Part II Computational Projects 2020

C3496B

Attach to front of report

Project number:

7.3

Minimization Methods

All programs are attached in the end of the report.

Question 1:

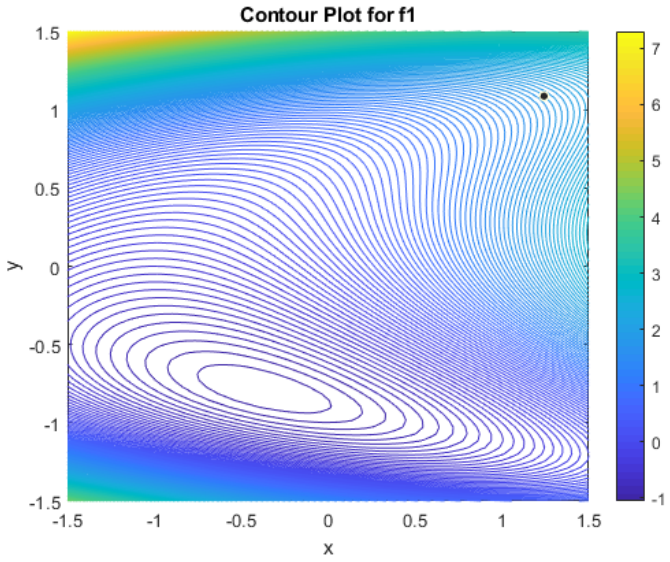

Figure 1,1: Contour Plot for f_1

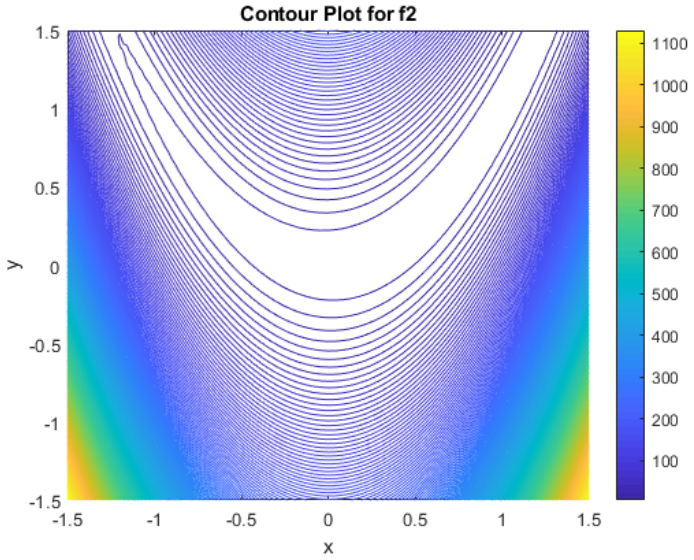

Figure 1,2: Contour Plot for f_2

The contour plots of f_1 and f_2 are shown above.

To work out the minimum of f_1 in the given region, we solve the equation $\nabla f_1 = 0$. The resultant value for (x, y) at the stationary point is $(2^{-\frac{2}{3}} - 1, -2^{-\frac{1}{3}})$. The corresponding Hessian matrix is

$$H_{f_1}(2^{-\frac{2}{3}} - 1, -2^{-\frac{1}{3}}) = \begin{bmatrix} 1 & 2^{\frac{2}{3}} \\ 2^{\frac{2}{3}} & 5 * 2^{\frac{1}{3}} \end{bmatrix} \tag{1}$$

Note that the Hessian has positive determinant $(= 3 * 2^{\frac{1}{3}})$ and trace. It is therefore positive definite. We conclude that $(2^{-\frac{2}{3}} - 1, -2^{-\frac{1}{3}})$ is indeed the minimum point of f_1 in the given region. The numerical value of f_1 at the minimum is $-2^{-1} - 2^{-\frac{1}{3}} + 2^{-\frac{7}{3}}$

To work out the minimum of f_2, we note that f_2 is non-negative for any real numbers x and y. We also note that 0 is attained at $(x, y) = (1, 1)$. Hence $(1, 1)$ is the minimum point of f_2 with $f_2(1, 1) = 0$.

