Project Report

Rongzhen Wei

Introduction

In this project report, six methods of converging minimum value are studied, namely, direct search method, Newton's Method BFGS method, steepest descent method, conjugate gradient method, and Nelder-Mead method

The minimum value problem is based on two given functions, and some properties and the exact solutions are provided in section 1. In the following parts, those six methods are simply introduced and applied in Matlab coding. The issues encountered and the solving ideas when implementing the algorithm are discussed, also some doubts. The final part is a comparison of those methods.

The different behaviors of two functions are presented when the same method is applied. The necessary adjustments and the converted problems are discussed, especially in NM method section.

1. Formulation

In this section, the two given functions are studied, and the maximum or minimum solutions are provided.

(a)

The first function is a two-dimensional distribution of pollutant concentration in a channel. It reads

$$f_1(x,y) = 7.9 + 0.13x + 0.21y - 0.05x^2 - 0.016y^2 - 0.007xy$$

Sketching the 3D image, I have

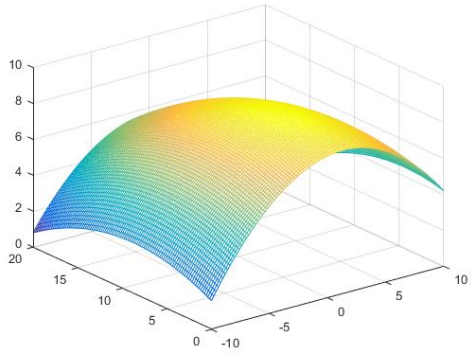

Fig.1

Based on the picture, we could refer that there probably is a maximum point around (1,1).

For finding this point, I first compute the partial derivatives, which is the Jacobian.

$$J = \nabla f_1 = \begin{bmatrix} 0.1x + 0.007y - 0.13 \\ 0.007x + 0.032y - 0.21 \end{bmatrix}$$

Then, solve the equations $\nabla f_1 = 0$,

$$\begin{cases} 0.1x + 0.007y - 0.13 = 0 \\ 0.007x + 0.032y - 0.21 = 0 \end{cases}$$

The solutions are

$$\begin{cases} x = \dfrac{2690}{3151} \approx 0.8537 \\ y = \dfrac{20090}{3151} \approx 6.3758 \end{cases}$$

Then, the Hessian Matrix are given by

$$H = \begin{bmatrix} \dfrac{\partial^2 f_1}{\partial x^2} & \dfrac{\partial^2 f_1}{\partial x \partial y} \\ \dfrac{\partial^2 f_1}{\partial y \partial x} & \dfrac{\partial^2 f_1}{\partial y^2} \end{bmatrix}$$

$$= \begin{bmatrix} -\dfrac{1}{10} & -\dfrac{7}{1000} \\ -\dfrac{7}{1000} & -\dfrac{4}{125} \end{bmatrix}$$

Apparently,

$$\det(H)=0.003151>0$$

For det(H)>0 and $-\frac{1}{10} < 0$, this point is the maximum point with

$$f_{1\,max} = f_1(0.8537,63758) = 8.6249$$

(b)

The second one is Rosenbrock's function, which is given by

$$f_2(x, y) = 100(y - x^2)^2 + (1 - x)^2$$

As well, plot the image

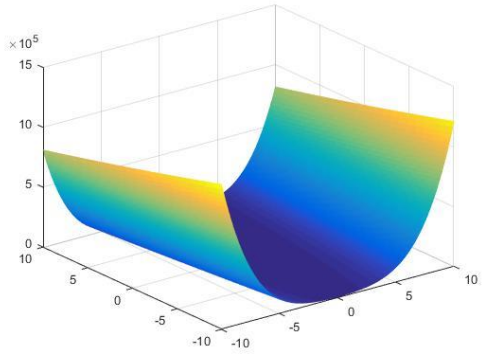

Fig.2

It indicates that the minimum point is around y=0.

Similarly, I compute the Jacobian and Hessian,

$$\nabla f_2 = \begin{bmatrix} 2x - 400x(-x^2 + y) - 2 \\ -200x^2 + 200y \end{bmatrix}$$

$$H = \begin{bmatrix} 1200x^2 - 400y + 2 & -400x \\ -400x & 200 \end{bmatrix}$$

Solving the equations $\nabla f_2 = 0$, I get

$$\begin{cases} x = 1 \\ y = 1 \end{cases}$$

At this point,

$$det(H_2) = 80000x^2 - 80000y + 400$$
$$= 400 > 0$$

And

$$1200x^2 - 400y + 2 = 802 > 0$$

This point is the minimum point with

$$f_{2\,min} = f_2(1,1) = 0$$

2. Direct Search Method

In Matlab, this method can be simply a code.

