7.3 Minimisation Methods

2 Steepest Descents

Note: an automated research algorithm is used to minimise $\phi(\lambda)$ for all three minimization methods. The algorithm quits automatically for $|\boldsymbol{x}_n - \boldsymbol{x}_{n-1}| < 10^{-6}$.

Question 1

Figure 1 and figure 2 are surface plots of functions (4) and (5). Suitable axis intervals, $-1.5 \leq x, y \leq 1.5$ are used in both plots.

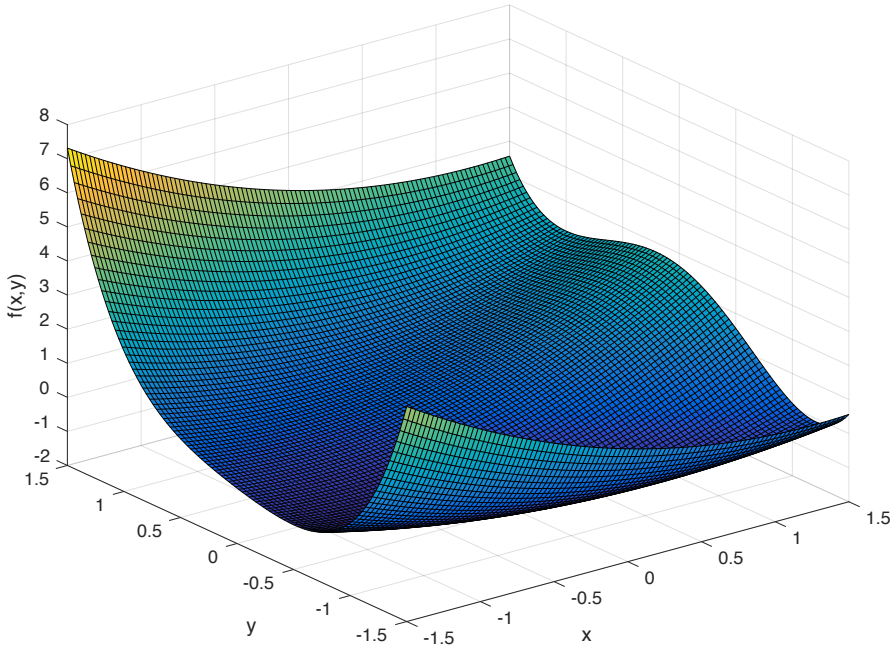

Figure 1: Surface plot of function (4)

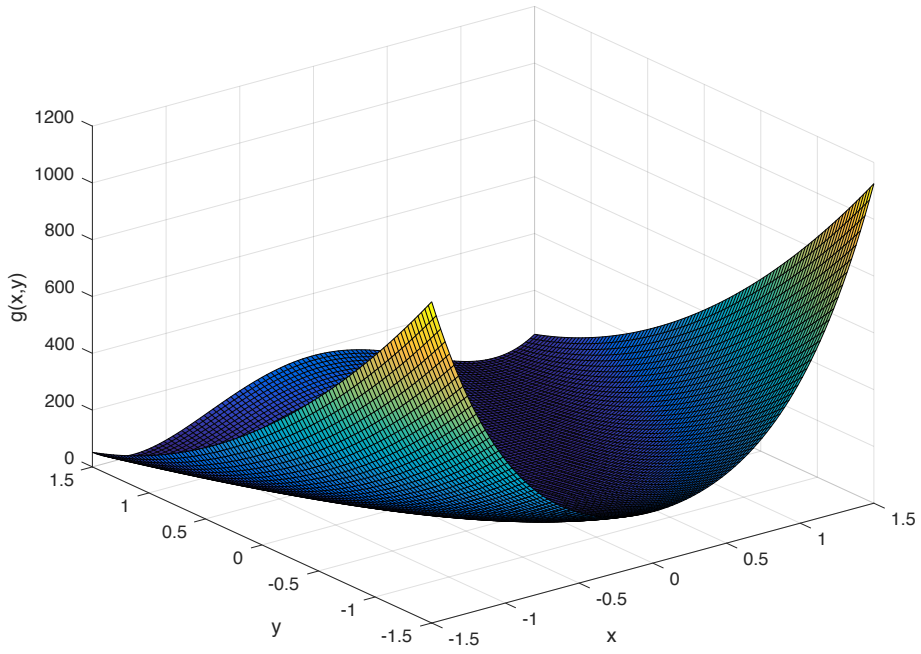

Figure 2: Surface plot of function (5)

The surface plots can help us estimate the minimum values of the functions, however we could work out the values analytically.

Let $f(x,y) = x + y + \frac{x^2}{4} - y^2 + \left(y^2 - \frac{x}{2}\right)^2$, then we have

$$\frac{\partial f}{\partial x} = 1 + x - y^2$$

and

$$\frac{\partial f}{\partial y} = 1 - 2y + 4y(y^2 - \frac{x}{2})$$

By consider $\frac{\partial f}{\partial x} = 0$ and $\frac{\partial f}{\partial y} = 0$, we obtain,

$$x = y^2 - 1 \quad (11)$$

and

$$1 - 2y + 4y\left(y^2 - \frac{x}{2}\right) = 0 \quad (12)$$

Substitute (11) into (12), we have

$$1 - 2y + 4y\left(y^2 - \frac{y^2}{2} + \frac{1}{2}\right) = 0$$

and hence

$$y = -\left(\frac{1}{2}\right)^{\frac{1}{3}} \approx -0.794$$

and

$$x = \left(\frac{1}{2}\right)^{\frac{2}{3}} - 1 \approx -0.370$$

Finally, we obtain the minimum value of function (4),

$$f \approx -1.095$$

Similarly, we define

$$g = (1-x)^2 + 80(y-x^2)^2$$

and consider $\frac{\partial g}{\partial x} = 0$ and $\frac{\partial g}{\partial y} = 0$, we obtain,

$$x - 1 + 160x(y-x^2) = 0 \qquad (13)$$

and

$$y = x^2 \qquad (14)$$

Substitute (14) into (13), we obtain

$$x = 1$$

and hence

$$y = 1$$

Finally, we obtain the minimum value of function (5),

$$g = 0$$

Question 2

Table 1 displays some results from the conjugate gradients algorithm. Where 'd' means the decrease achieved over the last step $d(\boldsymbol{x})$, i.e $d(\boldsymbol{x_n}) = f(\boldsymbol{x_n}) - f(\boldsymbol{x_{n-1}})$.

Table 1:

x	λ^*	f(x)	d(x)
x1	0.0829	-1.0212	-2.0773
x2	1.6471	-1.0921	-0.07094
x3	0.1575	-1.0952	-0.003024
x4	1.7169	-1.0953	-0.00011195
x5	0.1492	-1.0953	-3.4551e-06
x6	1.7191	-1.0953	-1.0612e-07
x7	0.1489	-1.0953	-2.7994e-09
x8	1.7106	-1.0953	-7.4094e-11
x9	0.1491	-1.0953	-1.7897e-12
x10	1.6754	-1.0953	-3.2419e-14

Furthermore, Program 1 also plots all iterations points as a sequence of line segments, where $\boldsymbol{x_n} = (x_n, y_n)$

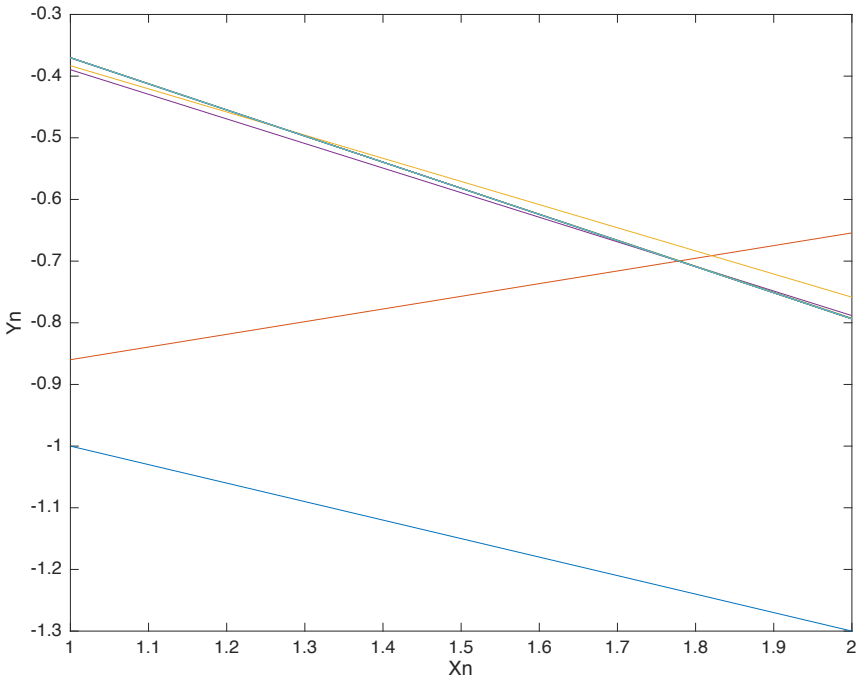

Figure 3: Plot of iteration points

The progress of the iterations can be seen by a scatter diagram of $(n, d(x_n))$, where n presents the n^{th} iteration.

