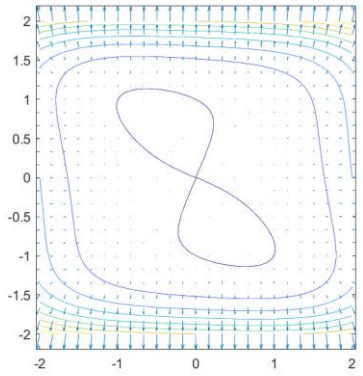
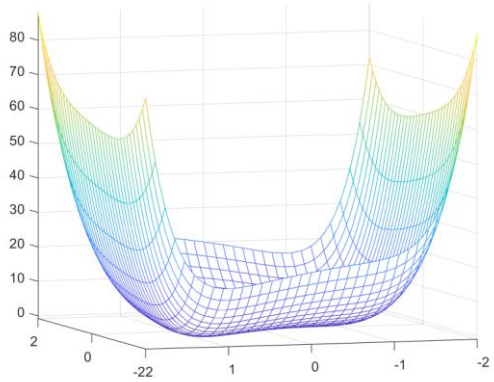
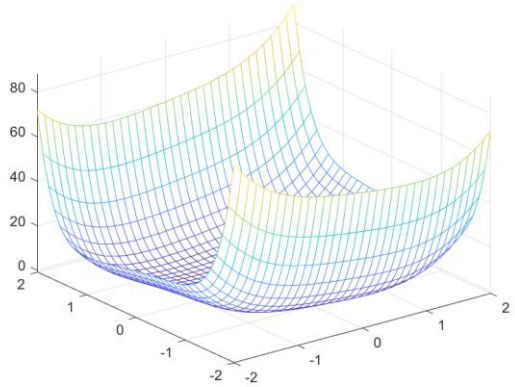
Requirements:

Plot the graph of Function $f(x1;\ x2) = x_1^2 + x_1^4 + x_2^6 - x_2^2 + 2x1x2$

```
f=@ (x) x(1)^2+x(1)^4+x(2)^6-x(2)^2+2*(x(1)*x(2));
F=@ (x,y) f([x,y]);
figure;fmesh(F,[-2,2])
figure; fcontour(F,[-2,2])
axis equal
hold on;
% Define the domain % Create a grid of x and y values
% Compute the gradient of the function
x1 = linspace(-2,2,25);
x2=x1;
[X1, X2] = meshgrid(x1,x2);
F1 = 2*X1 + 4*X1.^3 + 2*X2;
F2 = 6*X2.^5 - 2*X2 + 2*X1;
quiver(X1, X2, F1, F2)
```



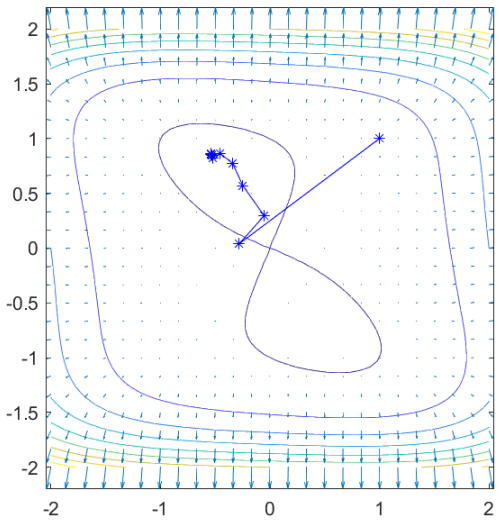




2- Code the following methods:

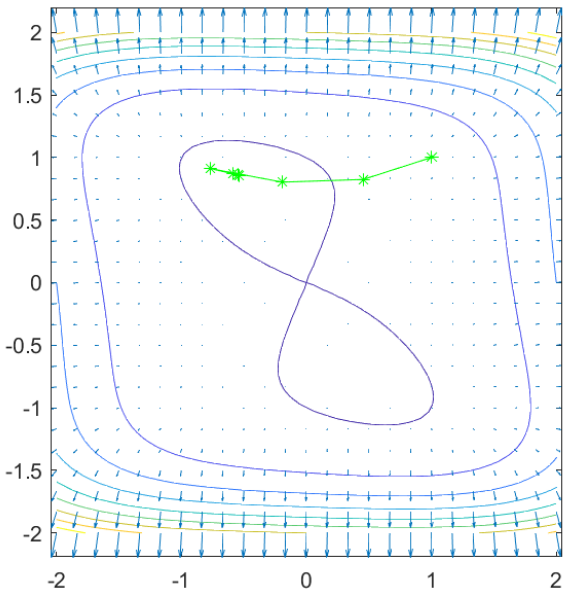
A- Gradient method

```
% Fisrt derivative for the given function
g=@(x) [2*x(1)+4*x(1)^3+2*x(2); 6*x(2)^5-2*x(2)+2*x(1)];
[xoptG,foptG,kG]=mygradient([1.0;1.0],1e-3,3000,f,g)
% gradient function
function [xoptG,foptG,kG]=mygradient(x0,e,maxit,f,g)
  kG=0;
  if ~iscolumn(x0)
        error('The initial vector has to be a column vector')
  end
  while norm(g(x0))>e
     alpha=backtG(0.4,0.01,0.4,f,g,x0);
     x=x0-alpha*g(x0);
     plot([x(1),x0(1)],[x(2),x0(2)],'b*-')
     x0=x
     kG=kG+1
     if kG>maxit
        error('The number of steps exceeds the maxit')
     end
  end
  xoptG=x;
  foptG=f(xoptG);
end
function alpha=backtG(alpha,c,rho,f,g,x)
    while f(x-alpha*g(x))>f(x)-alpha*c*dot(g(x),g(x))
        alpha=rho*alpha;
    end
```