[x,fval]=fminsearch(f, x0, options)

With this method, it is straightforward to get the results. Note that I apply fminsearch on $-f_1$ to find the maximum point.

The tolerate error is set as 1e-6. The following is what I have

Direct Search Method for f_1

Initial Point	Maximum point	Iterations
(-9,1)	(0.8537,6.3757)	54
(-8,18)	(0.8537,6.3758)	50
(1,19)	(0.8537,6.3757)	46
(9,10)	(0.8537,6.3758)	48

Direct Search Method for f_2

Initial Point	Minimum point	Iterations
(-1,1)	(1.000,1.0000)	54
(0,1)	(1.000,1.0000)	50
(2,1)	(1.000,1.0000)	46

In these cases, direct search method works precisely and quickly.

This method is good to get the maximum point for f_1, since the function is relatively simple. For instance, I tried to input (1e10,1e10), and I got the reasonable answer (which means tol=1e-6) with 119 iterations.

However, the reason why this method did well for f_2 is that these three initial guess points are close to the expected point. If I change the start point to those further from the minimum point, such as (100,100), Matlab shows "Maximum number of function evaluations has been exceeded". In that case, it stops at (2.8104,7.8995).

3. Newton's Method

The iteration scheme of Newton's Method is written as

$$x_{k+1} = x_k - f'(x_k)/\, f''(x_k)$$

The $f'(x_k)$ and $f''(x_k)$ are the Jacobian and the Hessian, respectively.

By iterating, I are able to get the answer.

For f_1, there is only three iterations needed to converge to the expected point for these all three initial points.

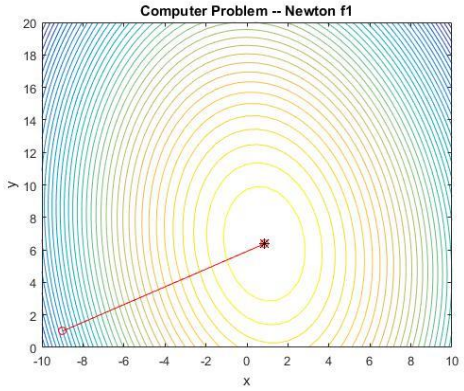

Fig.3 $x_0 = (-9,1)$ Iterations=3

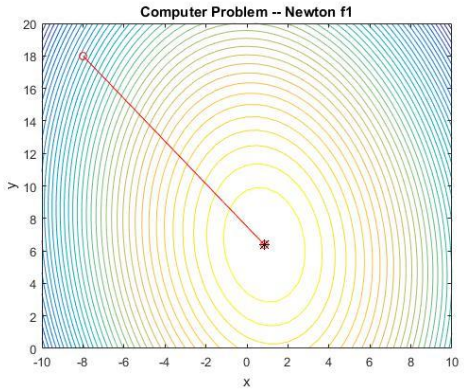

Fig.4 $x_0 = (-8,18)$ Iterations=3

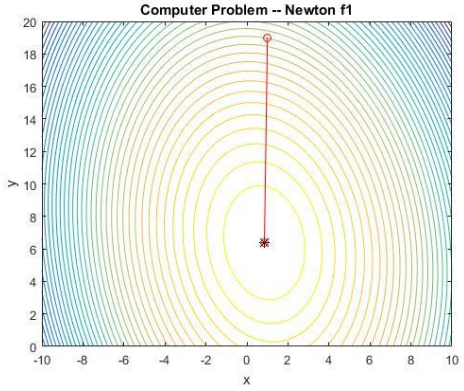

Fig.5 $x_0 = (1,19)$ Iterations=3

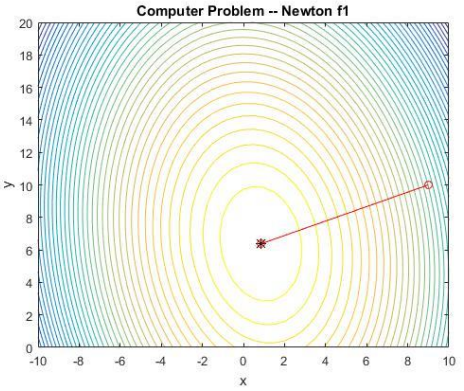

Fig.6 $x_0 = (9,10)$ Iterations=3

Also, the iterations are small when dealing f_2.