We note also that at the minimum points, the Hessian of f_2 is

$$H_{f_2}(1, 1) = \begin{bmatrix} 642 & -320 \\ -320 & 160 \end{bmatrix} \tag{2}$$

Question 2:

Starting from $(-1.0, -1.3)$, we run the Steepest Descent method for 10 iterations for f_1. The plot of the progress is presented below:

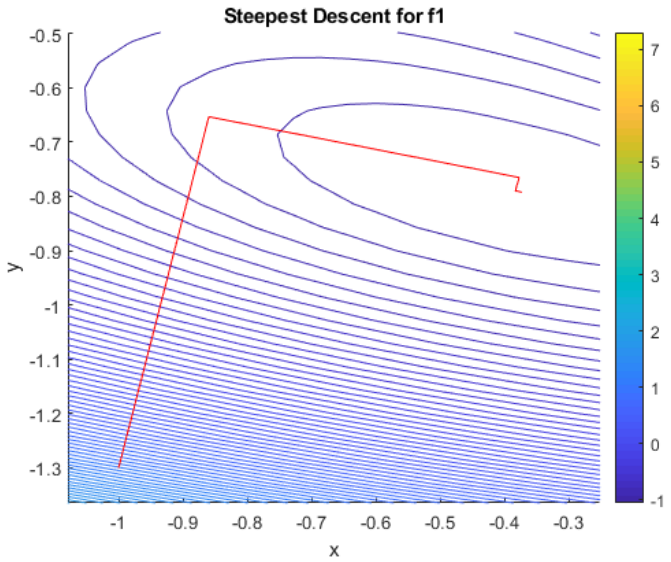

Figure 2,1: Steepest Descent for f_1, with 10 iterations

After 10 iterations, we finish the progress at $(-0.3734, -0.7923)$ with the final iteration resulting in an decrease of $2.0118 * 10^{-6}$ in the value of f_1. Since, from the contour plot, we are near the minimum, we may assume that further iterations will lead to decreases of f_1 with such order of magnitude. We may therefore estimate the minimal value of f_1, to 4 significant digits, to be -1.0953.

Similarly, the changes in x coordinate and y coordinate in the last 3 iterations are both of the order 10^{-3}. Hence by noting the stability near the minimum (i.e. x and y do not oscillate too much), we estimate the x and y intervals in which the minimum lies to be
$(-0.3734 - 0.0100, -0.3734 + 0.0100) = (-0.3834, -0.3634)$ and
$(-0.7923 - 0.0100, -0.7923 + 0.0100) = (-0.8023, -0.7823)$.

We now investigate the relationship between the precision of the estimated minimum point and the precision of the estimated minimal value of f_1. We use 'est' subscript to denote the estimated value, and 'min' subscript to denote the true minimal value.

Let ϵ to be the precision of the estimated minimum point, that is,

$$|x_{\text{est}} - x_{\text{min}}| \leq \epsilon \text{ and } |y_{\text{est}} - y_{\text{min}}| \leq \epsilon \tag{3}$$

Since $|x_{\text{est}}|$ and $|x_{\min}| \leq 1.5$, we have

$$\begin{aligned} |x_{\text{est}}^2 - x_{\min}^2| &= |x_{\text{est}} - x_{\min}||x_{\text{est}} + x_{\min}| \\ \leq &|x_{\text{est}} - x_{\min}|(|x_{\text{est}}| + |x_{\min}|) \leq 3\epsilon \end{aligned} \tag{4}$$