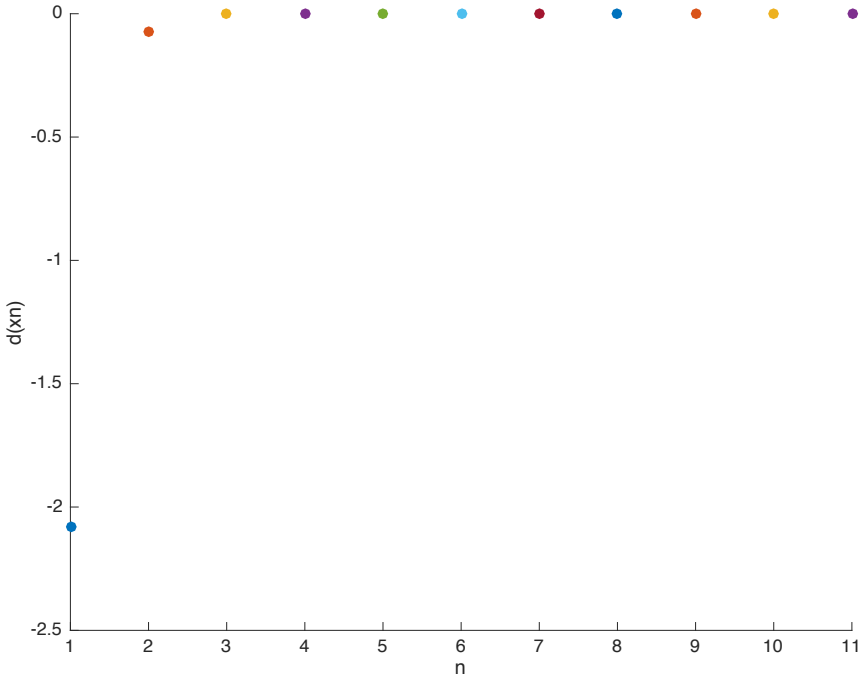

Figure 4: Scatter plot of $(n, d(x_n))$ for function (4)

Base on my numerical results, the estimated minimum value is -1.0953, and it is at $(-0.3700, -0.7937)$. The estimated interval is $-0.3701 < x < -0.03699$ and $-0.7938 < y < -0.7936$.

Table 2

$h = f(\boldsymbol{x_n}) - f(\boldsymbol{x_{n-1}})$	$E_n = f(\boldsymbol{x}^*) - f(\boldsymbol{x_n})$
-2.0773	-0.0741
-0.0706	-0.0035
-0.0033	-1.4287e-04
-1.3814e-04	-4.7241e-06
-4.5693e-06	-1.5478e-07

From table 2, we can deduce that the level on precision is at order 1, i.e. $E_n = O(h)$. (note $\boldsymbol{x}^*$ is the exact point, where $f(x)$ is minimum.

Using the property that, the function is a quadratic, hence it can be written as $f(x) = \frac{1}{2}\boldsymbol{x^T Q x} + \boldsymbol{b^T x}$, where **Q** is a symmetric negative definite matrix and $\boldsymbol{b} \in \boldsymbol{R^2}$, which allows steepest descent to achieve such level of precision.

Question 3

Figure 5 is a plot of the progress of the iteration, where the y-axis is the decrease achieved over the last step.

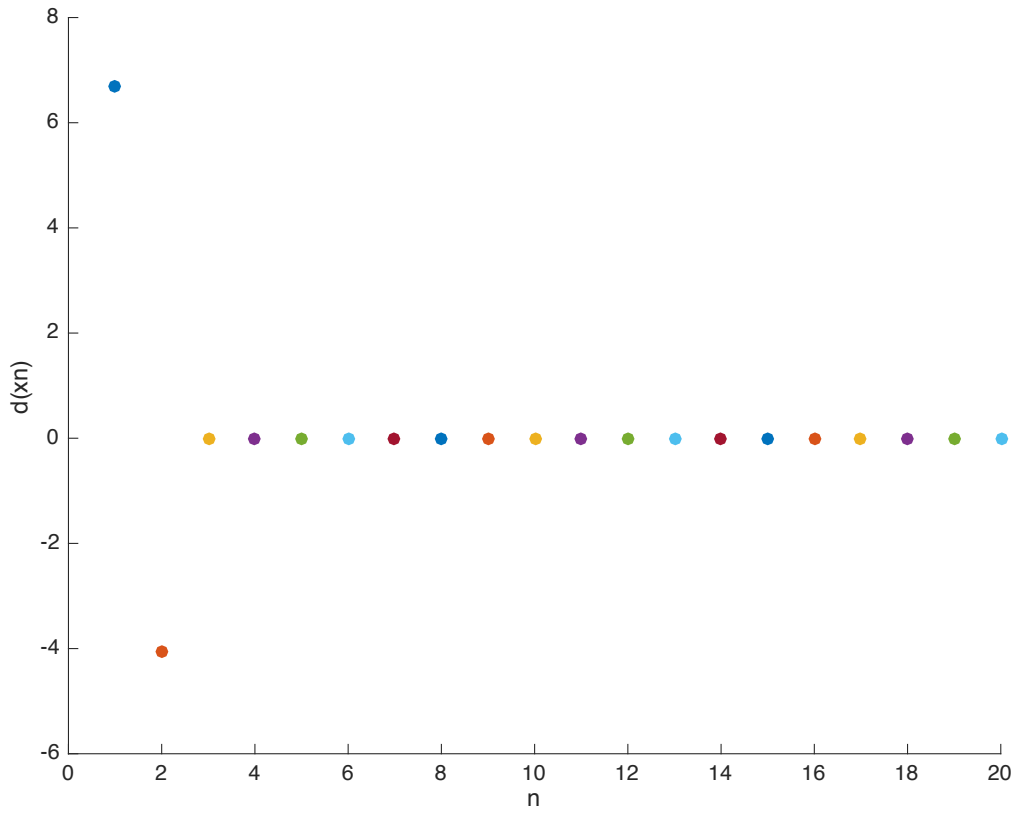

Figure 5

Table 3

n	$d = f(\boldsymbol{x_n}) - f(\boldsymbol{x_{n-1}})$
1	6.6819
2	-4.0561
3	-1.8823e-04
4	-1.8785e-04
5	-1.8782e-04
6	-1.8746e-04

From table 3, the decrease achieved over last step start converging to zero at the 3rd iteration. Hence after the 3rd step, I would think the iteration will eventually converge. From table 3, we can also conclude that the iterations converge logarithmically and sublinearly. Moreover, the rate of convergence is very slow.

