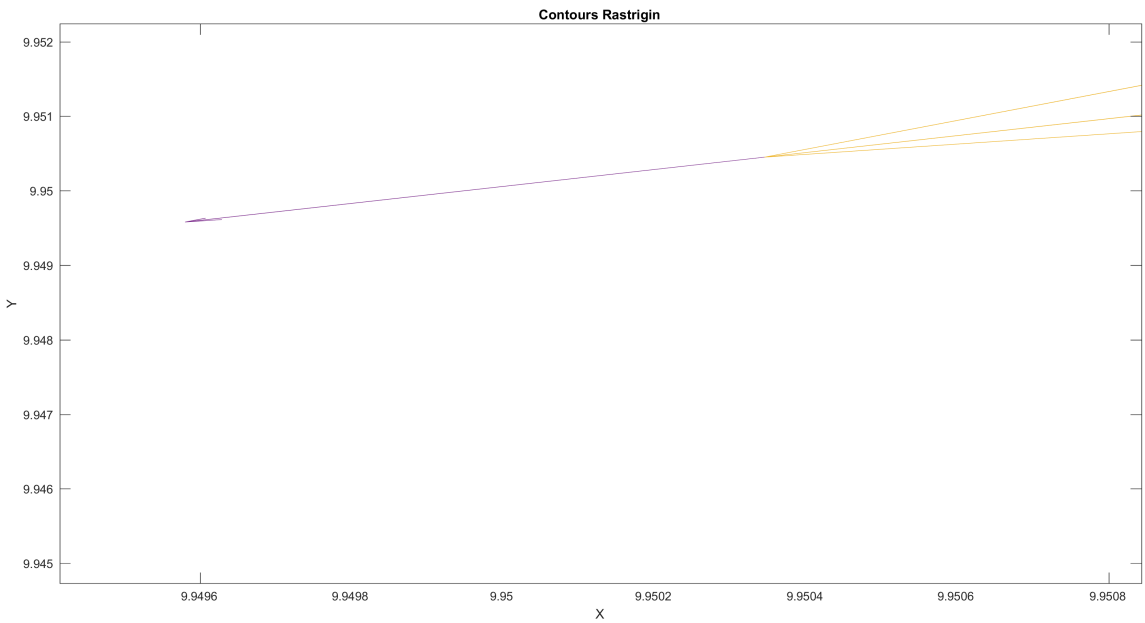
The number of iterations needed is 5.

The local minimum is found at $X^* = \begin{bmatrix} 9.9496 \\ 9.9496 \end{bmatrix}$

The value of the function at this minimum is 1.9899.







b) $x^{(0)} = \begin{bmatrix} -7 \\ -7.5 \end{bmatrix}$

We get :-

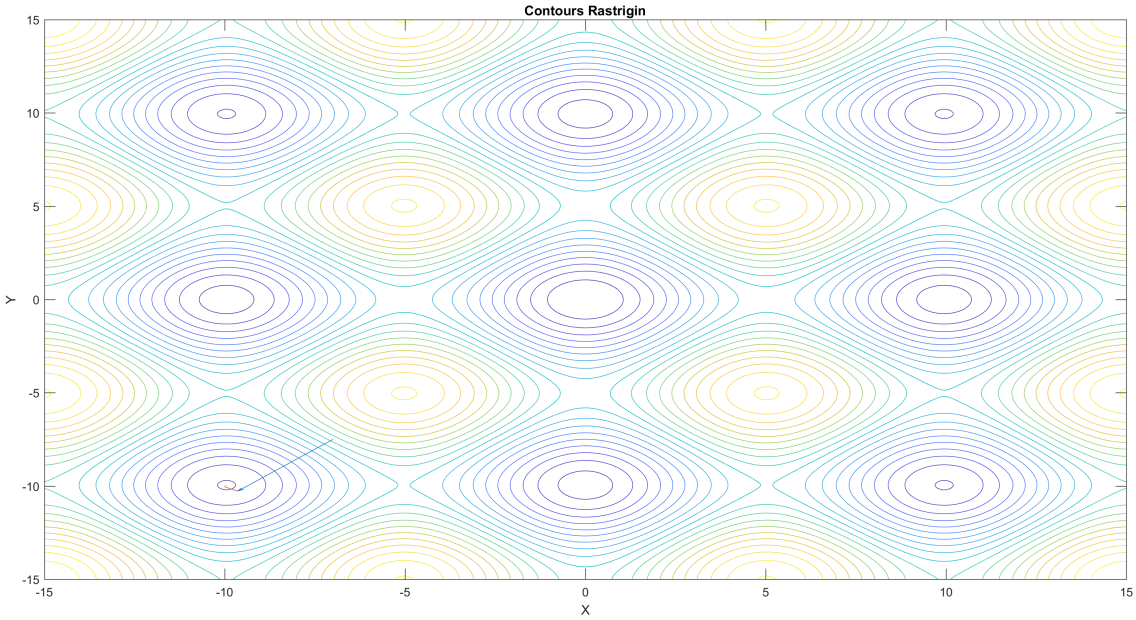
The points obtained by Powell Conjugate Gradient Algorithm are:

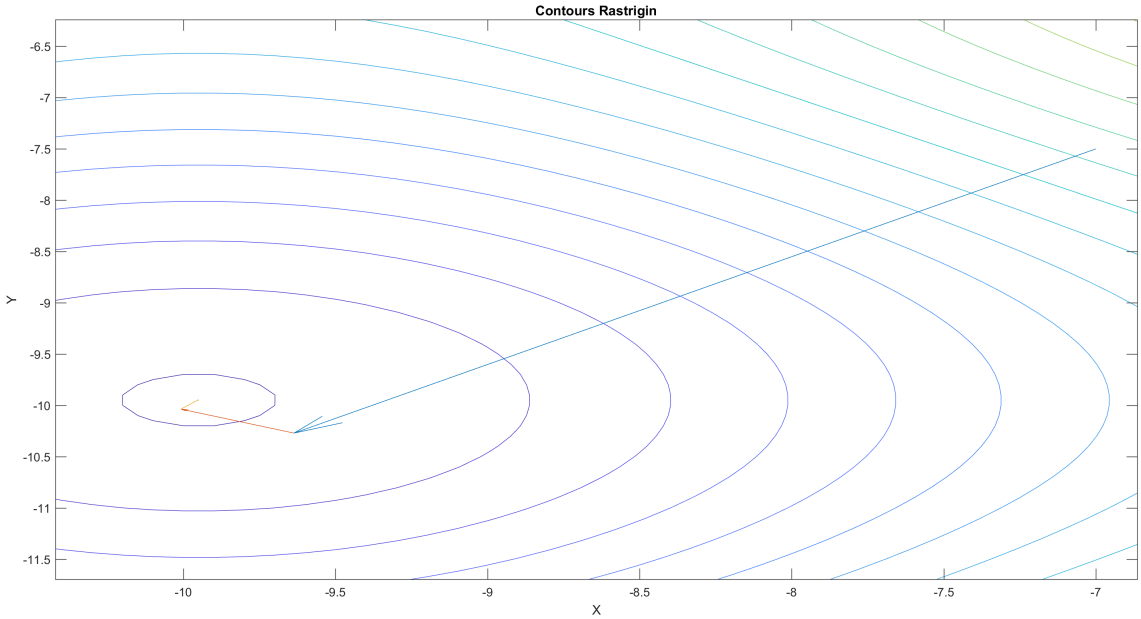
$$\begin{bmatrix} -7.0000 \\ -7.5000 \end{bmatrix} \begin{bmatrix} -9.6365 \\ -10.2709 \end{bmatrix} \begin{bmatrix} -10.0080 \\ -10.0370 \end{bmatrix} \begin{bmatrix} -9.9517 \\ -9.9474 \end{bmatrix} \begin{bmatrix} -9.9519 \\ -9.9493 \end{bmatrix} \begin{bmatrix} -9.9498 \\ -9.9487 \end{bmatrix} \begin{bmatrix} -9.9494 \\ -9.9488 \end{bmatrix} \begin{bmatrix} -9.9496 \\ -9.9496 \end{bmatrix}$$