and similarly

$$|y_{\text{est}}^2 - y_{\min}^2| = \leq 3\epsilon \tag{5}$$

Therefore

$$\begin{aligned} &|f_1(x_{\min}, y_{\min}) - f_1(x_{\text{est}}, y_{\text{est}})| \\ &= |x_{\min} - x_{\text{est}} + y_{\min} - y_{\text{est}} + \frac{x_{\min}^2 - x_{\text{est}}^2}{4} + (y_{\min}^2 - \frac{x_{\min}}{2})^2 - (y_{\text{est}}^2 - \frac{x_{\text{est}}}{2})^2| \\ &\leq |x_{\min} - x_{\text{est}}| + |y_{\min} - y_{\text{est}}| + |\frac{x_{\min}^2 - x_{\text{est}}^2}{4}| + |(y_{\min}^2 - \frac{x_{\min}}{2})^2 - (y_{\text{est}}^2 - \frac{x_{\text{est}}}{2})^2| \\ &\leq \epsilon + \epsilon + \frac{3}{4}\epsilon + |(y_{\min}^2 - \frac{x_{\min}}{2})^2 - (y_{\text{est}}^2 - \frac{x_{\text{est}}}{2})^2|, \text{, by 3, 4, and 5} \\ &= \frac{11}{4}\epsilon + |y_{\min}^2 + y_{\text{est}}^2 - \frac{x_{\min}}{2} - \frac{x_{\text{est}}}{2}||y_{\min}^2 - y_{\text{est}}^2 + \frac{x_{\text{est}}}{2} - \frac{x_{\min}}{2}| \\ &\leq \frac{11}{4}\epsilon + (|y_{\min}^2| + |y_{\text{est}}^2| + |\frac{x_{\min}}{2}| + |\frac{x_{\text{est}}}{2}|)(|y_{\min}^2 - y_{\text{est}}^2| + |\frac{x_{\text{est}}}{2} - \frac{x_{\min}}{2}|) \\ &\leq \frac{11}{4}\epsilon + (2.25 + 2.25 + 0.75 + 0.75)(3\epsilon + \frac{1}{2}\epsilon), \text{ by 3 and 5} \\ &= \frac{95}{4}\epsilon \leq 25\epsilon \end{aligned} \tag{6}$$

Hence the error of the estimated minimal value of f_1 is at most 25 times larger than that of the estimated minimum point.
Throughout the chain of inequalities above we exploit the fact that f_1 is continuous and is supported on a bounded domain.

Question 3:

Starting from $(0.676, 0.443)$, we run the Steepest Descent method for 15 and 1500 iterations for f_2. The plot of the progress is presented below:

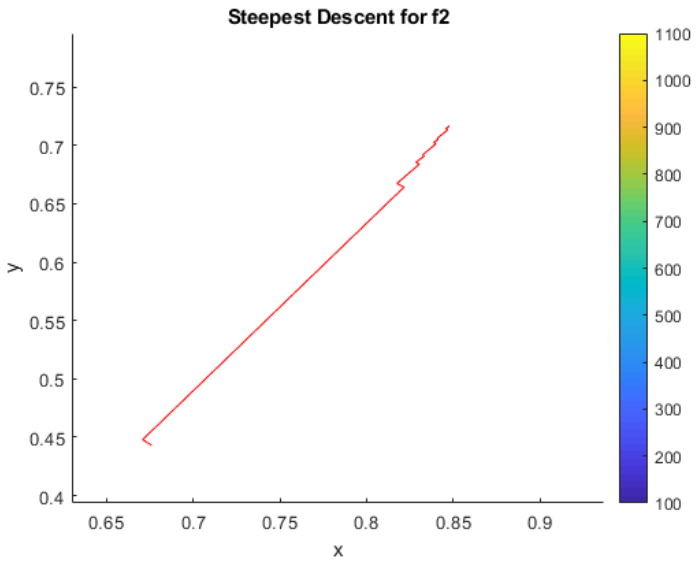

Figure 3,1: Steepest Descent for f_2, with 15 iterations

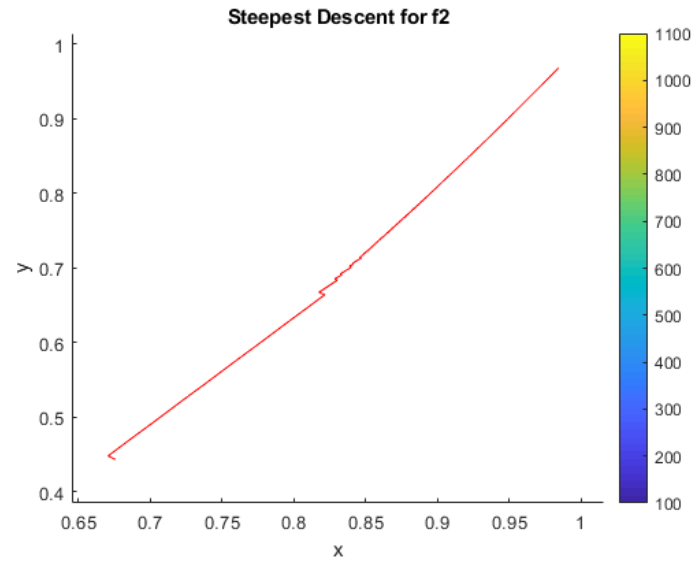

Figure 3,2: Steepest Descent for f_2, with 1500 iterations

From Figure 3.1 and 3.2 we see that the iteration converges, despite very slowly, to the theoretical minimum $(1, 1)$.

Now we vary λ^* slightly, producing the following graph. The first few original λ^*'s and new λ^*'s are also included.

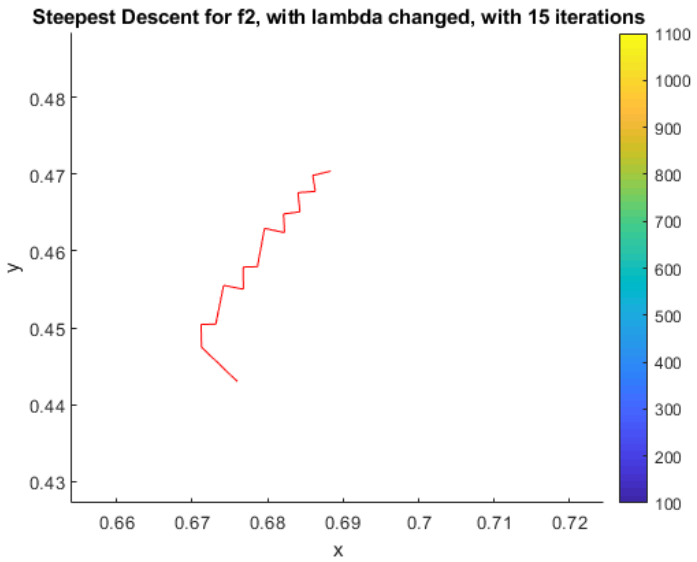

Figure 3,3: Steepest Descent for f_2, with λ^* changed and 15 iterations

The first three λ^* in Figure 3.1 are 0.0022, 0.6700 and 0.0017.
The first three λ^* in Figure 3.3 are 0.0020, 0.0060, and 0.0030.

Note that Figure 3.3 converges extremely slowly to the minimum $(1, 1)$. This indicates that a tiny change to λ^* can result in vast change to the rate of convergence, leading to an entirely different path of iterations.

In view of Figure 3.1 and 3.3, we conclude that the Steepest Descent Method is inefficient when there is a 'flat' region around the minimum point, which causes the iterations to zigzag slowly with small step sizes towards the minimum, or when the arithmetics are numerically unstable, which causes the erroneous evaluation of λ^*'s and thereby makes convergence slow or non-convergence.