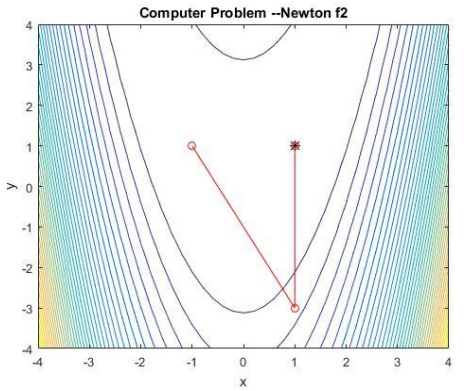

Fig.7 $x_0 = (-1,1)$ Iterations=4

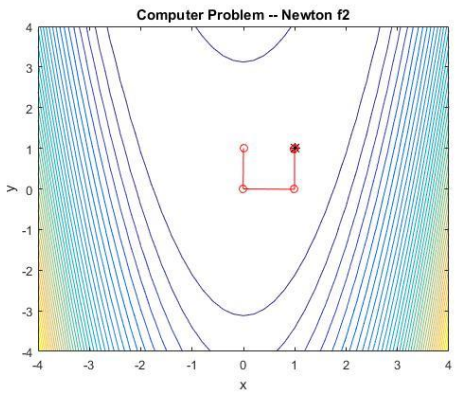

Fig.8 $x_0 = (0,1)$ Iterations=6

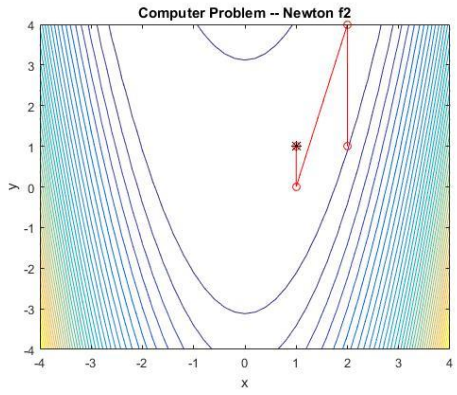

Fig.9 $x_0 = (2,1)$, Iterations=5

Newton's method is the one with least iterations among all methods. It shows a great advantage when tackling the minimum problem.

I believe the reason why it works well is because of the using of second-order partial derivative. Nevertheless, it is the disadvantage of this method, as finding the expression of H matrix and computing the value of H are not always easy as these two functions. The difficulty depends on the case.

4. BFGS method

For this method, a matrix B is used to approximate Hessian matrix, so second order derivatives are not required.

At first, the matrix B_0 is taken by Identity matrix as initial approximation. Next, like newton's method, but replacing the H to B, I have x_{k+1}.

$$x_{k+1} = x_k + s_k, \text{ where } B_k s_k = -J(x_k)$$

Then, update the B matrix by

$$y_k = J(x_{k+1}) - J(x_k)$$

$$B_{k+1} = B_k + (y_k y_k^T)/(y_k^T s_k) - (B_k s_k s_k^T B_k)/(s_k^T B_k s_k)$$

It works well for f_1.

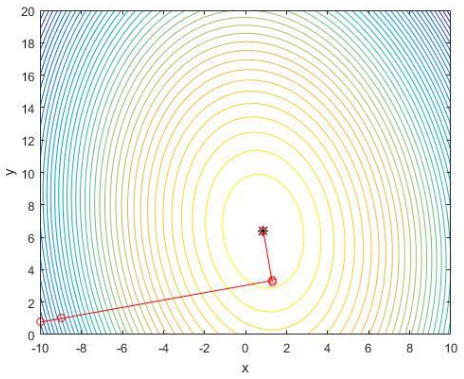

Fig.10 $x_0 = (-9,1)$ Iterations=7

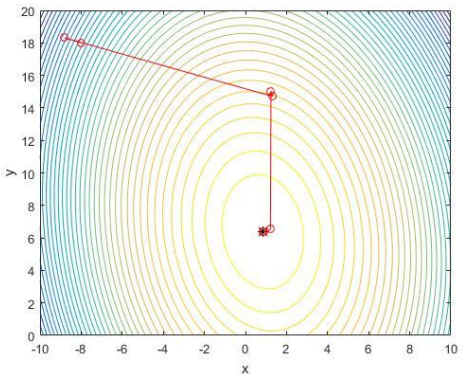

Fig.11 $x_0 = (-8,18)$ Iterations=8

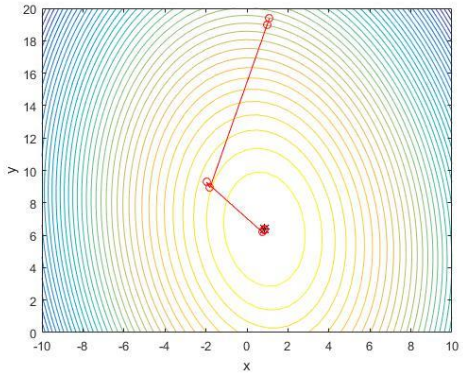

Fig.12 $x_0 = (1,19)$ Iterations=8

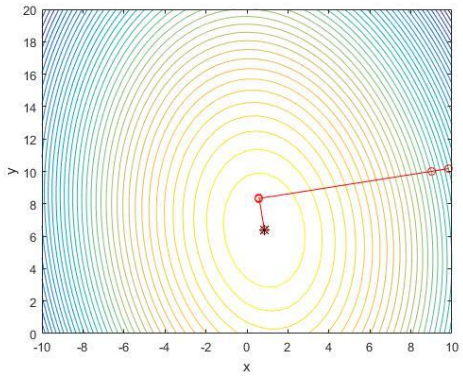

Fig.13 $x_0 = (9,10)$ Iterations=6

Interestingly, those points do not always go toward the direction of function f_1 decreasing.