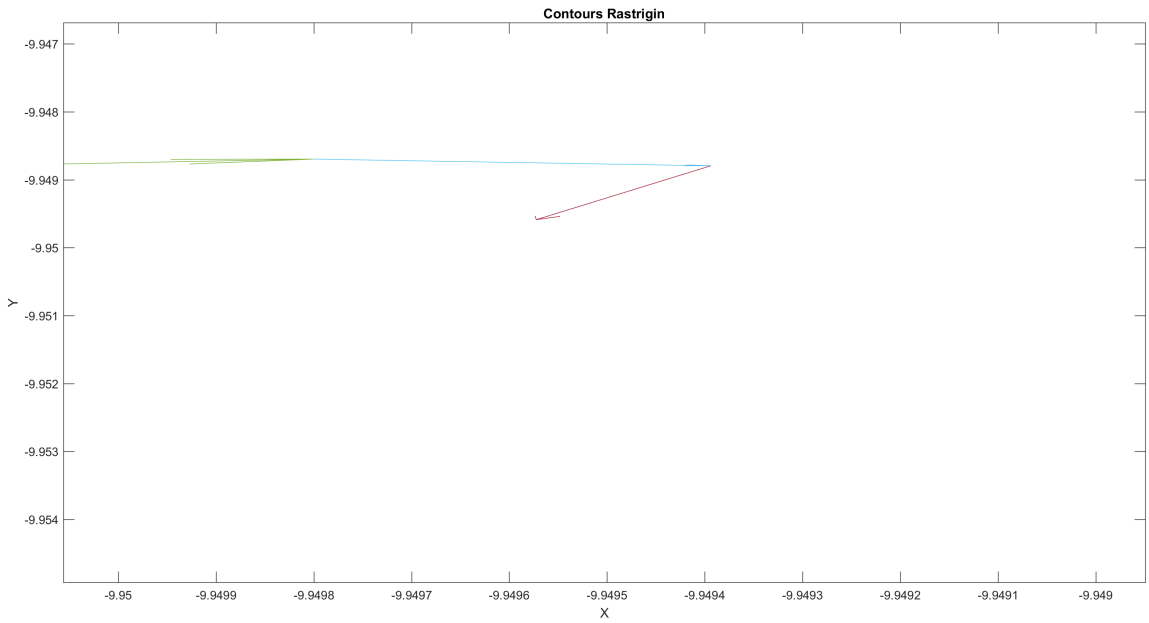
The number of iterations needed is 8.

The local minimum is found at $X^* = \begin{bmatrix} -9.9496 \\ -9.9496 \end{bmatrix}$

The value of the function at this minimum is 1.9899.


Contours Rastrigin
15
10
5
0
-5
-10
-15
Y
-15
-10
-5
0
5
10
15
X


Contours Rastrigin
-6.5
-7
-7.5
-8
-8.5
-9
-9.5
-10
-10.5
-11
-11.5
Y
-10
-9.5
-9
-8.5
-8
-7.5
-7
X



MATLAB CODE

```
%% Powell Conguate Gradient First Point

syms x y

f= 20 + (0.01)*(x.^2) +  (0.01)*(y.^2) - 10*(cos(0.2*pi*x)+cos(0.2*pi*y));

alphas=[];
betas=[];
x0=[7.5;9];

X=[];
l=[];
d=[];

i=2;

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

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

d(:,1)=-l(:,1);

while norm(l(:,i-1))>=0.0001

    alphas(i-1)=alphaFibonacci(X(:,i-1),d(:,i-1));

    X(:,i)=X(:,i-1)+alphas(i-1)*d(:,i-1);

    l(:,i)=subs(grad_f,{x,y},{X(1,i),X(2,i)});

    betas(i-1)=powel(0,(transpose(l(:,i))*(l(:,i)-l(:,i-1))/(transpose(l(:,i-1))*l(:,i-1))));

    d(:,i)=-l(:,i)+betas(i-1)*d(i-1);

    i=i+1;
end

N=i-1;
disp('The points obtained by Powel Conjugate Gradient Algorithm 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)})));


%% Powell Conguate Gradient Second Point

syms x y

f= 20 + (0.01)*(x.^2) +  (0.01)*(y.^2) - 10*(cos(0.2*pi*x)+cos(0.2*pi*y));

alphas=[];
betas=[];
x0=[-7;-7.5];

X=[];
l=[];
d=[];

i=2;

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

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

d(:,1)=-l(:,1);

while norm(l(:,i-1))>=0.0001

    alphas(i-1)=alphaFibonacci(X(:,i-1),d(:,i-1));

    X(:,i)=X(:,i-1)+alphas(i-1)*d(:,i-1);

    l(:,i)=subs(grad_f,{x,y},{X(1,i),X(2,i)});

    betas(i-1)=powel(0,(transpose(l(:,i))*(l(:,i)-l(:,i-1))/(transpose(l(:,i-1))*l(:,i-1))));

    d(:,i)=-l(:,i)+betas(i-1)*d(i-1);

    i=i+1;
end

N=i-1;
disp('The points obtained by Powel Conjugate Gradient Algorithm 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)})));

%% Arrows Plot
a=-15:0.1:15;
b=a;
[x,y]=meshgrid(a);

z= 20 + (0.01)*(x.^2) +  (0.01)*(y.^2) - 10*(cos(0.2*pi*x)+cos(0.2*pi*y));
%z = 3*(1-x).^2.*exp(-(y.^2) - (x+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 Rastrigin');

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

%% FUNCTION FOR Powel - computing the Beta values
function [b]=powel(u,v)

if(u>=v)
    b=u;
end

if(v>u)
    b=v;
end

end
```

Answer 3

The main code for all the three **Quasi Newton Algorithms** is the same. Only the value of ΔH changes, So, the functions used for computing ΔH for every Quasi-Newton Algorithm are included below the corresponding algorithm.

The Main MATLAB code for Quasi-Newton Method is shown after all the three Algorithms.

The objective is to minimize the Rastrigin's function using the **Quasi Newton - Rank One Algorithm**
To Run this **Quasi Newton - Rank One Algorithm** , comment the other two BFGS and DFP functions in the main code.

$f(x_1, x_2) = 20 + (\frac{x_1}{10})^2 + (\frac{x_2}{10})^2 - 10(cos(\frac{2\pi x_1}{10}) + cos(\frac{2\pi x_2}{10}))$

When we run the MATLAB code for:-

a) $x^{(0)} = \begin{bmatrix} 7.5 \\ 9 \end{bmatrix}$

We get :-

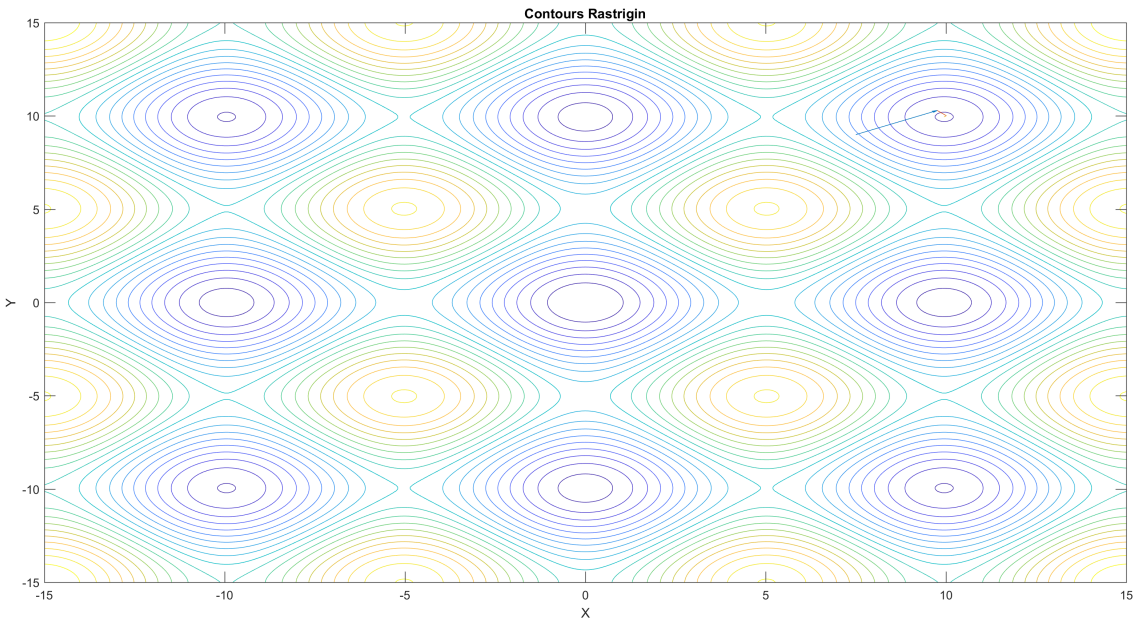
The points obtained by Rank One Algorithm are:

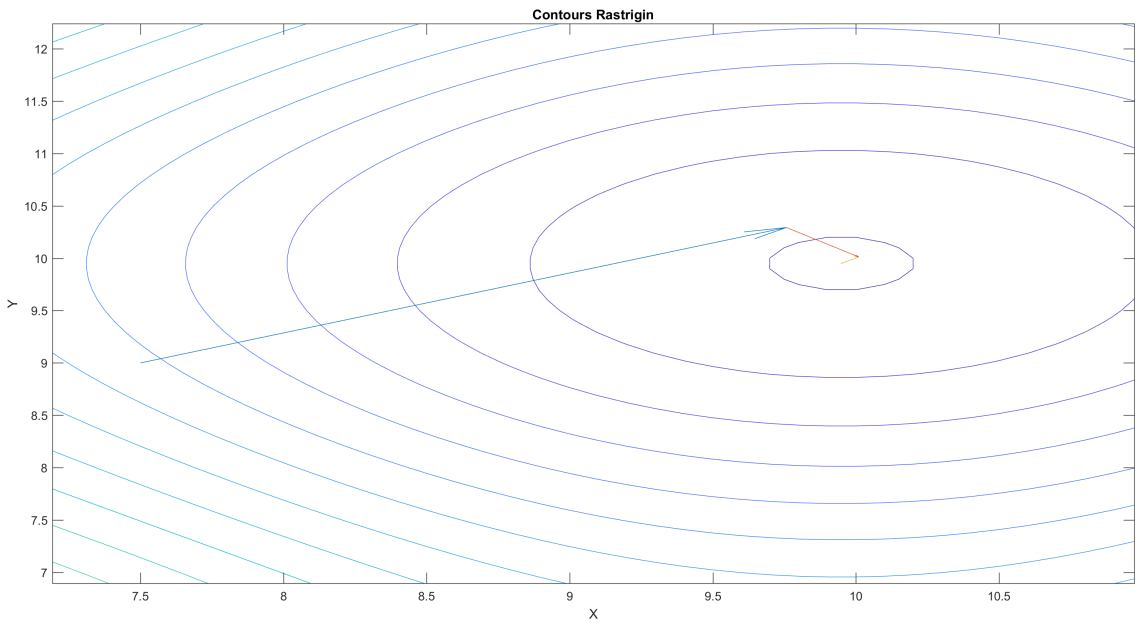
$$\begin{bmatrix} 7.5000 \\ 9.0000 \end{bmatrix} \begin{bmatrix} 9.7574 \\ 10.2931 \end{bmatrix} \begin{bmatrix} 10.0108 \\ 10.0113 \end{bmatrix} \begin{bmatrix} 9.9491 \\ 9.9508 \end{bmatrix} \begin{bmatrix} 9.9496 \\ 9.9496 \end{bmatrix}$$

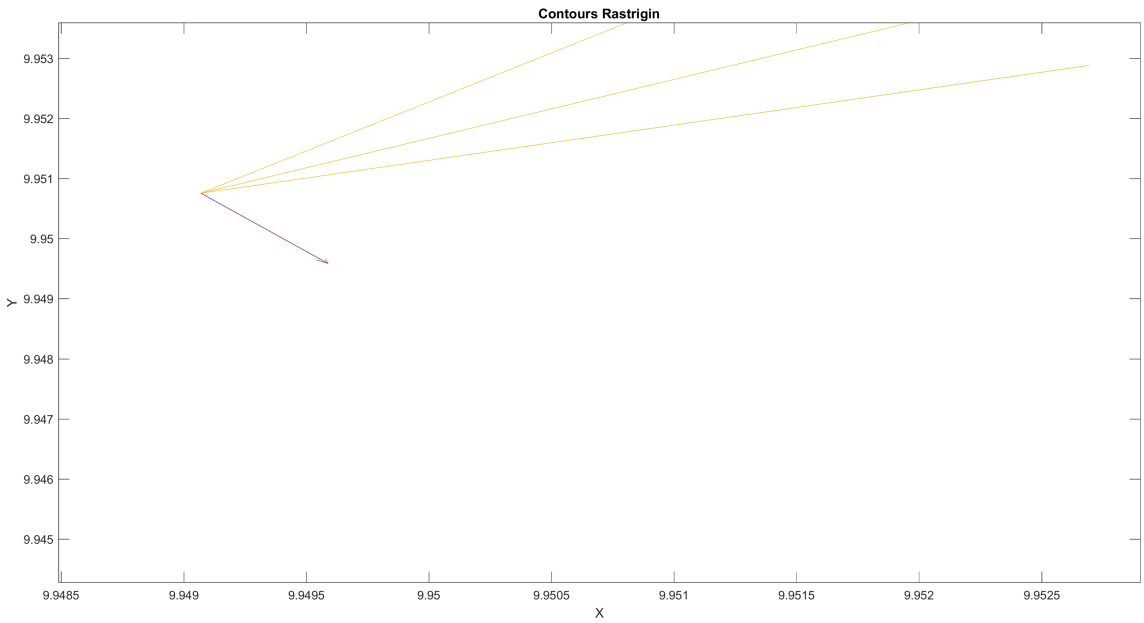
The number of iterations needed is 5.

The local minimum is found at $X^* = \begin{bmatrix} 9.9496 \\ 9.9496 \end{bmatrix}$

The value of the function at this minimum is 1.9899.


Contours Rastrigin
15
10
5
0
-5
-10
-15
Y
-15
-10
-5
0
5
10
15
X


Contours Rastrigin
12
11.5
11
10.5
10
9.5
9
8.5
8
7.5
7
Y
7.5
8
8.5
9
9.5
10
10.5
X



b) $x^{(0)} = \begin{bmatrix} -7 \\ -7.5 \end{bmatrix}$

We get :-

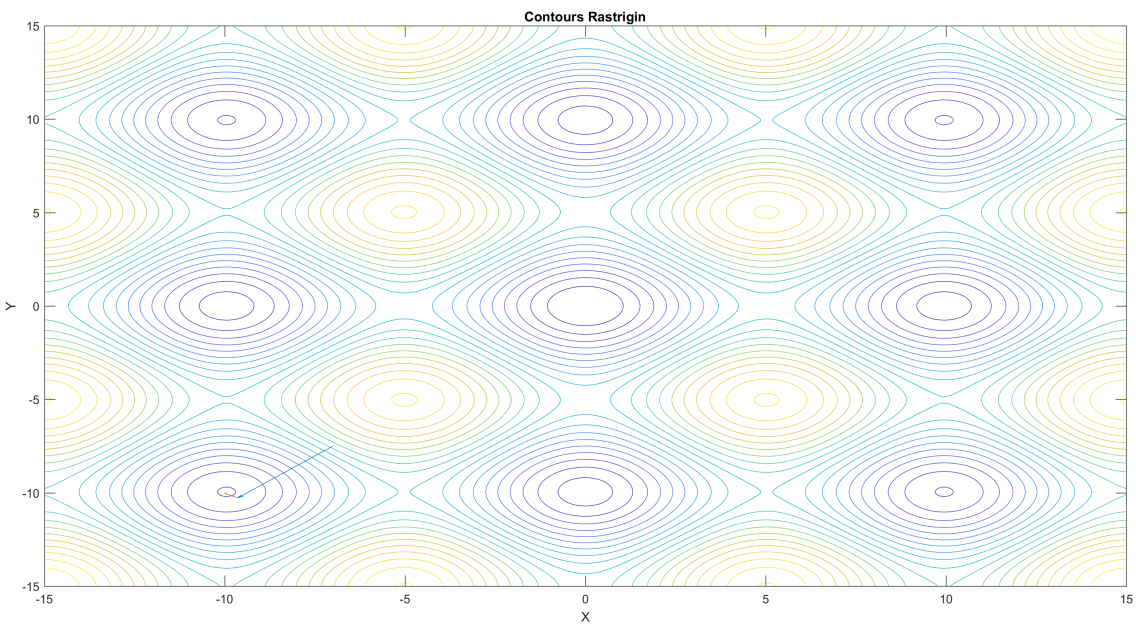
The points obtained by Rank One Algorithm are:

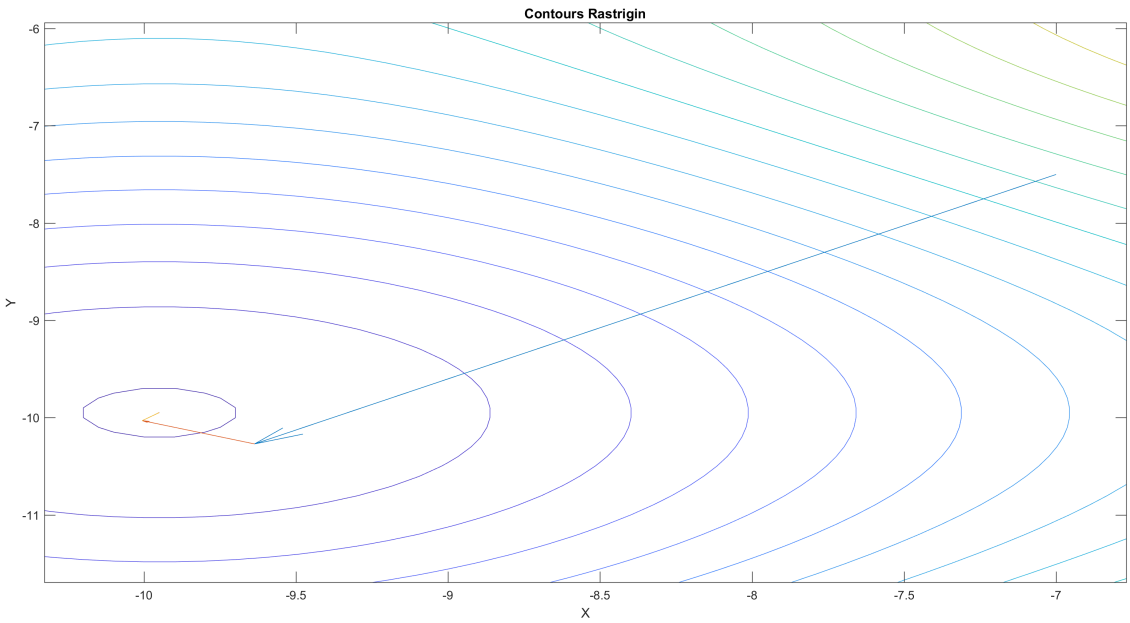
$$\begin{bmatrix} -7.0000 \\ -7.5000 \end{bmatrix} \begin{bmatrix} -9.6365 \\ -10.2709 \end{bmatrix} \begin{bmatrix} -10.0058 \\ -10.0312 \end{bmatrix} \begin{bmatrix} -9.9505 \\ -9.9493 \end{bmatrix} \begin{bmatrix} -9.9496 \\ -9.9496 \end{bmatrix}$$

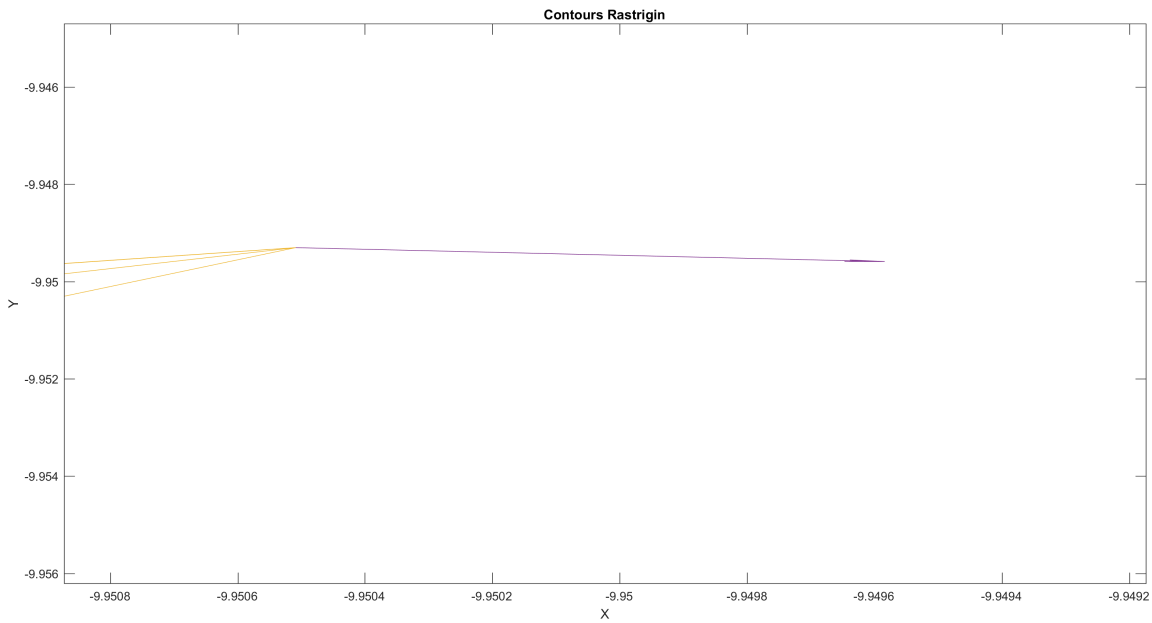
The number of iterations needed is 5.

The local minimum is found at $X^* = \begin{bmatrix} -9.9496 \\ -9.9496 \end{bmatrix}$

The value of the function at this minimum is 1.9899.







MATLAB CODE

```
%% The MAIN CODE is given in the end, Comment the BFGS, DFP function calls in main code

% Un-Comment the function call to DeltaRankone.

function [DELTA]=DeltaRankone(U,V,W)

DELTA = ((U-W*V)*transpose(U-W*V))/(transpose(V)*(U-W*V));

end
```

Answer 4

The main code for all the three **Quasi Newton Algorithms** is the same. Only the value of ΔH changes, So, the functions used for computing ΔH for every Quasi-Newton Algorithm are included below the corresponding algorithm.

The Main MATLAB code for Quasi-Newton Method is shown after all the three Algorithms.

The objective is to minimize the Rastrigin's function using the **Quasi Newton - DFP Algorithm**
To Run this **Quasi Newton - DFP** , comment the other two BFGS and RankOne functions in the main code.

$f(x_1, x_2) = 20 + (\frac{x_1}{10})^2 + (\frac{x_2}{10})^2 - 10(cos(\frac{2\pi x_1}{10}) + cos(\frac{2\pi x_2}{10}))$

When we run the MATLAB code for:-

a) $x^{(0)} = \begin{bmatrix} 7.5 \\ 9 \end{bmatrix}$

We get :-

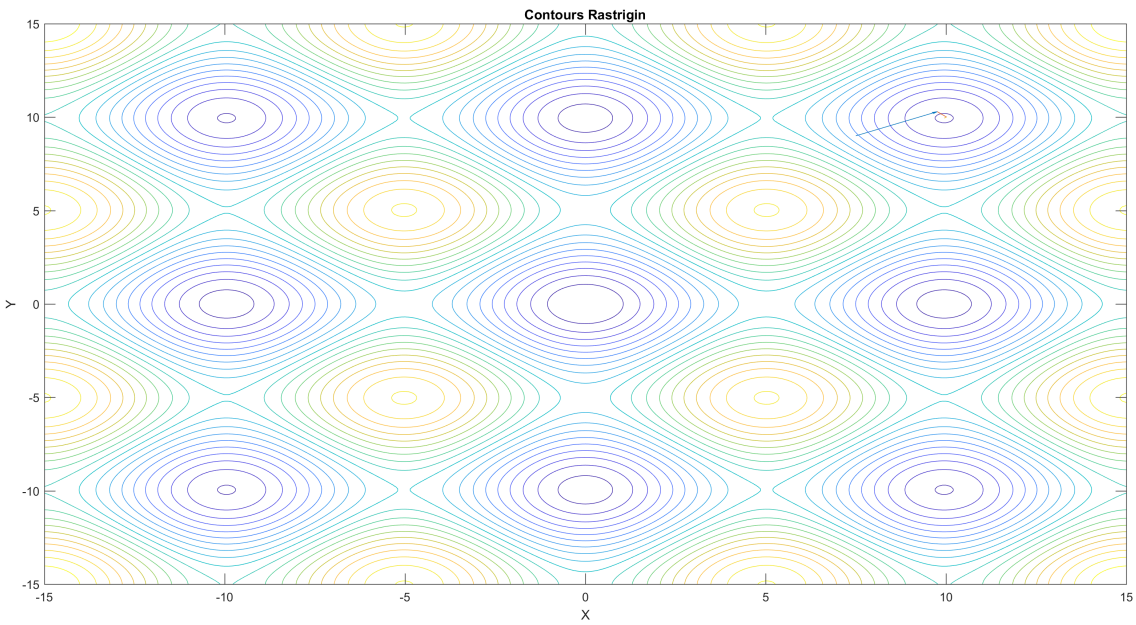
The points obtained by DFP Algorithm are:

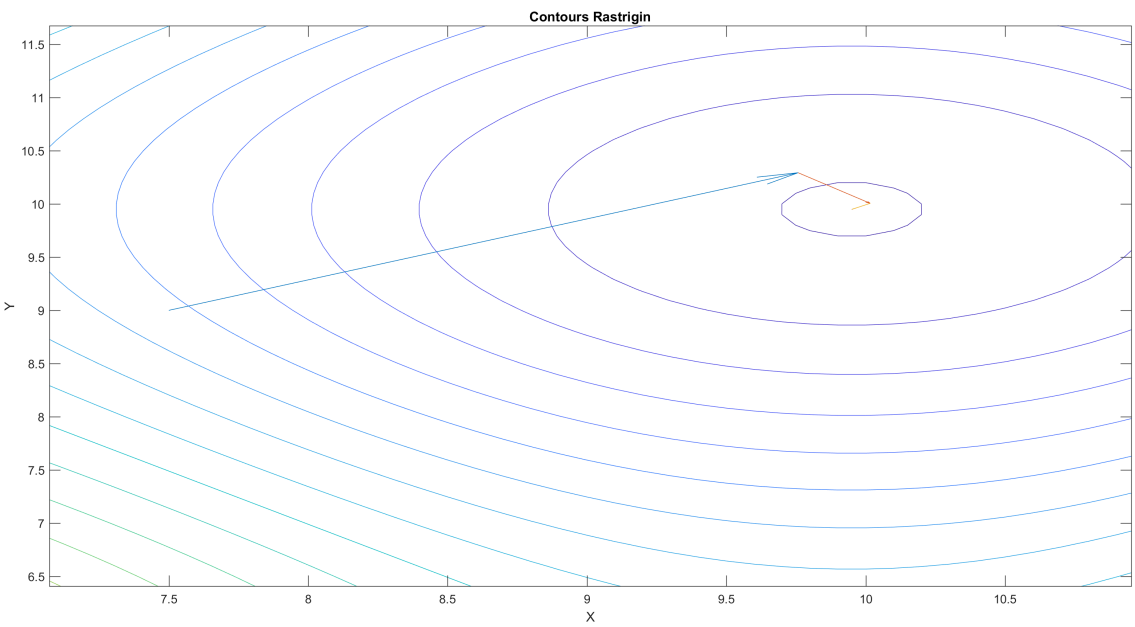
$$\begin{bmatrix} 7.5000 \\ 9.0000 \end{bmatrix} \begin{bmatrix} 9.7574 \\ 10.2931 \end{bmatrix} \begin{bmatrix} 10.0161 \\ 10.0054 \end{bmatrix} \begin{bmatrix} 9.9504 \\ 9.9492 \end{bmatrix} \begin{bmatrix} 9.9496 \\ 9.9496 \end{bmatrix}$$

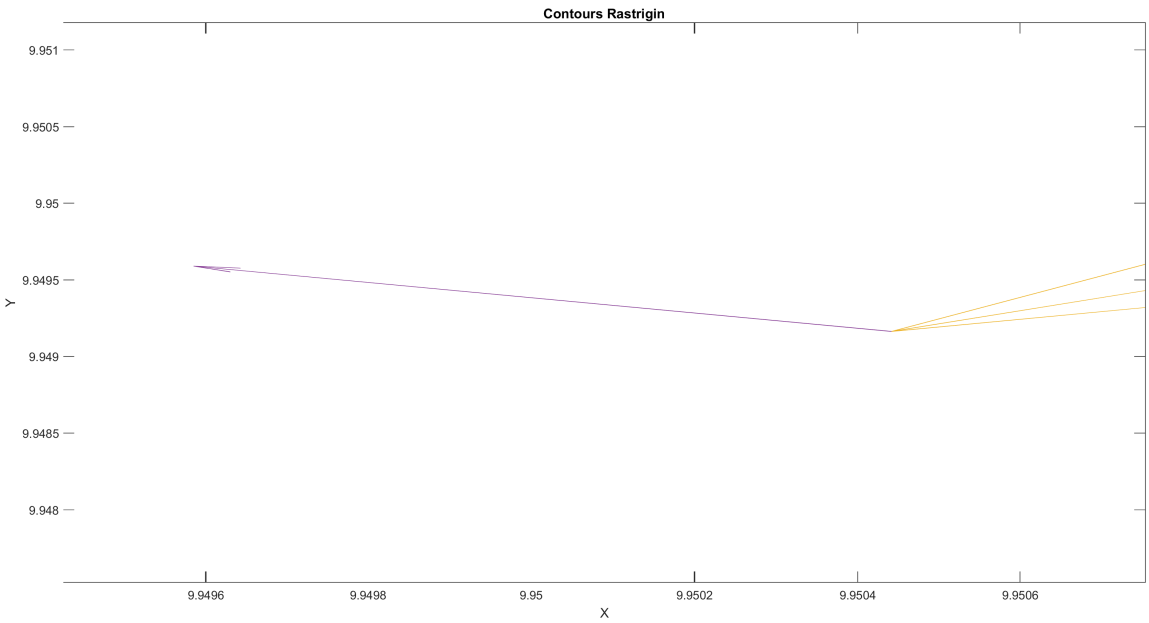
The number of iterations needed is 5.

The local minimum is found at $X^* = \begin{bmatrix} 9.9496 \\ 9.9496 \end{bmatrix}$

The value of the function at this minimum is 1.9899.







b) $x^{(0)} = \begin{bmatrix} -7 \\ -7.5 \end{bmatrix}$

We get :-

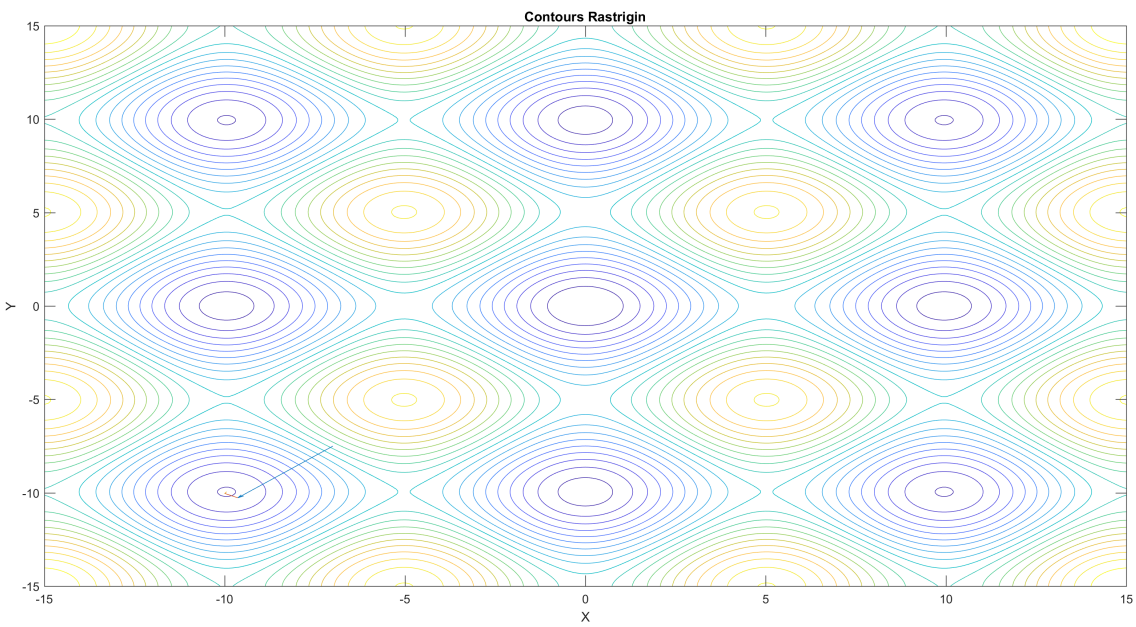
The points obtained by DFP Algorithm are:

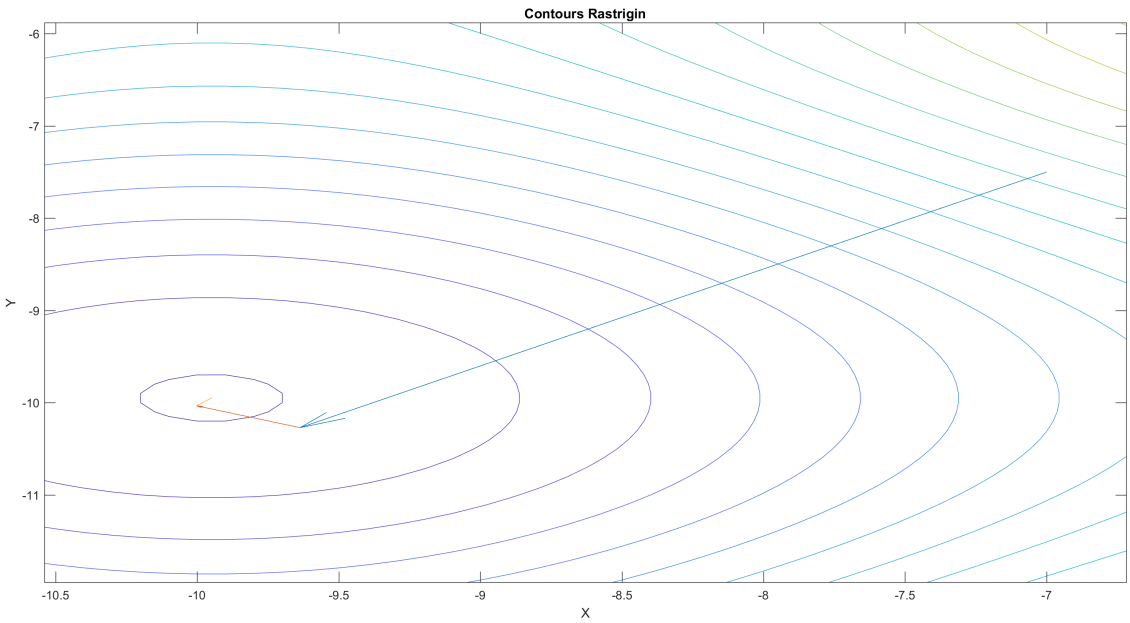
$$\begin{bmatrix} -7.0000 \\ -7.5000 \end{bmatrix} \begin{bmatrix} -9.6365 \\ -10.2709 \end{bmatrix} \begin{bmatrix} -10.0027 \\ -10.0336 \end{bmatrix} \begin{bmatrix} -9.9493 \\ -9.9503 \end{bmatrix} \begin{bmatrix} -9.9496 \\ -9.9496 \end{bmatrix}$$

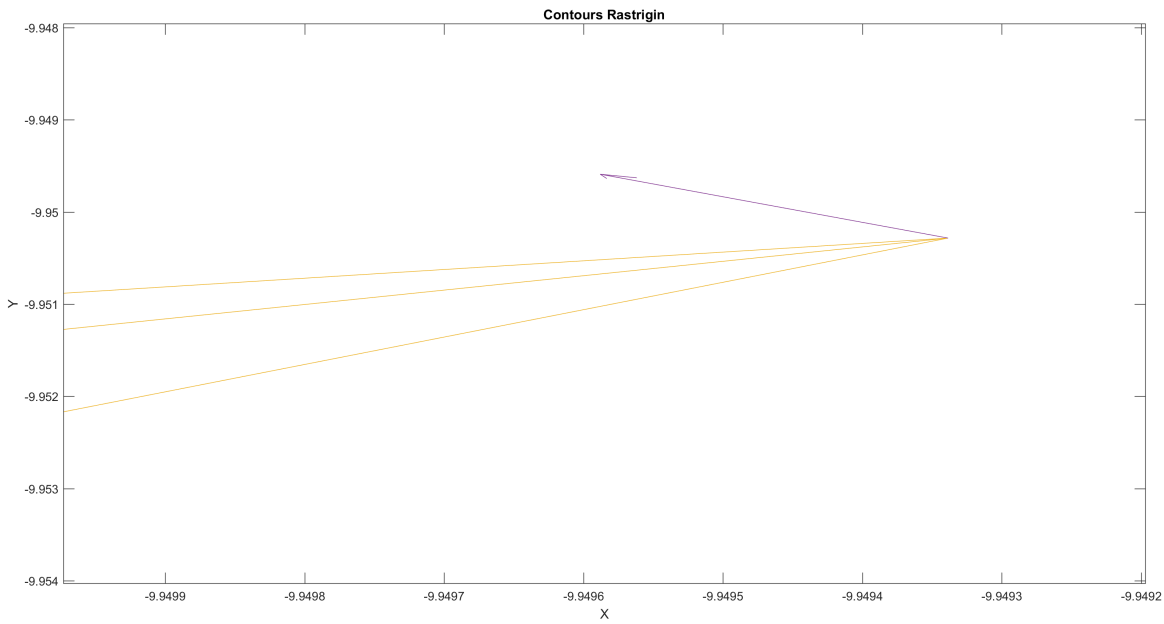
The number of iterations needed is 5.

The local minimum is found at $X^* = \begin{bmatrix} -9.9496 \\ -9.9496 \end{bmatrix}$

The value of the function at this minimum is 1.9899.







MATLAB CODE

```
%% The MAIN CODE is given in the end, Comment the BFGS, RankOne function calls in main code

% Un-Comment the function call to DFP.

function [DELTA]=DeltaDFP(U,V,W)

DELTA = (U*transpose(U))/(transpose(V)*U) - ((W*V)*transpose(W*V))/(transpose(V)*W*V) ;

end
```

Answer 5

The main code for all the three **Quasi Newton Algorithms** is the same. Only the value of ΔH changes, So, the functions used for computing ΔH for every Quasi-Newton Algorithm are included below the corresponding algorithm.

The Main MATLAB code for Quasi-Newton Method is shown after all the three Algorithms.

The objective is to minimize the Rastrigin's function using the **Quasi Newton - BFGS Algorithm**
To Run this **Quasi Newton - BFGS** , comment the other two DFP and RankOne functions in the main code.

$f(x_1, x_2) = 20 + (\frac{x_1}{10})^2 + (\frac{x_2}{10})^2 - 10(cos(\frac{2\pi x_1}{10}) + cos(\frac{2\pi x_2}{10}))$

When we run the MATLAB code for:-

a) $x^{(0)} = \begin{bmatrix} 7.5 \\ 9 \end{bmatrix}$

We get :-

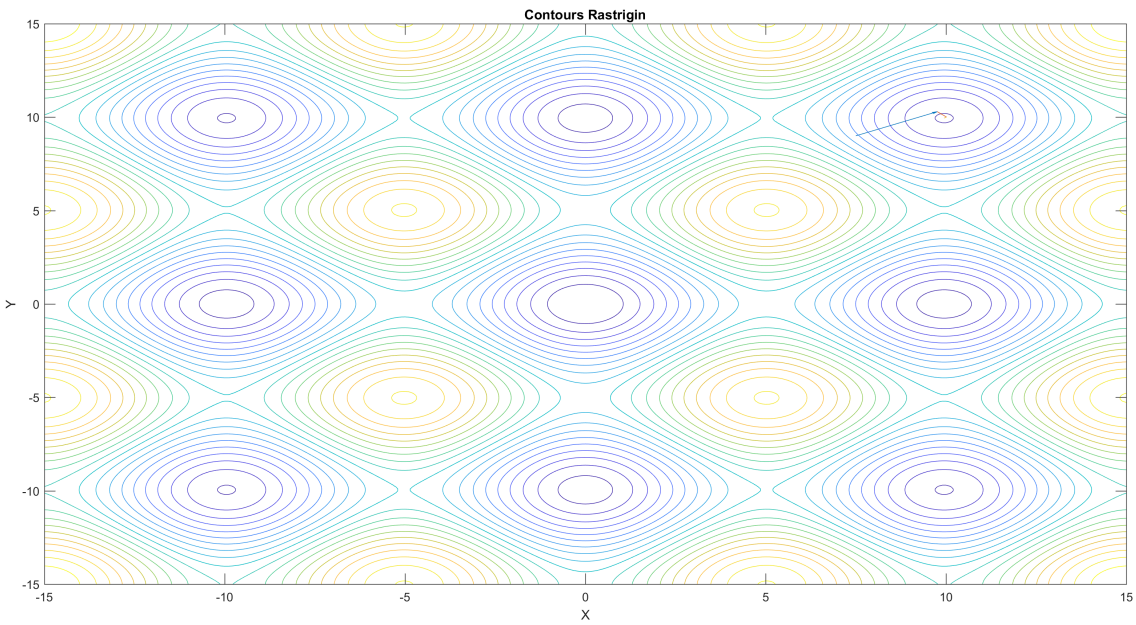
The points obtained by BFGS Algorithm are:

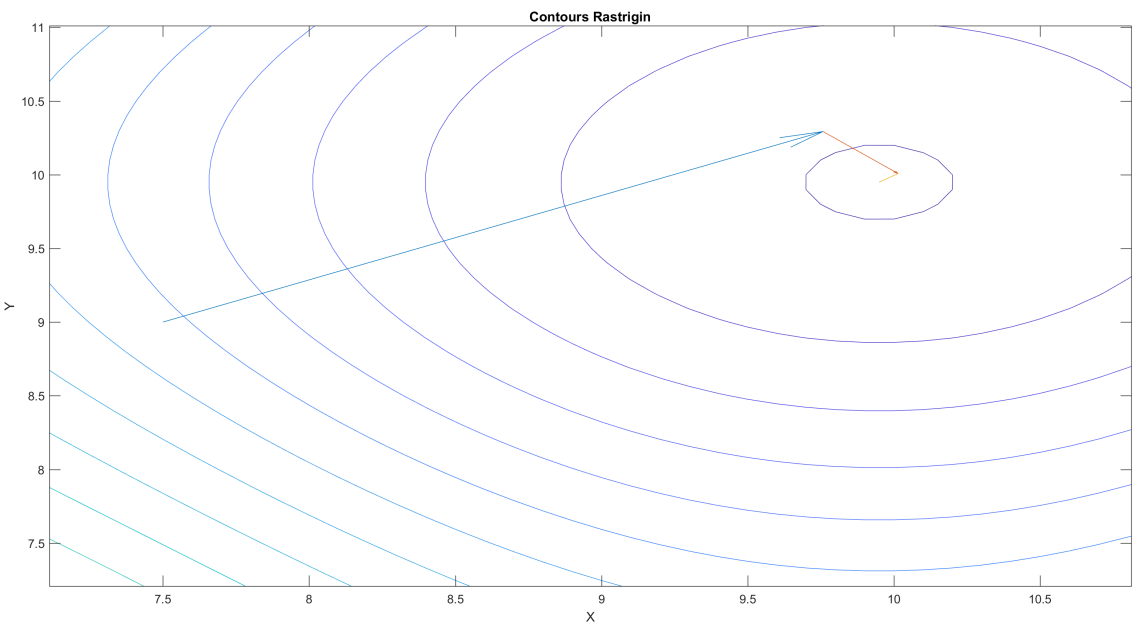
$$\begin{bmatrix} 7.5000 \\ 9.0000 \end{bmatrix} \begin{bmatrix} 9.7574 \\ 10.2931 \end{bmatrix} \begin{bmatrix} 10.0140 \\ 10.0078 \end{bmatrix} \begin{bmatrix} 9.9499 \\ 9.9498 \end{bmatrix} \begin{bmatrix} 9.9496 \\ 9.9496 \end{bmatrix}$$

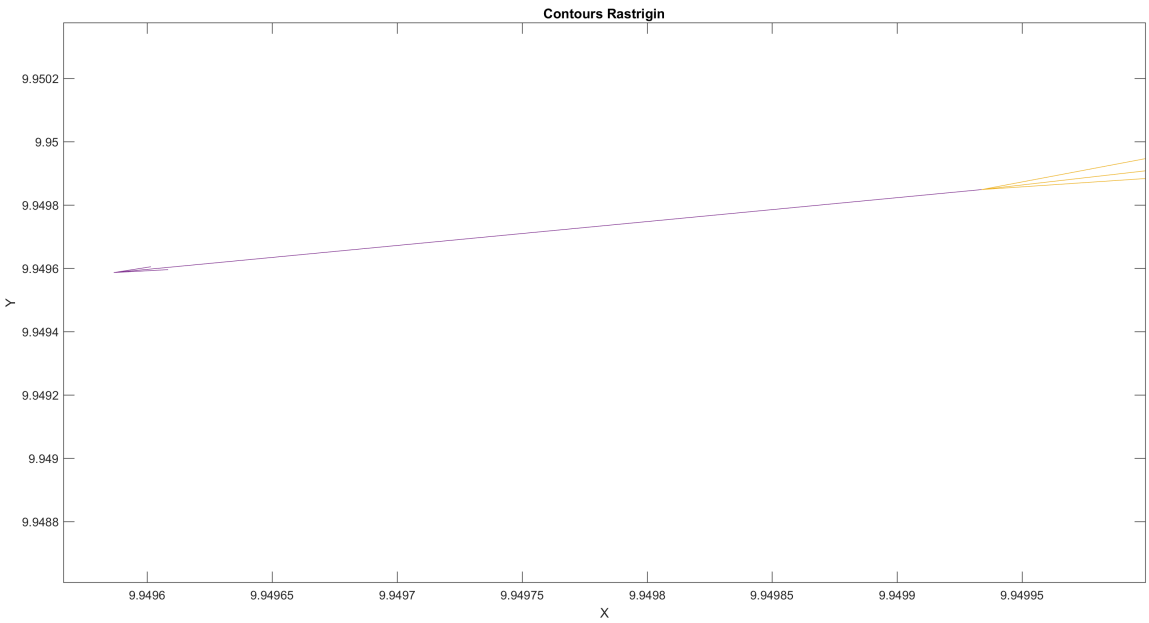
The number of iterations needed is 5.

The local minimum is found at $X^* = \begin{bmatrix} 9.9496 \\ 9.9496 \end{bmatrix}$

The value of the function at this minimum is 1.9899.







b) $x^{(0)} = \begin{bmatrix} -7 \\ -7.5 \end{bmatrix}$

We get :-

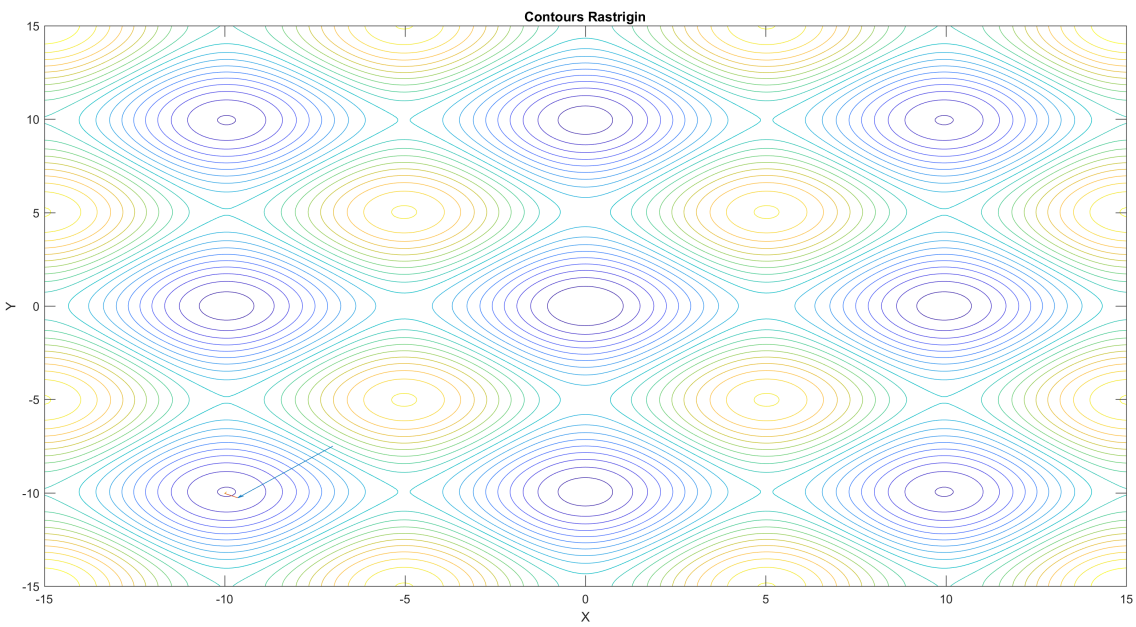
The points obtained by BFGS Algorithm are:

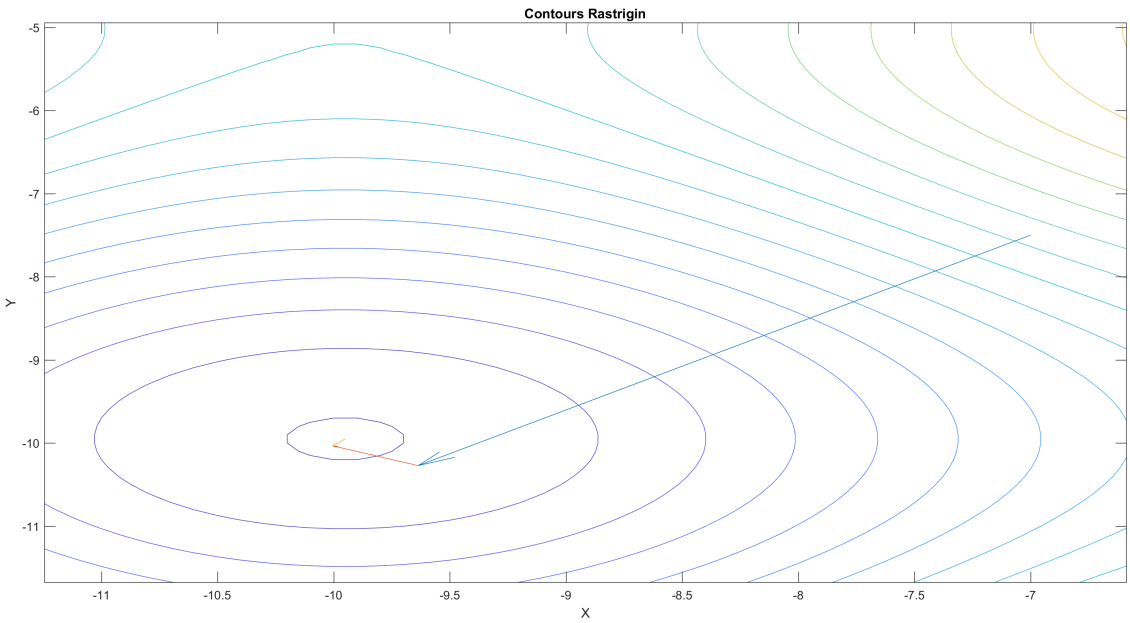
$$\begin{bmatrix} -7.0000 \\ -7.5000 \end{bmatrix} \begin{bmatrix} -9.6365 \\ -10.2709 \end{bmatrix} \begin{bmatrix} -10.0041 \\ -10.0330 \end{bmatrix} \begin{bmatrix} -9.9498 \\ -9.9498 \end{bmatrix} \begin{bmatrix} -9.9496 \\ -9.9496 \end{bmatrix}$$

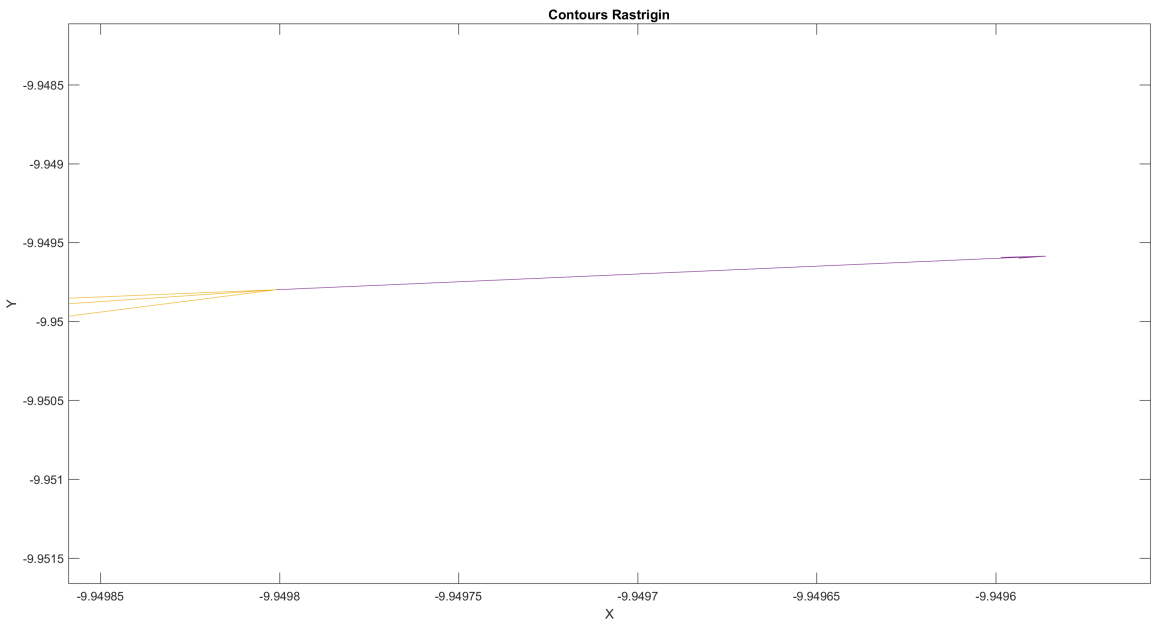
The number of iterations needed is 5.

The local minimum is found at $X^* = \begin{bmatrix} -9.9496 \\ -9.9496 \end{bmatrix}$

The value of the function at this minimum is 1.9899.







MATLAB CODE

```
%% BFGS FUNCTION for computing DELTA H

function [DELTA]=DeltaBFGS(U,V,W)

Q = (1+(transpose(V)*W*V)/(transpose(V)*U));

R = Q*(U*transpose(U))/(transpose(V)*U);

S = ((W*V*transpose(U)) + transpose(W*V*transpose(U)))/(transpose(V)*U);

DELTA = R - S ;

end

% THE MAIN CODE FOR RUNNING QUASI NEWTON METHODS, and calling DELTA H functions

%% Quasi Newton Correction First Point
```

```
syms x y

f= 20 + (0.01)*(x.^2) +  (0.01)*(y.^2) - 10*(cos(0.2*pi*x)+cos(0.2*pi*y));

alphas=[];
x0=[7.5;9];
H0=eye(2);

X=[];
l=[];
d=[];

i=2;

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

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

d(:,1)=-H0*l(:,1);
Hi=H0;

while norm(l(:,i-1))>=0.0001

    alphas(i-1)=alphaFibonacci(X(:,i-1),d(:,i-1));

    X(:,i)=X(:,i-1)+alphas(i-1)*d(:,i-1);

    l(:,i)=subs(grad_f,{x,y},{X(1,i),X(2,i)});

    deltaX=X(:,i)-X(:,i-1);
    deltaG=l(:,i)-l(:,i-1);

   % deltaH=DeltaRankone(deltaX,deltaG,Hi);
   % deltaH=DeltaDFP(deltaX,deltaG,Hi);
    deltaH=DeltaBFGS(deltaX,deltaG,Hi);

    Hf=Hi+deltaH;
    Hi=Hf;

    % Making Hf to be Positive definite for Rank-One algorithm.

  PSD=eig(Hf);
    if (PSD(1)<=0) || (PSD(2)<=0)
        Hf=eye(2);
    end

    d(:,i)=-Hf*l(:,i);

    i=i+1;
end

N=i-1;
disp('The points obtained by Quasi-Newton Algorithm 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)})));

%% Quasi Newton Second Point
syms x y
```

```
f= 20 + (0.01)*(x.^2) +  (0.01)*(y.^2) - 10*(cos(0.2*pi*x)+cos(0.2*pi*y));

alphas=[];
x0=[-7;-7.5];
H0=eye(2);

X=[];
l=[];
d=[];

i=2;

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

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

d(:,1)=-H0*l(:,1);
Hi=H0;

while norm(l(:,i-1))>=0.0001

    alphas(i-1)=alphaFibonacci(X(:,i-1),d(:,i-1));

    X(:,i)=X(:,i-1)+alphas(i-1)*d(:,i-1);

    l(:,i)=subs(grad_f,{x,y},{X(1,i),X(2,i)});

    deltaX=X(:,i)-X(:,i-1);
    deltaG=l(:,i)-l(:,i-1);

  % deltaH=DeltaRankone(deltaX,deltaG,Hi);
  % deltaH=DeltaDFP(deltaX,deltaG,Hi);
    deltaH=DeltaBFGS(deltaX,deltaG,Hi);

    Hf=Hi+deltaH;
    Hi=Hf;

      PSD=eig(Hf);
    if (PSD(1)<=0) || (PSD(2)<=0)
        Hf=eye(2);
    end

    d(:,i)=-Hf*l(:,i);

    i=i+1;
end

N=i-1;
disp('The points obtained by Quasi-Newton Algorithm 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)})));

%% Arrows Plot
a=-15:0.1:15;
b=a;
[x,y]=meshgrid(a);
```

```
z= 20 + (0.01)*(x.^2) +  (0.01)*(y.^2) - 10*(cos(0.2*pi*x)+cos(0.2*pi*y));
%z = 3*(1-x).^2.*exp(-(y.^2) - (x+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 Rastrigin');

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

%% Function for Fibonacci Search
% TO BE USED In same folder for running any of the codes above.

function  [result]=alphaFibonacci(H,D)

syms c d al

z= 20 + (0.01)*(c.^2) +  (0.01)*(d.^2) - 10*(cos(0.2*pi*c)+cos(0.2*pi*d));

Y0=H;
m=D;

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);

% Fibonacci Search

E=0.01;
```

```
epsilon=0.001;

Dist=norm(q-p);

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

F_N1=(1+2*epsilon)*(Dist/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;

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

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

pfin=p;
qfin=q;

result=(pfin+qfin)/2;


end
```

THE END