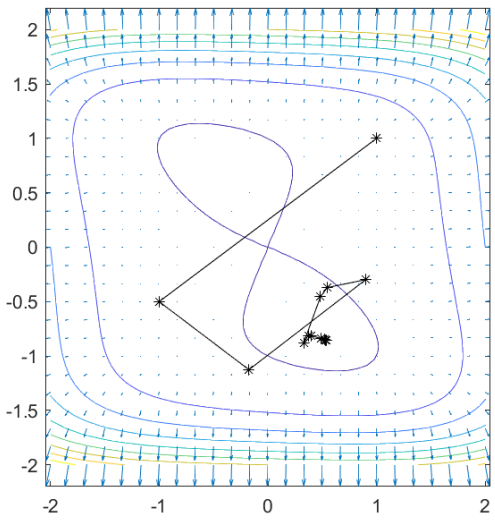
B- Newton’s method:

```
% We need the second derivative for the Newton’s method
% for the Hessian metrix

h=@(x) [2+12*x(1)^2,2 ; 2,30*x(2)^4-2];
[xoptN,foptN,k2]= mynew([1.0;1.0], f,g,h,1e-3,3000)
function [xoptN, foptN,kN] = mynew(x0,f,g,h,e,maxit)
   kN=0;
    while norm(g(x0))>e
       p= -h(x0)\g(x0);
       x1=x0 +p;
       kN=kN+1;
       if kN >maxit
           error('increase the value of maxit')
       end
       plot([x0(1),x1(1)],[x0(2),x1(2)],'g*-')
       x0= x1;
    end
    xoptN=x1
    foptN= f(xoptN) % the optimal value for the function

end
```



C- BFGS method:

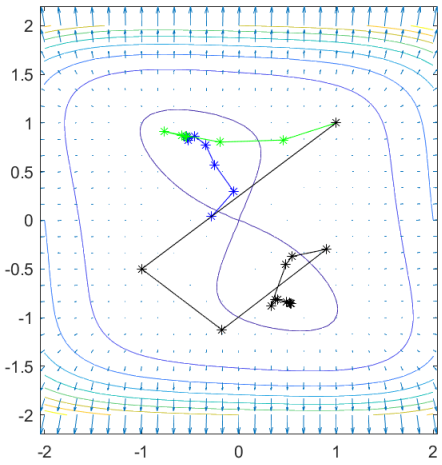
```
[xoptBFGS,foptBFGS,kBFGS]=mybfgs([1.0;1.0],f,g)
function [xoptB,foptB,kB]=mybfgs(x0,f,g,e,maxit)
    if nargin<4
        e=1e-3;
    end
    if nargin<5
        maxit=200;
    end
    if ~iscolumn(x0)
        error('the initial vector has to be a column vector')
    end
    kB=0;
    H=eye(length(x0));
    H0=H;
    while norm(g(x0))>e
        p=-H*g(x0);
        alpha=backtBFGS(0.5,0.01,0.5,f,g,p,x0);
        x1=x0+alpha*p;
        kB=kB+1;
        if kB>maxit
            error('maxit reached')
        end
        s=x1-x0;
        y=g(x1)-g(x0);
        H=(H0-s*y'/dot(y,s))*H*(H0-y*s'/dot(y,s))+s*s'/dot(y,s);
        plot([x0(1),x1(1)],[x0(2),x1(2)],'k*-');
        x0=x1;
    end
    xoptB=x1;
    foptB=f(xoptB);
end
function alpha=backtBFGS(alpha,c,rho,f,g,p,x)
  while f(x+alpha*p)>f(x)+alpha*c*dot(g(x),p)
    alpha=rho*alpha;
  end
end
```



The three methods together

The Xopt, Fopt, and the number of steps needed for the initial value **[**1.0;1.0**]**