Table 4

Original value of d (decrease achieved over last step)	Value of d with perturbation ε	
$\varepsilon = 0$	$\varepsilon = 10^{-6}$	$\varepsilon = 10^{-7}$
6.6819	6.6819	6.6819
-4.0561	-4.0531	-4.0558
-1.8823e-04	18.3342	-1.8899e-04
-1.8785e-04	-7.6564	-1.8844e-04
-1.8782e-04	-9.0400e-05	-1.8844e-04

From table 4, we can see that a perturbation of $O(10^{-6})$ on λ^* changes the value of d dramatically from the 3rd literation onwards. Where as a perturbation of $O(10^{-7})$ on λ^* has a very small affect on d. Hence the iteration path is sensitive to the variations in the choice of λ^* up $O(10^{-6})$.

In case the gradients of functions are nearly orthogonally to the shortest direction to a minimum point, steepest descents method can be inefficient. This means the method is approaching to the minimum point in a 'zig-zag' way. would causes. The Rosenbrock function clearly illustrates this issue.

3 Conjugate Gradients

Question 4

Table 5 displays some results from the conjugate gradients algorithm.

Table 5

n	λ^*	$f(\boldsymbol{x})$	$d = f(\boldsymbol{x_n}) - f(\boldsymbol{x_{n-1}})$ decrease achieved over last step
1	0.0829	-1.0212	2.0773
2	1.5476	-1.0877	0.0665
3	0.1793	-1.0953	0.0076
4	0.2461	-1.0953	2.4510e-08
5	1.0753	-1.0953	6.0388e-08
6	0.1497	-1.0953	1.1857e-12

Figure 9 shows a plot of iteration points $\boldsymbol{x_0}, \boldsymbol{x_1}, \boldsymbol{x_2}$ …

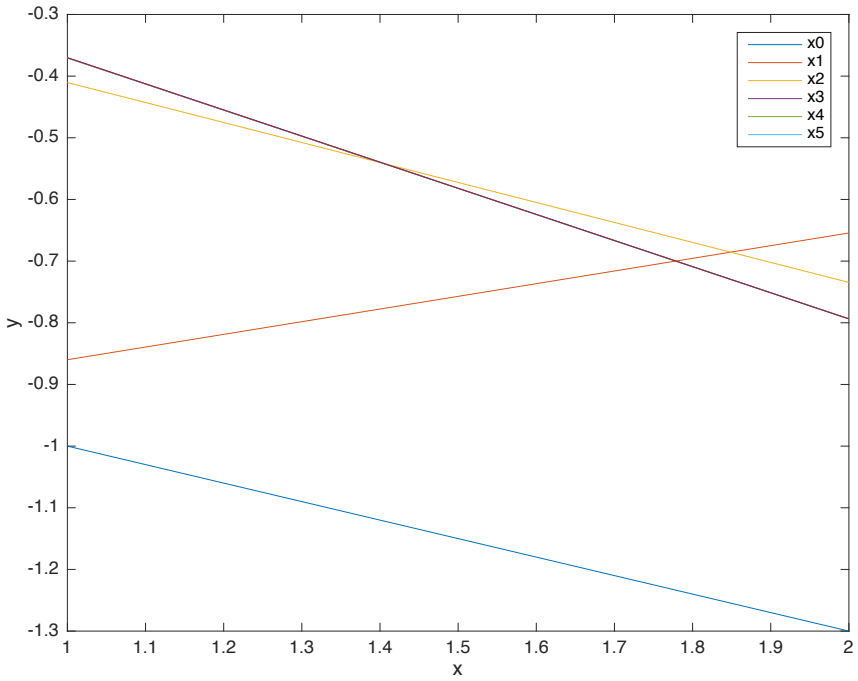

Figure 6: A plot of progress of iteration

Figure 10 is a scatter diagram of the decrease achieved over the last step for each iteration, which shows the progress of the iteration.

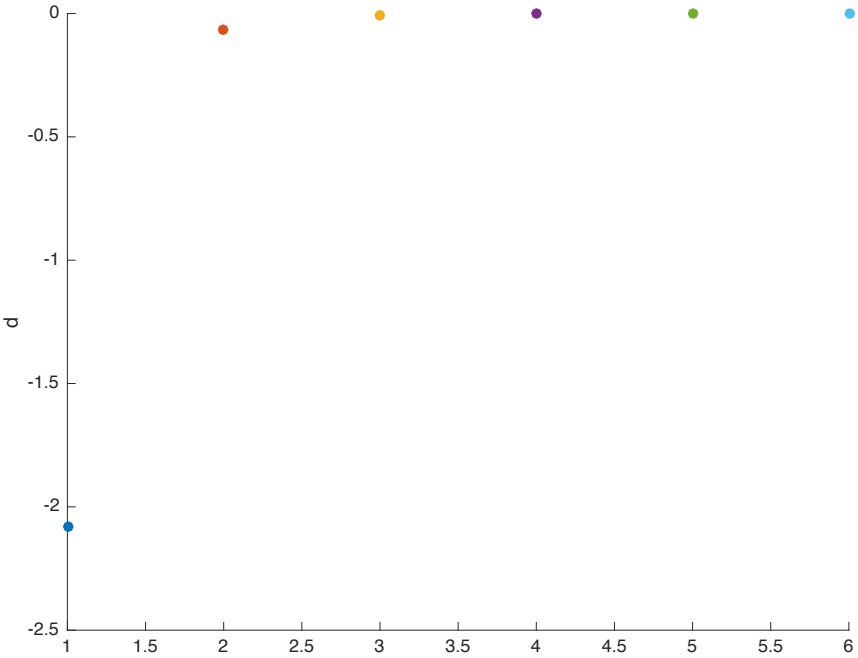

Figure 7

Base on my numerical results, the estimated minimum value is -1.0953, and it is at $(-0.3700, -0.7937)$. The estimated interval is $-0.3701 < x < -0.03699$ and $-0.7938 < y < -0.7936$.

Table 6

$h = f(\boldsymbol{x_n}) - f(\boldsymbol{x_{n-1}})$	$E_n = f(\boldsymbol{x}^*) - f(\boldsymbol{x_n})$
-2.0773	-0.0741
-0.0665	-0.0076
-0.0076	-8.4899e-08
-2.4510e-08	-6.0389e-08

From table 6, we can deduce that the level on precision is at order 1, i.e. $E_n = O(h)$ Using the property that the function is a quadratic, which allows conjugate gradients to achieve such level of precision.

Question 5

Figure 11 is a plot of the progress of iterations. From 56th iteration (d=-4.1402e-06), it will converge eventually.

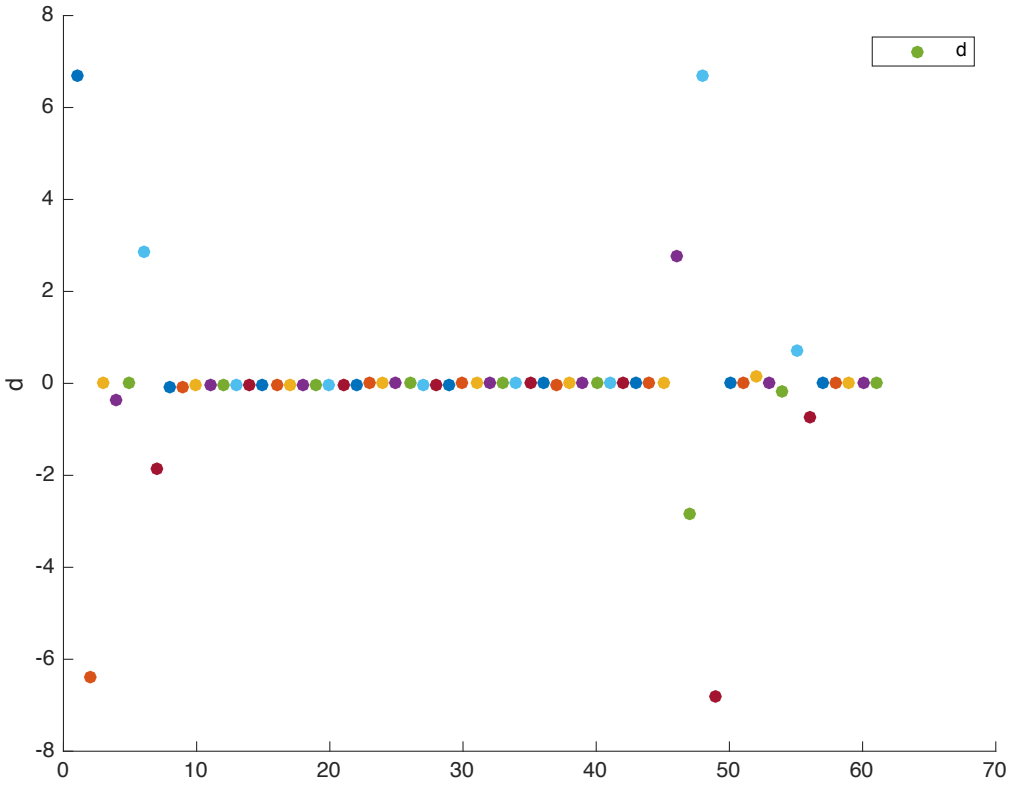

Figure 8

Figures 12 is a plot of iteration points $\boldsymbol{x_0}, \boldsymbol{x_1}, \boldsymbol{x_2}$ …, which demonstrate the sequences converges logarithmically and sublinearly.