This situation is much worse in the computing of f_2

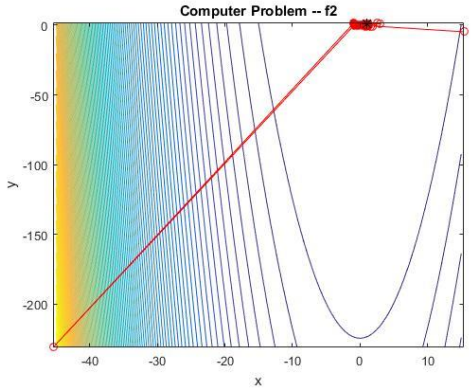

Fig.14 $x_0 = (-1,1)$ Iterations=124

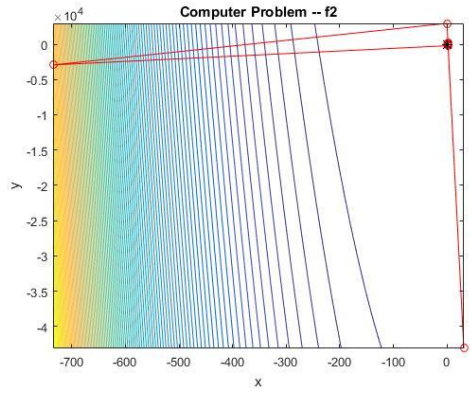

Fig.15 $x_0 = (0,1)$ Iterations=39

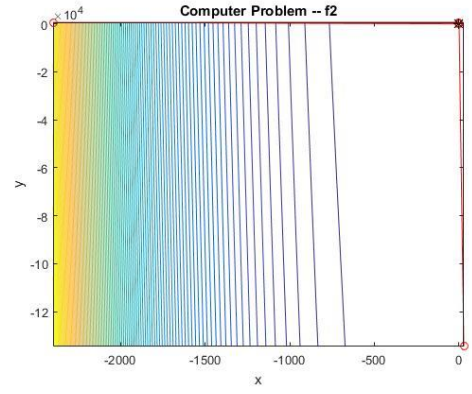

Fig 16 $x_0 = (2,1)$ Iterations=45

As we can see, some points show up in a very far position away from the expected point.

I think it occurs because the difference between B and H. when this difference is great, the updating point goes to a wrong direction.

Additionally, I changed the condition that determines the return value, such that it will not stop at any unexpected points. The previous one is

$$tol < 10^{-6}$$

$$\text{where } tol = |f(x_{k+1}) - f(x_k)|$$

The current one is

$$s_k < 10^{-6}$$

Therefore, the iterations of f_2 are much greater than those of f_1.

5. Steepest Descent Method

Newton's method requires Hessian matrix, and BFGS method provides a way to find the approximate Hessian. Unlike them, Steepest descent method just needs the first order derivatives.

The solution is written as

$$x_{k+1} = x_k + \alpha \nabla f(x_k)$$

Where α is given by a minimization problem.

$$\min(f(x_k + \alpha \nabla f(x_k)))$$

Eventually, we will have the reasonable solution.

For solving the minimization problem, I substitute $x + \alpha \nabla f(\mathrm{x})$ for x, then get a function of α.

If we do so on f_1, we will have a quadratic function $g(\alpha)$. What I do next is taking derivative and finding zero.

```
h=diff(g);
q=solve(h,r);
```

q is the value minimizing $g(\alpha)$.

The answers come out quickly.