METHOD	**GRADIENT**	**NEWTON**	**BFGS**
N OF STEPS	12	6	17
XOPT	[-0.5413 0.8586]	[-0.5414 0.8586]	[0.5412 -0.8586]
FOPT	-0.8872	-0.8872	-0.8872

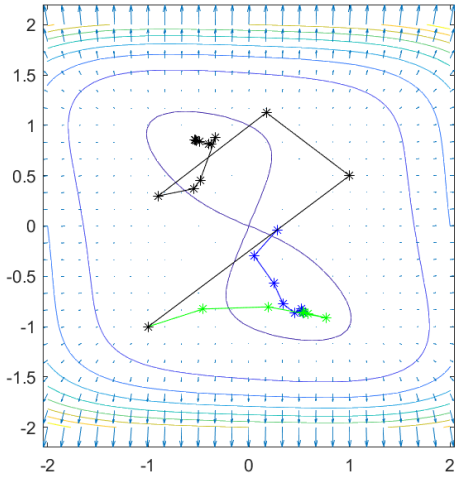


As we can see from the figures above, **Gradient** and **Newton** end at the same coordinates [0.5414, -0.8587] with the same **fopt** = -0.8872 with less steps for the **Newton's** method which mean the **Newton** is faster than Gradient, on the other hand, we found that **BFGS** method behaves differently and gave the same value of fopt but in different coordinate**.**

Let's try different initial values and see what will happens:

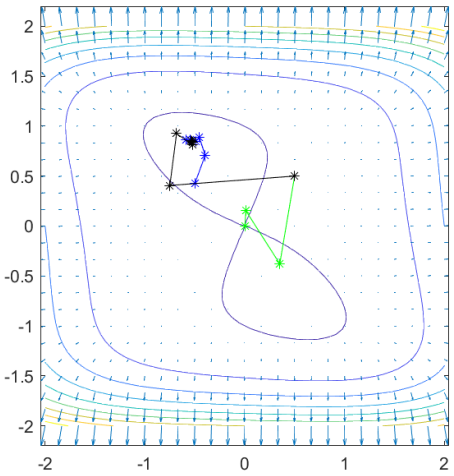
The Xopt, Fopt, and the number of steps needed for the initial value [**-1.0;-1.0**]

METHOD	GRADIENT	NEWTON	BFGS
N OF STEPS	12	6	17
XOPT	[0.5413 -0.8586]	[0.5414 -0.8586]	[-0.5412 0.8586]
FOPT	-0.8872	-0.8872	-0.8872



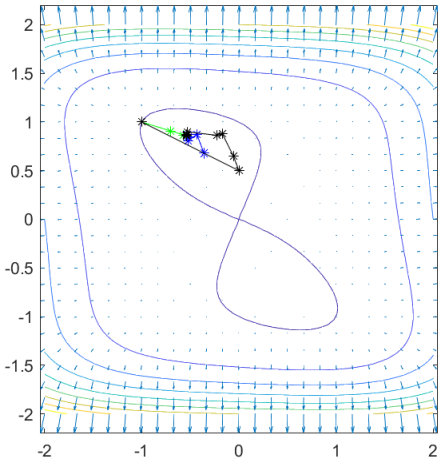
The Xopt, Fopt, and the number of steps needed for the initial value **[0.5;0.5]**

METHOD	GRADIENT	NEWTON	BFGS
N OF STEPS	10	4	12
XOPT	[-0.5414 0.8587]	1.0e-09 * [0.2414 0.2414]	[-0.5413 0.8585]
FOPT	-0.8872	1.1655e-19	-0.8872



The Xopt, Fopt, and the number of steps needed for the initial value **[-1.0;1.0]**

METHOD	**GRADIENT**	**NEWTON**	**BFGS**
N OF STEPS	9	4	14
XOPT	[-0.5413 0.8586]	[-0.5413 0.8586]	[-0.5413 0.8587]
FOPT	-0.8872	-0.8872	-0.8872



We tried different initial values and we observed that the object function has more than one local minimizer, Gradient method and Newton's method can find the first one and BFGS can find the second one on the opposite direction, however, when we change the initial value and choose a vector close to Zero, Newton's method act differently and cannot find the local minimizer, while Gradient and BFGS methods can find the local minimizer but BFGS find the local at the same coordinate this time. And at some points the three methods could lead to the same point (**the same optimal value for the function)**

Conclusion:

1- The chosen initial value could lead to different results.
2- The function has more than one local minimizer, and it is possible to find them by any method.
3- The Newton function act differently when we choose an initial vector close to zero.
4- Every time we choose different initial value, we find that the Newton's method is the faster method that find the fopt at minimum steps.
5- The function does not have a global minimizer.