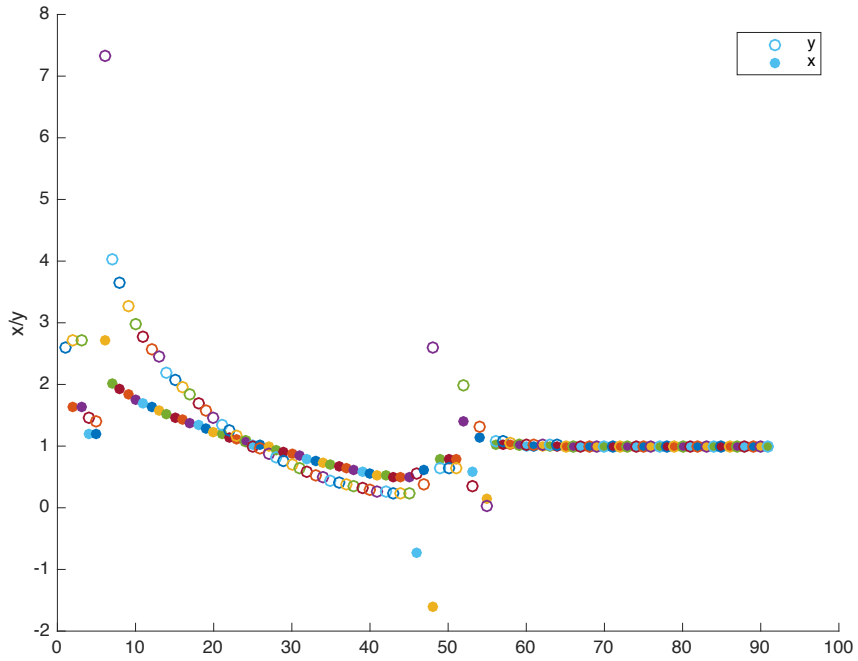

Figure 9

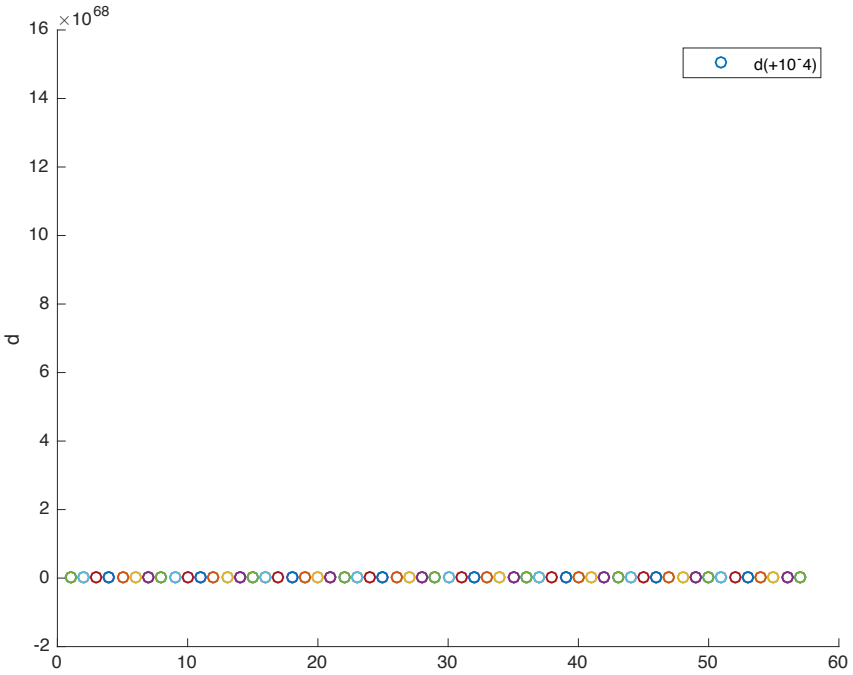

Figure 10

Figure 13 is produced from an adapted program with $\lambda^{*\prime} = \lambda^* + 10^{-4}$, which shows that the iteration path is sensitively depend on the choice λ^*. Whereas figure 13 is produced from an adapted program with $\lambda^{*\prime} = \lambda^* + 10^{-5}$ (with filled dots as the original value of d in comparison), which shoes the the iteration path is not sensitively depend. Hence, we deduce that the path is only sensitively depend on the choice of λ^* up to $O(10^{-4})$.

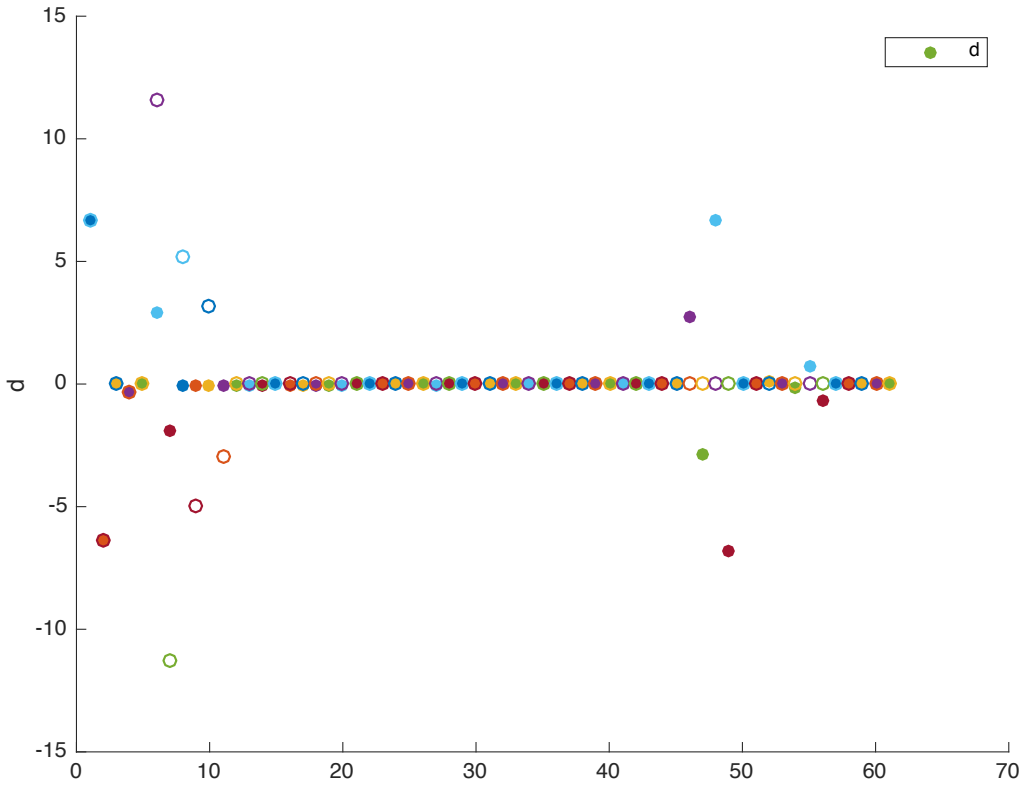

Figure 11

Conjugate gradients generally requires a certain amount of iterations to reach to the minimum. Hence some simple cases, it would be inefficient to use conjugate gradients algorithm. For instance, when the eigenvalues are all equation, it would be more efficient to use the method of steepest descents, which only takes 1 iteration to reach to the minimum.