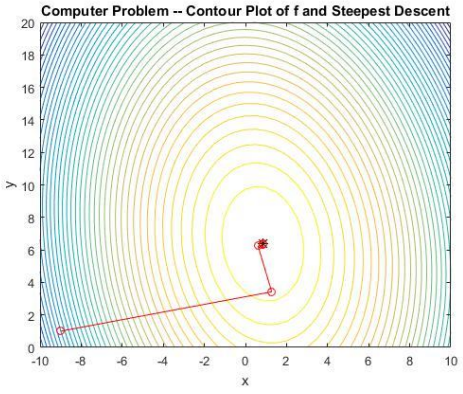

Fig 17. $x_0 = (-9,1)$ Iterations=7

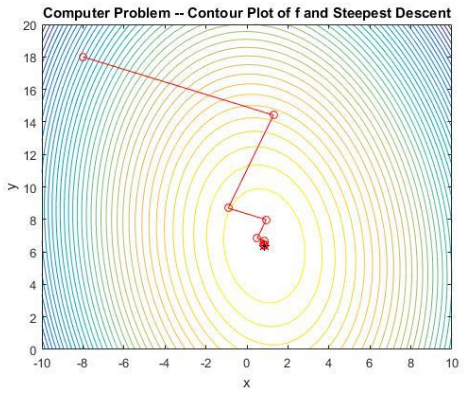

Fig 18 $x_0 = (-8,18)$ Iterations=12

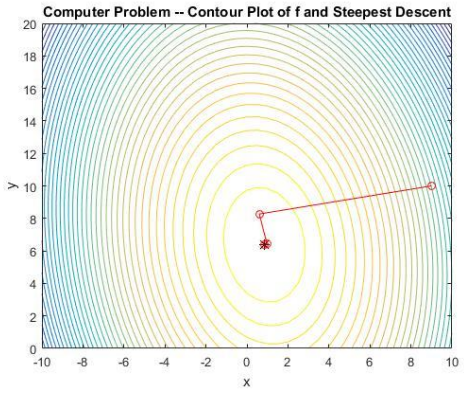

Fig 19. $x_0 = (1,19)$ Iterations=10

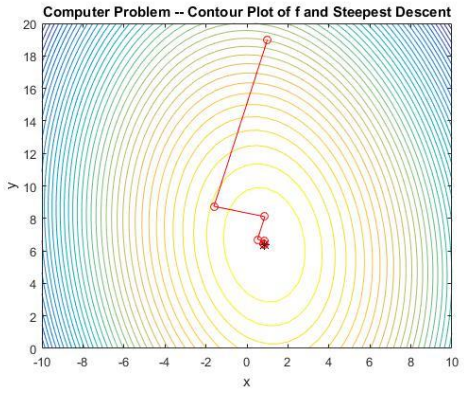

Fig 20. $x_0 = (9,10)$ Iterations=6

For f_2, after substitution, I get a quartic function, so I use matlab built-in function fzero to solve $g'(\alpha) = 0$. The initial guess is set as 0.

gg=matlabFunction(diff(g));
q=fzero(gg,0);

However, the calculation time is really long, which almost took about 5 hours.

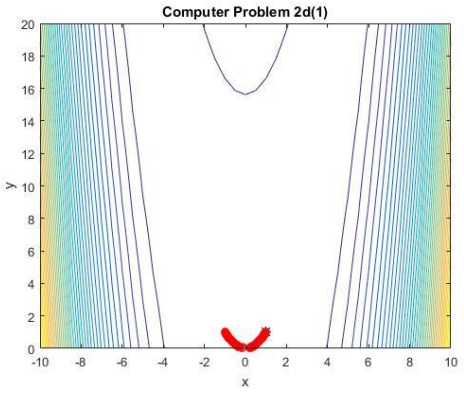

Fig.21 $x_0 = (-1,1)$ Iterations=10021

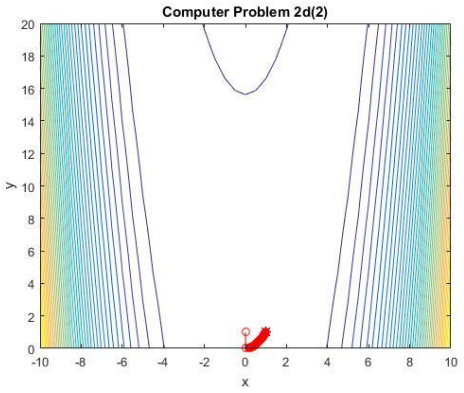

Fig.22 $x_0 = (0,1)$ Iterations=10156

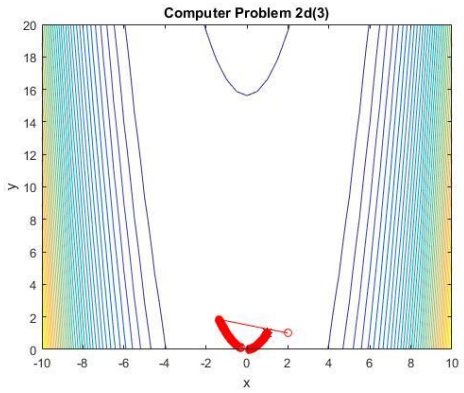

Fig 23 $x_0 = (2,1)$ Iterations=3455

Honestly, I am not sure if I made any mistakes here. If not, I would say that this method is not suitable for f_2.

6. Conjugate Gradient Method

For some particular problem like f_2, Steepest descent method uses too many iterations. To improve it, conjugate gradient method generates sequence of conjugate search directions, implicitly accumulating information about Hessian matrix.