Question 4:

Starting from $(-1.0, -1.3)$, we run the Conjugate Gradient Descent method for 10 iterations for f_1. The plot of the progress is presented below:

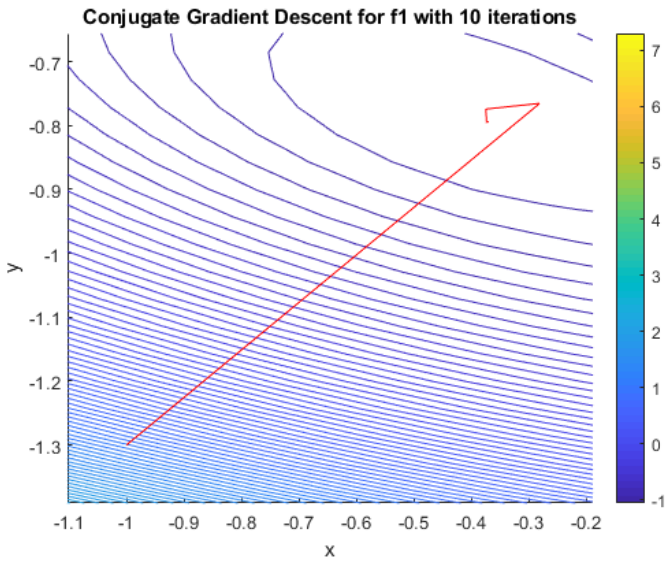

Figure 4,1: Steepest Descent for f_1, with 10 iterations

After 10 iterations we finish at $(-0.3700, -0.7937)$ and the iterations are stable at this point, which indicates that the Conjugate Gradient Descent method applied to f_1 does converge to the minimum.

We note that the estimated x- and y-coordinates, rounded up to 4 decimal places, do not vary in the last four iterations. We estimate that the x-coordinate of the minimum point lies in $(-0.3701, -0.3699)$ and that the y-coordinate of the minimum point lies in $(-0.7938, -0.7936)$. We also see that the Conjugate Gradient Descent method is a better method than the Steepest Descent method for f_1, since it gives a better approximation with same number of iterations. In fact, it only takes 6 iterations to obtain the first four decimal places for the minimum point.

Question 5:

Starting from $(0.676, 0.443)$, we run the Conjugate Gradient Descent method for 10 iterations for f_2. The plot of the progress is presented below:

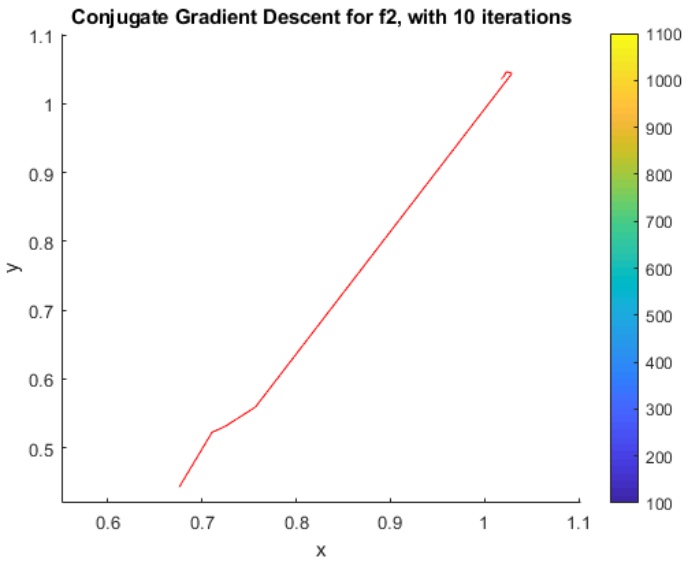

Figure 5,1: Conjugate Gradient Descent for f_2, with 10 iterations

We finish the iterations at $(1.0180, 1.0364)$, which is very close to the minimum point. The rate of convergence is much faster than the Steepest Descent method. Now we investigate how sensitive the iteration path is to variations of λ^*'s.