Compared with steepest descents

Conjugate gradients automatically quits after 6 iterations, due to the iteration points are not changing significantly ($|x_{n+1} - x_n| < 10^{-6}$). Hence it offers an improvement over steepest descents. Furthermore, it over comes the limitations that the steepest descents have (see Question 3).

Question 6

Table 7 shows some results from DFP algorithm implemented on equation (6)

Table 7

n	λ^*	(x, y, z)	$f(x)$	d decrease achieved over last step	H
0		1 1 1	2.6000		$\begin{matrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{matrix}$
1	0.3942	0.2904 0.8423 -0.1826	0.1560	-2.4440	0.8510 0.0142 -0.2603 0.0142 1.0049 0.0232 -0.2603 0.0232 0.5456
2	2.5522	0.1702 -0.0236 -0.1695	0.0116	-0.1444	0.8528 0.1426 -0.2573 0.1426 2.4658 0.0238 -0.2573 0.0238 0.5447
3	4.2203	0.2958 0.0946 -0.2047	0.0181	0.0066	1.6093 0.4360 -0.4810 0.4360 1.8980 -0.0816 -0.4810 -0.0816 0.6103

Table 8 shows that the value of reduction achieved over last step does not fluctuate a lot to a small perturbation to λ^*. Hence the result obtained after three iterations is not sensitively depend to small changes of λ^* up to $O(10^{-2})$

Table 8

Original value of d (decrease achieved over last step)	Value of d with perturbation ε			
$\varepsilon = 0$	$\varepsilon = 10^{-1}$	$\varepsilon = 10^{-2}$	$\varepsilon = 10^{-3}$	$\varepsilon = 10^{-4}$
-2.4440	-2.2868	-2.4425	-2.4440	-2.4440
-0.1444	-0.0764	-0.1371	-0.1436	-0.1443
0.0066	0.0990	0.0065	0.0065	0.0066

Note that $G_{ijk} = \begin{pmatrix} f_{xx} & f_{xy} & f_{xz} \\ f_{yx} & f_{yy} & f_{yz} \\ f_{zx} & f_{zy} & f_{zz} \end{pmatrix} = \begin{pmatrix} 0.8 & 0 & 1 \\ 0 & 0.4 & 0 \\ 1 & 0 & 2 \end{pmatrix}$, and from equation (10), we deduce that

$$H = \begin{pmatrix} 3.3333 & 0 & -1.6667 \\ 0 & 2.5 & 0 \\ -1.6667 & 0 & 1.3333 \end{pmatrix}$$

And from table 7, we can see that H does indeed tend to the inverse Hessian matrix.

Question 7

Table 5 displays some results from the DFP algorithm implemented on equation (4).

Table 9

n	λ^*	$f(x)$	d decrease achieved over last step	H
0	0.0829	1.0561	-2.0773	1 0 0 1
1	1.5502	-1.0212	-0.0666	0.9728 -0.1559 -0.1559 0.1101
2	2.3322	-1.0878	-0.0075	1.5446 -0.3334 -0.3334 0.1491
3	1.0498	-1.0953	-1.7812e-06	1.6117 -0.4304 -0.4304 0.2872
4	0.9077	-1.0953	-4.5242e-10	1.6742 -0.4329 -0.4329 0.2860
5	2.9178	-1.0953	-4.4409e-16	1.6664 -0.4199 -0.4199 0.2646

Figure 15 shows a plot of iteration points $\boldsymbol{x_0}, \boldsymbol{x_1}, \boldsymbol{x_2}$ …

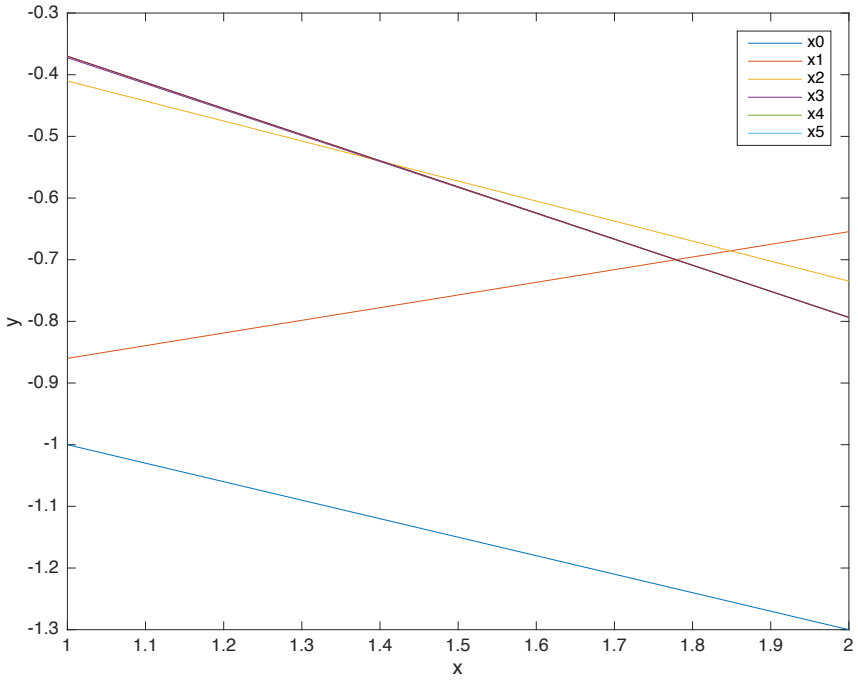

Figure 12

Figure 16 is a scatter diagram of the decrease achieved over the last step for each iteration, which shows the progress of the iteration.

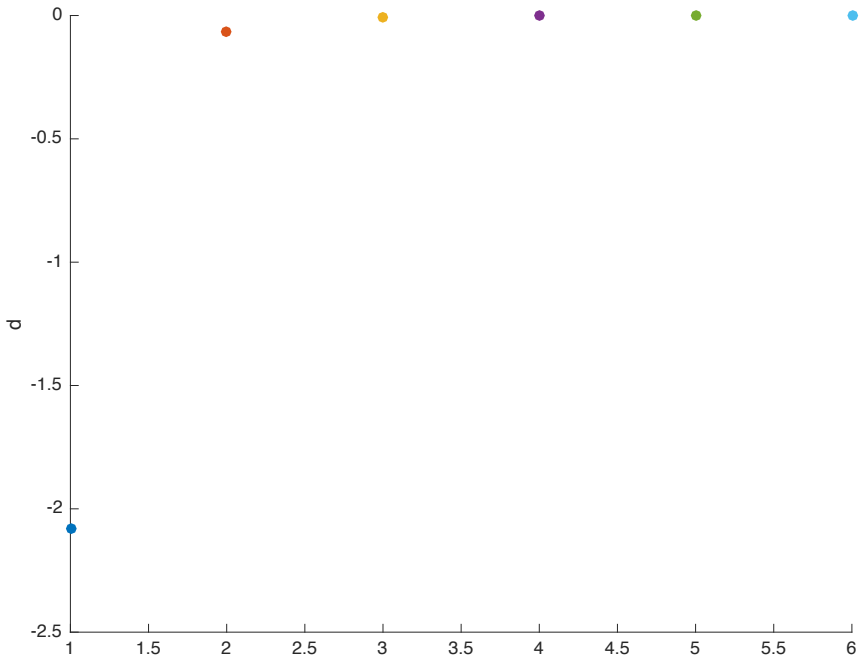

Figure 13

Base on my numerical results, the estimated minimum value is -1.0953, and it is at $(-0.3700, -0.7937)$. The estimated interval is $-0.3701 < x < -0.03699$ and $-0.7938 < y < -0.7936$.