Similarly, the iterative scheme is

$$x_{k+1} = x_k + \alpha s_k$$

Where α is given by

$$\min(f(x_k + \alpha s_k))$$

Then, update s_k based on ∇f,

$$g_{k+1} = \nabla f(x_{k+1})$$

$$\beta_{k+1} = \frac{g_{k+1}^T g_{k+1}}{g_k^T k_k}$$

$$s_{k+1} = g_{k+1} + \beta_{k+1} s_k$$

Then, we will get the convergent solutions.

For f_1, the pictures are pretty much similar as Newton's Method. It shows this method works as well as Newton's in quadratic case.

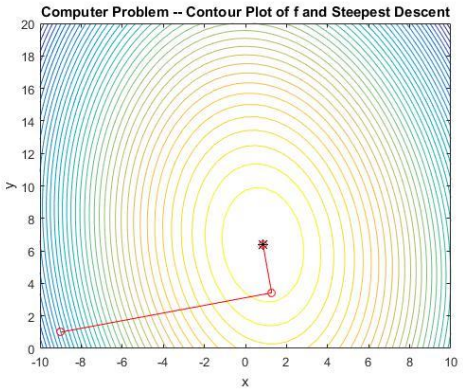

Fig 24. $x_0 = (-9,1)$ Iterations=4

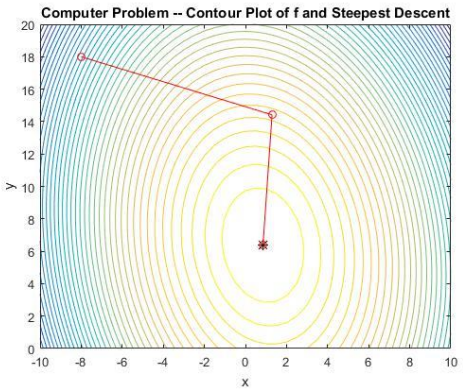

Fig 25. $x_0 = (-8,18)$ Iterations=4

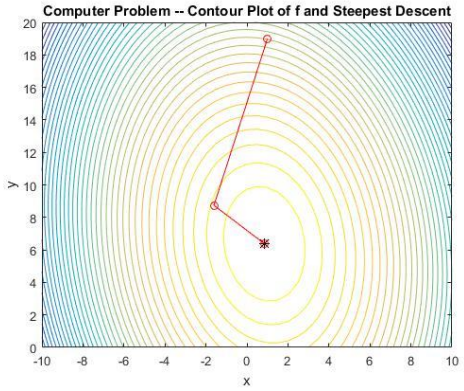

Fig 26. $x_0 = (1,19)$ Iterations=4

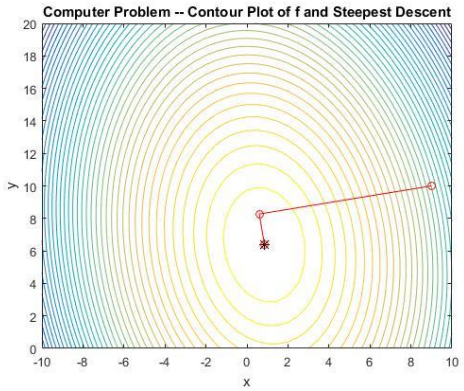

Fig 27. $x_0 = (-9,10)$ Iterations=4

For f_2 , although there are still many iterations used, it is much better than what I have by steepest descent method.