Table 10

$h = f(\boldsymbol{x_n}) - f(\boldsymbol{x_{n-1}})$	$E_n = f(\boldsymbol{x}^*) - f(\boldsymbol{x_n})$
-2.0773	-0.0741
-0.0666	-0.0075
-0.0075	-1.7817e-06
-1.7812e-06	-4.5242e-10

From table 10, we can deduce that the level on precision is at order 2, i.e. $E_n = O(h^2)$ Using the property that the function is a quadratic, and it is twice differentiable, which allows DFP algorithm to achieve such level of precision.

The true value of inverse of Hessian the matrix is $\begin{pmatrix} 1.6667 & -0.4200 \\ -0.4200 & 0.2646 \end{pmatrix}$, obtained from Program 'Question7ii'. Hence, from table 7, we deduce that H is converging to the inverse of the Hessian matrix.

Question 8

Figure 17 is a plot of the progress of the iteration, where the y-axis is the decrease achieved over the last step.

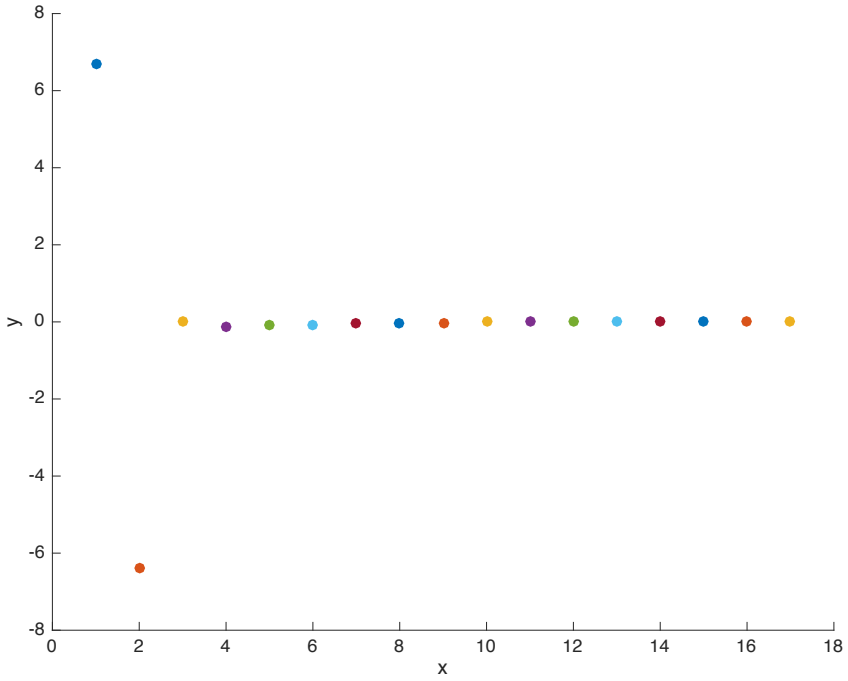

Figure 14

Table 11

n	d	H	
1	6.6819	1.5106	-0.4435
		-0.4435	0.4554
2	-6.3905	-1.1000	-0.4571
		-0.4571	0.4527
3	-7.2713e-06	0.0844	0.2757
		0.2757	0.9068
4	-0.1359	0.1494	0.4789
		0.4789	1.5389
5	-0.0739	0.0998	0.2906
		0.2906	0.8478
6	-0.0840	0.0545	0.1431
		0.1431	0.3816
7	-0.0478	0.1733	0.4654
		0.4654	1.2527
8	-0.0210	0.1467	0.3612
		0.3612	0.8909
9	-0.0301	0.0874	0.1943
		0.1943	0.4371
10	-0.0115	0.2592	0.5923
		0.5923	1.3568

11	-0.0049	0.2743	0.5784
		0.5784	1.2224
12	-0.0024	0.2157	0.4412
		0.4412	0.9085
13	-5.3508e-04	0.4787	0.9720
		0.9720	1.9788
14	-2.8185e-05	0.5149	1.0325
		1.0325	2.0769
15	-7.4761e-07	0.5002	1.0012
		1.0012	2.0098

From table 11, the value of reduction achieved over last step start converging to zero at the 10^{th} iteration. Hence after the 10^{th} step, the iteration will eventually converge.
From figure 14, we conclude that the iterations converge logarithmically and sublinearly.

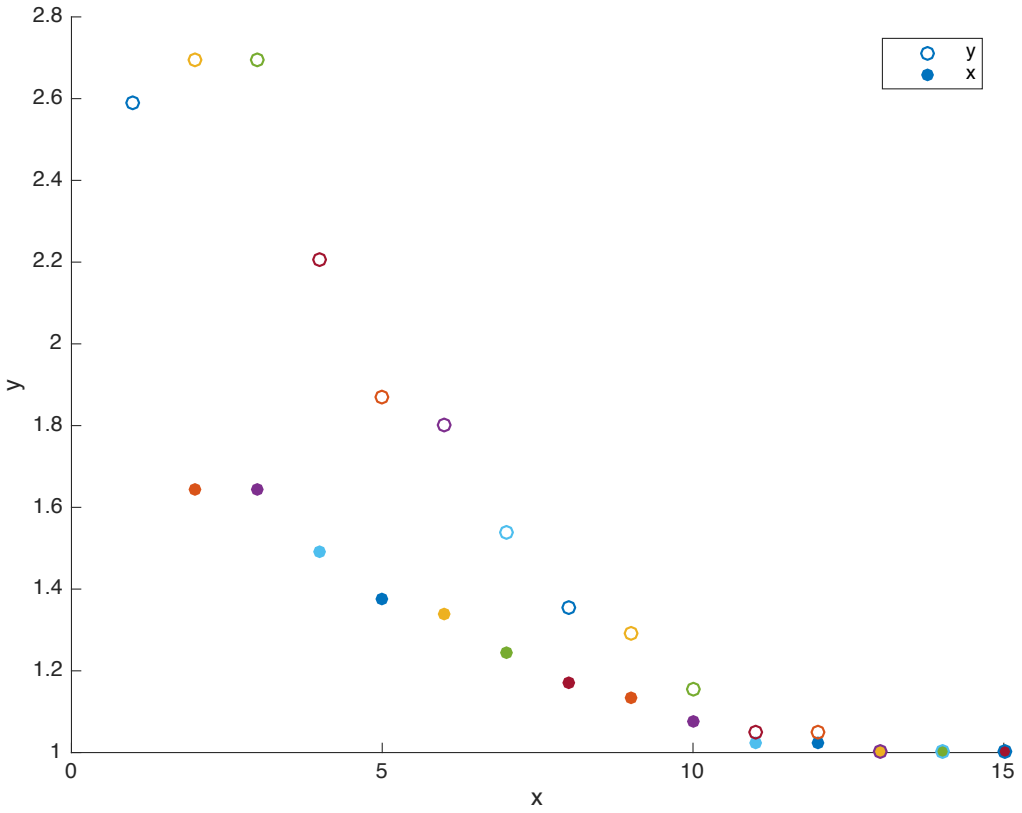

Figure 15

Table 12

Original value of d (decrease achieved over last step)	Value of d with perturbation ε	
$\varepsilon = 0$	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-5}$
6.6819	6.6820	6.6819
-6.3905	-6.2363	-6.3770
-7.2713e-06	-0.5592	-5.8841e-05

-0.1359	5.0813	-0.1380
-0.0739	-4.8207	-0.0783

From table 12, we see that a perturbation of $O(10^{-4})$ on λ^* changes the value of d dramatically from the 3rd iteration onwards. Where as a perturbation of $O(10^{-5})$ on λ^* has a very small affect on d. Hence the iteration path is sensitive to the variations in the choice of λ^* up $O(10^{-4})$.

The DFP algorithm requires the second derivatives, which can be expensive for some complex functions, especially the ones that are difficult to differentiate.

The true value of inverse of Hessian the matrix is $\begin{pmatrix} 0.5000 & 1.0000 \\ 1.0000 & 2.0062 \end{pmatrix}$, obtained from Program 'Question8ii'.Also, from table 11, we can see that H is converging to the true value.

Question 9

All three algorithms are considered to be efficient for equation (4). Where steepest descents automatically quit at the 10th iteration; and both conjugate gradients and DFP algorithm 6 quits at the 6th iteration.

For equation (5), the method of steepest descents becomes the least efficient. DFP algorithm is more efficient than conjugate gradient, which converges after 10 iterations compared with 56 iterations for conjugate gradient.

Equation (6) has 3 variables, which maybe cause difficulty to find a suitable initial choice of $\boldsymbol{x_0}$**.** Moreover, $\phi(\lambda)$ would be more complex and the search of a precise λ^* may be more costly. From Question 6, we demonstrated that DFP algorithm works for any initial choice of $\boldsymbol{x_0}$ and results are not sensitive to small changes in λ^*. Hence DFP is more efficient than steepest descents and conjugate gradients.

References

1. https://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf
2. https://www.daletronrud.com/crystallography/papers/CD/node5.html

Appendix

Programs for Question 1

```
%surface plot of equation (4)
n=100;
x=linspace(-1.5,1.5,n);
y=linspace(-1.5,1.5,n);
[X,Y]=meshgrid(x,y);
F=X+Y+X.^2/4-Y.^2+(Y.^2-X/2).^2;
surf(X,Y,F);
xlabel('x')
ylabel('y')
zlabel('f(x,y)')

%surface plot of equation (5)
n=100;
x=linspace(-1.5,1.5,n);
y=linspace(-1.5,1.5,n);
[X,Y]=meshgrid(x,y);
G=(1-X).^2+80*(Y-X.^2).^2;
surf(X,Y,G)
xlabel('x')
ylabel('y')
zlabel('g(x,y)')
```

Programs for Question 2

```
%Q2-first part
n=0;
x0=-1.0;
y0=-1.3;
df1=@(x,y)1+x-y^2;
df2=@(x,y)1-2*y+4*y*(y^2-x/2);
g1=df1(x0,y0);
g2=df2(x0,y0);
f=@(x,y)x+y+x^2/4-y^2+(y^2-x/2)^2;
fstar=f((1/2)^(2/3)-1,-(1/2)^(1/3));

while n<6

phi=@(k)(x0-k*g1)+(y0-k*g2)+((x0-k*g1)^2)/4-(y0-k*g2)^2+((y0-k*g2)^2-(x0-
k*g1)/2)^2;
kstar=fminbnd(phi,-10,10);
x1=x0-kstar*g1;
y1=y0-kstar*g2;
%d=decrease achieved over the last step
d=f(x1,y1)-f(x0,y0);
h=d
E=fstar-f(x1,y1)
scatter(n+1,d,'filled')
xlabel('n')
ylabel('d(xn)')
hold on
if norm([x1 y1]-[x0 y0])<10^-6
    n=1200;
end
x0=x1;
y0=y1;
g1=df1(x0,y0);
g2=df2(x0,y0);
n=n+1;
```

```
    end
hold off
```

Programs for Question 3

```
%Q3-first part
n=0;
x0=0.676;
y0=0.443;

df1=@(x,y)-2*(1-x)-320*x*(y-x^2);
df2=@(x,y)160*(y-x^2);
g1=df1(x0,y0);
g2=df2(x0,y0);
f=@(x,y)(1-x)^2+80*((y-x^2)^2);

while n<5


phi=@(k)(1-(x0-k*g1))^2+80*(((y0-k*g2)-(x0-k*g1)^2)^2);
kstar=fminbnd(phi,-10,10)+10^-7;
x1=x0-kstar*g1;
y1=y0-kstar*g2;
%d=decrease achieved over the last step
d=f(x1,y1)-f(x0,y0)
scatter(n+1,x1,'filled','r')
%scatter(n+1,y1)
xlabel('n')
ylabel('d(xn)')
hold on
%disp([x0,y0]);
if norm([x1 y1]-[x0 y0])<10^-6
    n=2500;
end
x0=x1;
y0=y1;
g1=df1(x0,y0);
g2=df2(x0,y0);
n=n+1;
    end
```

Programs for Question 4

```
n=0;
x0=-1.0;
y0=-1.3;
plot([x0,y0])
hold on
fx=@(x,y)1+x-y^2;
fy=@(x,y)1-2*y+4*y*(y^2-x/2);
fx0=fx(x0,y0);
fy0=fy(x0,y0);
f=@(x,y)x+y+x^2/4-y^2+(y^2-x/2)^2;
%scatter(x0,y0,'filled')
phi=@(k)(x0-k*fx0)+(y0-k*fy0)+(x0-k*fx0)^2/4-(y0-k*fy0)^2+((y0-k*fy0)^2-
(x0-k*fx0)/2)^2;
kstar=fminbnd(phi,-1,1)
x1=x0-kstar*fx0;
y1=y0-kstar*fy0;
F=f(x1,y1)
d=f(x1,y1)-f(x0,y0)
```

```
plot([x1,y1])
%scatter(x1,y1,'filled')
hold on
fx1=fx(x1,y1);
fy1=fy(x1,y1);
sx0=-fx0;
sy0=-fy0;

while n<11

    beta1=(fx1^2+fy1^2)/(fx0^2+fy0^2);
    sx1=-fx1+beta1*sx0;
    sy1=-fy1+beta1*sy0;
    phi=@(k)(x1+k*sx1)+(y1+k*sy1)+(x1+k*sx1)^2/4-
(y1+k*sy1)^2+((y1+k*sy1)^2-(x1+k*sx1)/2)^2;
    kstar=fminbnd(phi,-10^10,10^10)
    x2=x1+kstar*sx1;
    y2=y1+kstar*sy1;
    fx2=fx(x2,y2);
    fy2=fy(x2,y2);
    F=f(x2,y2)
    d=f(x2,y2)-f(x1,y1)
    plot([x2,y2])
    %scatter(x2,y2,'filled')
   hold on
    x0=x1;
    y0=y1;
    fx0=fx1;
    fy0=fy1;
    sx0=sx1;
    sy0=sy1;
    x1=x2;
    y1=y2;
    fx1=fx2;
    fy1=fy2;
    n=n+1;

if norm([x1 y1]-[x0 y0])<10^-5
    n=12;
end

end
xlabel('x')
ylabel('y')
legend('x0','x1','x2','x3','x4','x5')
hold off
```

Programs for Question 5

```
n=0;
x0=0.676;
y0=0.443;
fx=@(x,y)-2*(1-x)-320*x*(y-x^2);
fy=@(x,y)160*(y-x^2);
fx0=fx(x0,y0);
fy0=fy(x0,y0);
f=@(x,y)(1-x)^2+80*((y-x^2)^2);
phi=@(k)(1-(x0-k*fx0))^2+80*(((y0-k*fy0)-(x0-k*fx0)^2)^2);
kstar=fminbnd(phi,-10,10);
x1=x0-kstar*fx0;
y1=y0-kstar*fy0;
d=f(x1,y1)-f(x0,y0)
```

```
scatter(n+1,d,'filled')
%scatter(n+1,y1)
hold on
fx1=fx(x1,y1);
fy1=fy(x1,y1);
sx0=-fx0;
sy0=-fy0;

while n<80

    beta1=(fx1^2+fy1^2)/(fx0^2+fy0^2);
    sx1=-fx1+beta1*sx0;
    sy1=-fy1+beta1*sy0;
    phi=@(k)(1-(x1+k*sx1))^2+80*(((y1+k*sy1)-(x1+k*sx1)^2)^2);
    kstar=fminbnd(phi,-10,10);
    x2=x1+kstar*sx1;
    y2=y1+kstar*sy1;
    fx2=fx(x2,y2);
    fy2=fy(x2,y2);
    F=f(x2,y2);
    d=f(x2,y2)-f(x1,y1)
    scatter(n+2,d,'filled')
    %scatter(n+2,y2)
   hold on
    x0=x1;
    y0=y1;
    fx0=fx1;
    fy0=fy1;
    sx0=sx1;
    sy0=sy1;
    x1=x2;
    y1=y2;
    fx1=fx2;
    fy1=fy2;
    n=n+1;

if norm([x1 y1]-[x0 y0])<10^-8
    n=2500;
end

end
ylabel('d')
legend('d')
```