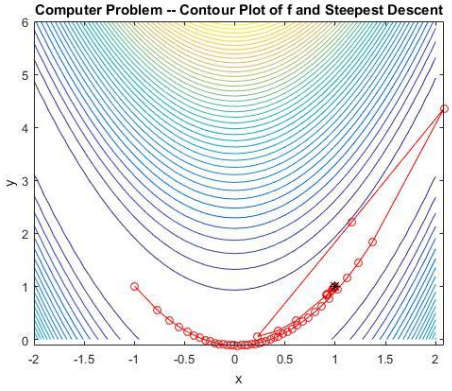

Fig 28. $x_0 = (-1,1)$ Iterations=119

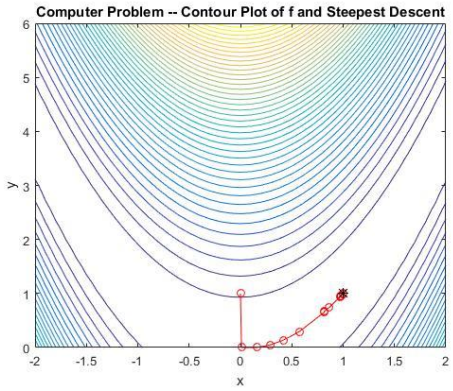

Fig 29. $x_0 = (0,1)$ Iterations=31

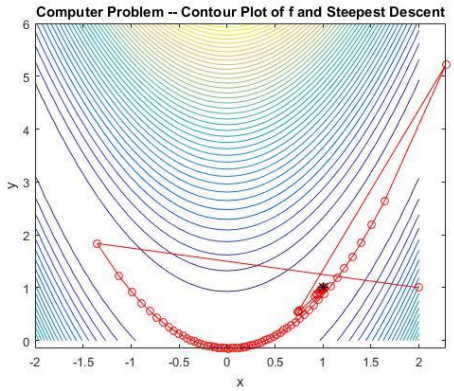

Fig 30. $x_0 = (2,1)$ Iterations=175

Moreover, the change of condition also is also applied for f_2, which is

$$s_k < 10^{-6}$$

7. Nelder-Mead Method

This method is different from all other methods. It is based on geometry. By replacing the point toward smaller value of function, the size of the triangle is

decreased as well. With the triangle gets small enough, the minimum value is found.

For coding, conditional operations are used a lot.

First, I need to rank the initial points.

```
if f(B)>f(W)
    T=B;
    B=W;
    W=T;
  else
    if f(B)>f(G)
      T=G;
      G=B;
      B=T;
    else
      if f(G)>f(W)
        T=G;
        G=W;
        W=T;
      end
    end
  end
```

such that B is for 'Best', G is for 'Good', and W is for 'Worst'.

Then, replace w by a reasonable point.

```
  M=(G+B)/2;
  R=2*M-W;
  E=2*R-M;
if f(R)<f(W)
    W=R;
    if f(E)<f(W)
      W=E;
    end
  else
    C1=(R+M)/2;
    C2=(M+W)/2;
    if f(C1)>f(C2)
      C=C2;
    else
      C=C1;
    end
    if f(C)<f(W)
      W=C;
    else
      W=(B+W)/2;
    end
  end
```

Eventually, the answers are provided.

For f_1, this method works well.

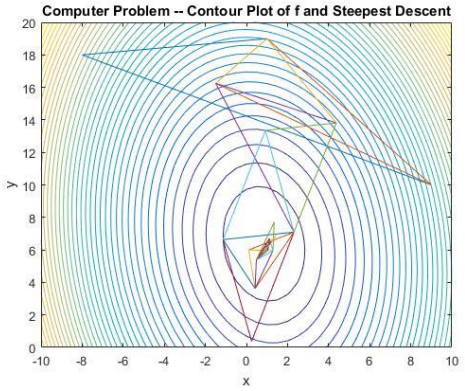

Fig 31. $x_1 = (-8,18),\ x_2 = (1,19),\ x_3 = (9,10)$
Iterations=32

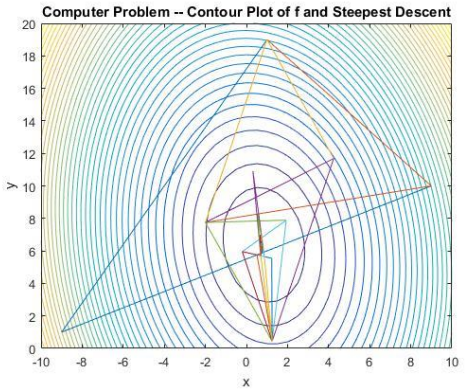

Fig 32. $x_1 = (-9,1),\ x_2 = (1,19),\ x_3 = (9,110)$
Iterations=33

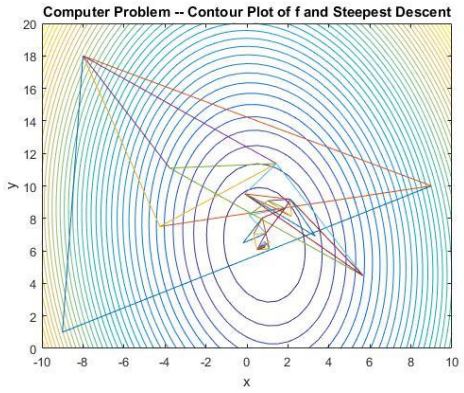

Fig 33. $x_1 = (-9,1),\ x_2 = (-8,18),\ x_3 = (9,10)$
Iterations=33

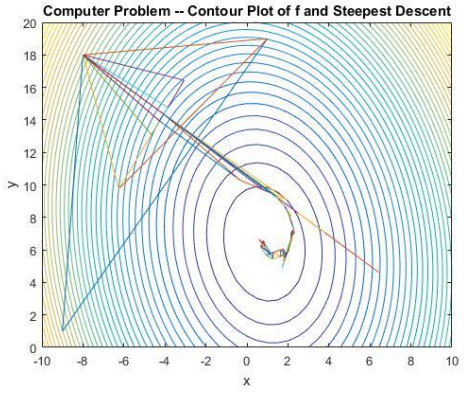

Fig 34. $x_1 = (-9,1),\ x_2 = (-8,18),\ x_3 = (1,19)$
Iterations=114

However, when applying to f_2, what I get is (-0.9950,1.0000) rather than (1,1).

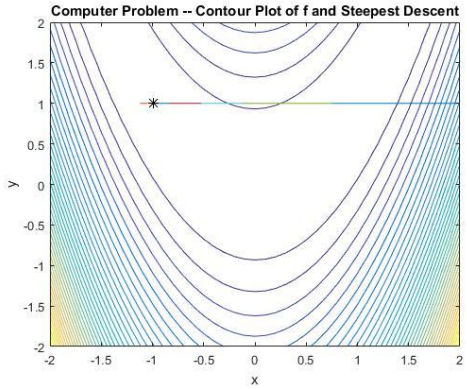

Fig 35. $x_1 = (-1,1),\ x_2 = (0,1),\ x_3 = (2,1)$
Iterations=32

It is because the three initial points are colinear. substitute 1 for y, we have

$$f_2(x) = 100x^4 - 199x^2 - 2x + 101$$

Take derivative,

$$\frac{df_2}{dx} = 400x^3 - 398x - 2$$

Solving $\frac{df_2}{dx} = 0$, I get

$$\begin{cases} x_1 = 1 \\ x_2 = \frac{7\sqrt{2}}{20} - \frac{1}{2} \approx -0.005025 \\ x_3 = -\frac{7\sqrt{2}}{20} - \frac{1}{2} \approx -0.994975 \end{cases}$$

For a better understanding, I plot the picture.

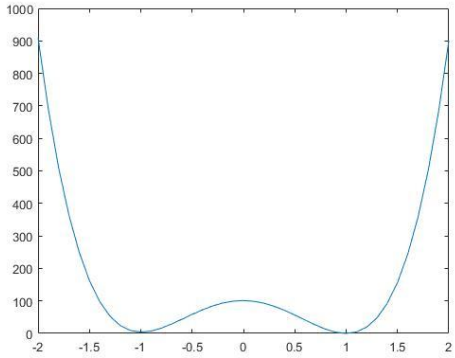

Fig . 36 $f_2(x) = 100x^4 - 199x^2 - 2x + 101$

This is a minimum point but not the least value.

I change the third point to (3,1). The result becomes (1,1), although the three points are still colinear.

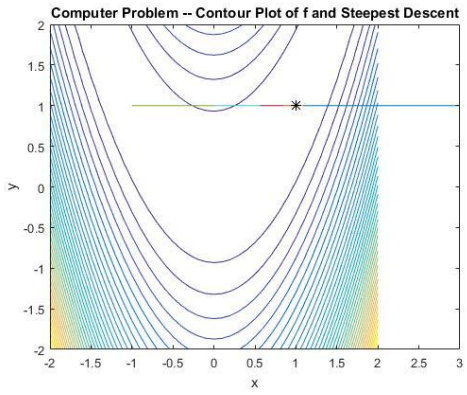

Fig 37. $x_1 = (-1,1),\ x_2 = (0,1),\ x_3 = (3,1)$
Iterations=26

Finally, I change the third point to (2,0), making it a triangle, and, of course, get a good result.

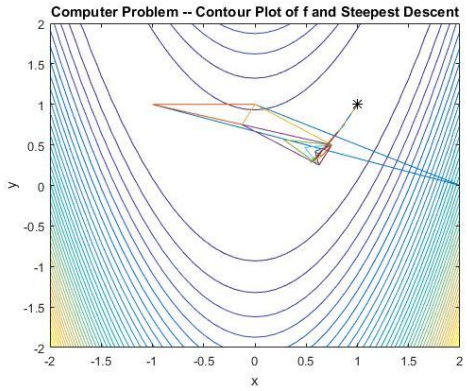

Fig 38. $x_1 = (-1,1),\ x_2 = (0,1),\ x_3 = (2,1)$
Iterations=85

8. Discussion

Taking $\boldsymbol{f_2}$ and initial guess (-1,1) as an example, compare the methods.

f_2 starting from (-1,1)

Method	iterations
Direct search	100
Newton	4
BFGS	124
Steepest descent	10021
Conjugate gradient	119

Among those methods, Newton's method shows a great advantage in searching minimum value, because it is very close to what I mathematically solving this type of problems. Its limitation is H required. Therefore, if I do not have enough information, such as H, conjugate gradient is a good option.

Steepest descent method depends on the function, and it is not good for this case. Although it works well at some other case like $\boldsymbol{f_1}$, it takes too long time to run.

Direct search method depends on the initial points. Fortunately, the given points are fine for this method.

As to Nelder-Mead method, it is effective. Importantly, the starting points are supposed to be colinear.