Programs for Question 6

```
n=1;
x0=zeros(3,1);
g0=zeros(3,1);
s0=zeros(3,1);
H0=eye(3)
X1=zeros(3,1);
X2=zeros(3,1);
G1=zeros(3,1);
S1=zeros(3,1);
H1=zeros(3,3);
p1=zeros(3,1);
q1=zeros(3,1);
x0(1)=1;
x0(2)=1;
x0(3)=1;
```

```
x0
scatter3(x0(1),x0(2),x0(3),'filled')
hold on
fx=@(x,y,z)0.8*x+z;
fy=@(x,y,z)0.4*y;
fz=@(x,y,z)2*z+x;
g0(1)=fx(x0(1),x0(2),x0(3));
g0(2)=fy(x0(1),x0(2),x0(3));
g0(3)=fz(x0(1),x0(2),x0(3));
f=@(x,y,z)0.4*x^2+0.2*y^2+z^2+x*z;
F=f(x0(1),x0(2),x0(3))
s0=-H0*g0;

kstar=[0.3942 2.5522 4.2203]+0*10^-4*[1 1 1];
X1=x0+kstar(1)*s0
F=f(X1(1),X1(2),X1(3));
d=f(X1(1),X1(2),X1(3))-f(x0(1),x0(2),x0(3));
%scatter(X1,'b','filled')
scatter3(X1(1),X1(2),X1(3),'filled')

while n<3

G1(1)=fx(X1(1),X1(2),X1(3));
G1(2)=fy(X1(1),X1(2),X1(3));
p1=G1-g0;
q1=X1-x0;
H1=H0-(H0*p1*p1.'*H0)/(p1.'*H0*p1)+(q1*q1.')/(p1.'*q1)
S1=-H1*G1;
X2=X1+kstar(n+1).*S1
F=f(X2(1),X2(2),X2(3));
d=f(X2(1),X2(2),X2(3))-f(X1(1),X1(2),X1(3));
%scatter(n+1,d,'b','filled')
scatter3(X2(1),X2(2),X2(3),'filled')
x0=X1;
g0=G1;
X1=X2;
H0=H1;
n=n+1;

if norm(X1-x0)<10^-6
    n=12;
end
end
G1(1)=fx(X1(1),X1(2),X1(3));
G1(2)=fy(X1(1),X1(2),X1(3));
p1=G1-g0;
q1=X1-x0;
H1=H0-(H0*p1*p1.'*H0)/(p1.'*H0*p1)+(q1*q1.')/(p1.'*q1)

xlabel('x')
ylabel('y')
zlabel('z')
hold off
```

Programs for Question 7

```
n=1;
x0=zeros(2,1);
g0=zeros(2,1);
s0=zeros(2,1);
H0=eye(2);
X1=zeros(2,1);
```

```
X2=zeros(2,1);
G1=zeros(2,1);
S1=zeros(2,1);
H1=zeros(2,2);
p1=zeros(2,1);
q1=zeros(2,1);
x0(1)=-1.0;
x0(2)=-1.3;
fx=@(x,y)1+x-y^2;
fy=@(x,y)1-2*y+4*y*(y^2-x/2);
g0(1)=fx(x0(1),x0(2));
g0(2)=fy(x0(1),x0(2));
f=@(x,y)x+y+x^2/4-y^2+(y^2-x/2)^2;
fstar=f((1/2)^(2/3)-1,-(1/2)^(1/3));
F=f(x0(1),x0(2));
s0=-H0*g0;

%plot(x0)
%hold on
phi=@(k)(x0(1)+k*s0(1))+(x0(2)+k*s0(2))+(x0(1)+k*s0(1))^2/4-
(x0(2)+k*s0(2))^2+((x0(2)+k*s0(2))^2-(x0(1)+k*s0(1))/2)^2;
kstar=fminbnd(phi,-10,10);
X1=x0+kstar*s0;
F=f(X1(1),X1(2));
d=f(X1(1),X1(2))-f(x0(1),x0(2))
E=fstar-f(X1(1),X1(2))
%scatter(n+1,d,'filled')
%plot(X1)
%hold on

while n<19

G1(1)=fx(X1(1),X1(2));
G1(2)=fy(X1(1),X1(2));
p1=G1-g0;
q1=X1-x0;
H1=H0-(H0*p1*p1.'*H0)/(p1.'*H0*p1)+(q1*q1.')/(p1.'*q1);
S1=-H1*G1;
phi=@(k)(X1(1)+k*S1(1))+(X1(2)+k*S1(2))+(X1(1)+k*S1(1))^2/4-
(X1(2)+k*S1(2))^2+((X1(2)+k*S1(2))^2-(X1(1)+k*S1(1))/2)^2;
kstar=fminbnd(phi,-10,10);
X2=X1+kstar.*S1;
F=f(X2(1),X2(2));
d=f(X2(1),X2(2))-f(X1(1),X1(2))
E=fstar-f(X2(1),X2(2))
scatter(n+1,d)
hold on
x0=X1;
g0=G1;
X1=X2;
H0=H1;
n=n+1;

if norm(X1-x0)<10^-6
    n=220;
end
end
xlabel('x')
ylabel('y')
%legend('x0','x1','x2','x3','x4','x5')
hold off
```

```
G=zeros(2,2);
G(1,1)=1;
x=(1/2)^(2/3)-1;
y=-(1/2)^(1/3);
G(1,2)=-2*y;
G(2,1)=-2*y;
G(2,2)=-2+4*(y^2-x/2)+8*y^2;
H=inv(G)
```

Programs for Question 8

```
n=1;
x0=zeros(2,1);
g0=zeros(2,1);
s0=zeros(2,1);
H0=eye(2);
X1=zeros(2,1);
X2=zeros(2,1);
G1=zeros(2,1);
S1=zeros(2,1);
H1=zeros(2,2);
p1=zeros(2,1);
q1=zeros(2,1);
x0(1)=0.676;
x0(2)=0.443;
fx=@(x,y)-2*(1-x)-320*x*(y-x^2);
fy=@(x,y)160*(y-x^2);
g0(1)=fx(x0(1),x0(2));
g0(2)=fy(x0(1),x0(2));
f=@(x,y)(1-x)^2+80*((y-x^2)^2);
s0=-H0*g0;

%plot(x0)

phi=@(k)(1-(x0(1)+k*s0(1)))^2+80*((x0(2)+k*s0(2))-(x0(1)+k*s0(1))^2)^2;
kstar=fminbnd(phi,-10,10);
X1=x0+kstar*s0;
F=f(X1(1),X1(2));
d=f(X1(1),X1(2))-f(x0(1),x0(2));
scatter(n,X1(1),'filled')
scatter(n,X1(2))
hold on
%plot(X1)

while n<500

G1(1)=fx(X1(1),X1(2));
G1(2)=fy(X1(1),X1(2));
p1=G1-g0;
q1=X1-x0;
H1=H0-(H0*p1*p1.'*H0)/(p1.'*H0*p1)+(q1*q1.')/(p1.'*q1)
S1=-H1*G1;
phi=@(k)(1-(X1(1)+k*S1(1)))^2+80*((X1(2)+k*S1(2))-(X1(1)+k*S1(1))^2)^2;
kstar=fminbnd(phi,-10,10);
X2=X1+kstar.*S1;
F=f(X2(1),X2(2));
d=f(X2(1),X2(2))-f(X1(1),X1(2));
scatter(n+1,X2(1),'filled')
scatter(n+1,X2(2))

%plot(X2)
x0=X1;
```

```
g0=G1;
X1=X2;
H0=H1;
n=n+1;

if norm(X1-x0)<10^-8
    n=2500;
end
end
xlabel('x')
ylabel('y')
legend('y','x')
hold off

G=zeros(2,2);
x=1;
y=1;
G(1,1)=2-320*(y-x^2)+640*x^2;
G(1,2)=-320*x;
G(2,1)=-320*x;
G(2,2)=160;
H=inv(G)
```