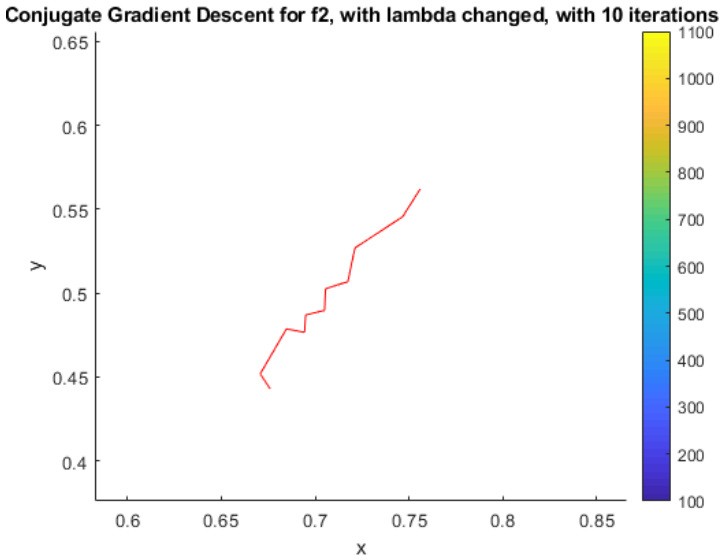

Figure 5,2: Conjugate Gradient Descent for f_2, with λ^* changed and 10 iterations

The first three λ^* in Figure 5.1 are 0.2100, 0.0450 and 0.1300.
The first three λ^* in Figure 5.2 are 0.0080, 0.0600, and 0.0090.

We see that variations in λ^*'s do not obstruct convergence too much compared to the Steepest Descent method, which implies that the Conjugate Gradient Descent method is a more numerically stable method for general minimization problems. Nonetheless, the Conjugate Gradient Descent method is still moderately affected by the numerical accuracy.

Question 6:

Define

$$f_3 = 0.4x^2 + 0.2y^2 + z^2 + xz \tag{7}$$

The Hessian of f_3 is

$$H_{f_3} = \begin{bmatrix} 0.8 & 0 & 1 \\ 0 & 0.4 & 0 \\ 1 & 0 & 2 \end{bmatrix} \tag{8}$$

Starting from $(1, 1, 1)$ we apply the DFP algorithm to the quadratic function f_3 using the sequence of values

$$\lambda^* = 0.3942, 2.5522, 4.2202.$$

The estimated minimum is $(4.317 * 10^{-6}, -4.243 * 10^{-5}, 2.656 * 10^{-5})$, which is very close to the true minimum $(0, 0, 0)$.
The iterated $\mathbf{H}$ matrix is

$$\mathbf{H} = \begin{bmatrix} 3.3333 & 0 & -1.6667 \\ 0 & 2.5 & 0 \\ -1.6667 & 0 & 1.3333 \end{bmatrix} \tag{9}$$

This is indeed the inverse Hessian matrix of f_3 at the minimum.

Now we investigate the sensitivity of DFP algorithm to changes in λ^*.
We use the sequence of values

$$\lambda^* = 0.3940, 2.5520, 4.1840.$$

The estimated minimum point is $(0.0027, -3.3627 * 10^{-5}, 0.0024)$, and the iterated $\mathbf{H}$ matrix is

$$\mathbf{H} = \begin{bmatrix} 3.3246 & 0.0001 & -1.6742 \\ 0.0001 & 2.5000 & 0.0001 \\ -1.6742 & 0.0001 & 1.3268 \end{bmatrix} \tag{10}$$

From the above result we see that the estimations given by the DFP algorithm changes continuously with respect to changes in values of λ^*.

Question 7:

Starting from $(-1.0, -1.3)$, we run the DFP algorithm for 10 iterations for f_1. We finish at $(-0.370039, -0.793700)$. The plot of the progress is presented below:

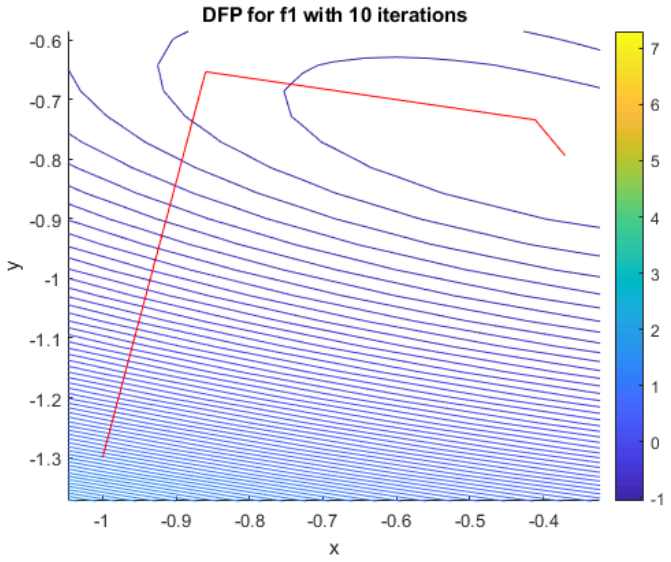

Figure 7,1: DFP algorithm for f_1, with 10 iterations

Clearly the algorithm is converging to the minimum point. We also note that the DFP algorithm applied to f_1 only takes 4 iterations to estimate the minimum point to 4 decimal places, which is better than the Steepest Descent and the Conjugate Descent method.

In addition, he estimated x- and y-coordinates up to 6 decimal places do not change after 7 iterations. Hence we estimate the x-interval to be $(-0.370040, -0.370038)$ and y-interval to be $(-0.793701, -0.793699)$.

The iterated $\mathbf{H}$ matrix is

$$\mathbf{H} = \begin{bmatrix} 1.6666 & -0.4200 \\ -0.4200 & 0.2646 \end{bmatrix} \tag{11}$$

Therefore $\mathbf{H}^{-1}$ is

$$\mathbf{H}^{-1} = \begin{bmatrix} 1.0001 & 1.5874 \\ 1.5874 & 6.2990 \end{bmatrix} \tag{12}$$

We see that $\mathbf{H}^{-1}$ is indeed converging to the Hessian of $f_1(1)$.

Question 8:

Starting from $(0.676, 0.443)$, we run the DFP algorithm for 10 iterations for f_2. The plot of the progress is presented below:

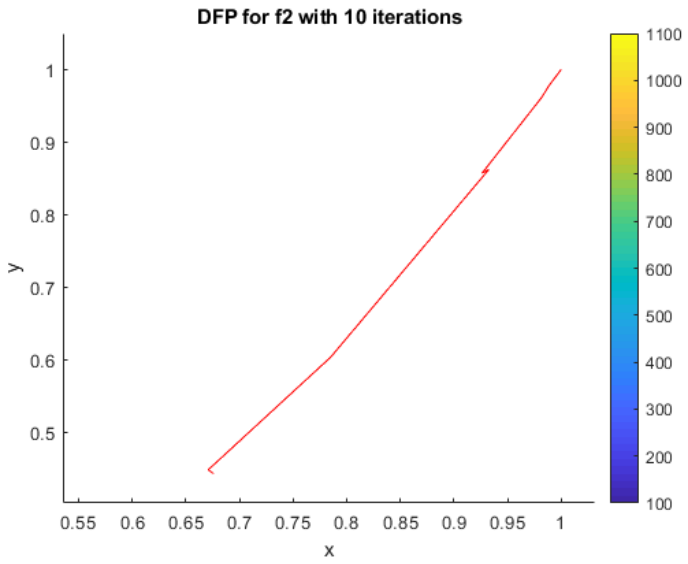

Figure 8,1: DFP algorithm for f_2, with 10 iterations

After 10 iterations we finish at $(1.0000, 1.0000)$, which is the minimum point. It takes only 10 iterations to converge to the minimum point, which is much better performance than the Steepest Descent method and the Conjugate Gradient Descent method.

The $\mathbf{H}^{-1}$ matrix is

$$\mathbf{H}^{-1} = \begin{bmatrix} 641.8849 & -319.9501 \\ -319.9501 & 159.9778 \end{bmatrix} \tag{13}$$

We see that, again, the iterated $\mathbf{H}^{-1}$ converges to the Hessian of $f_2(2)$.

Now we investigate how sensitive the DFP algorithm is to variations in λ^*'s.

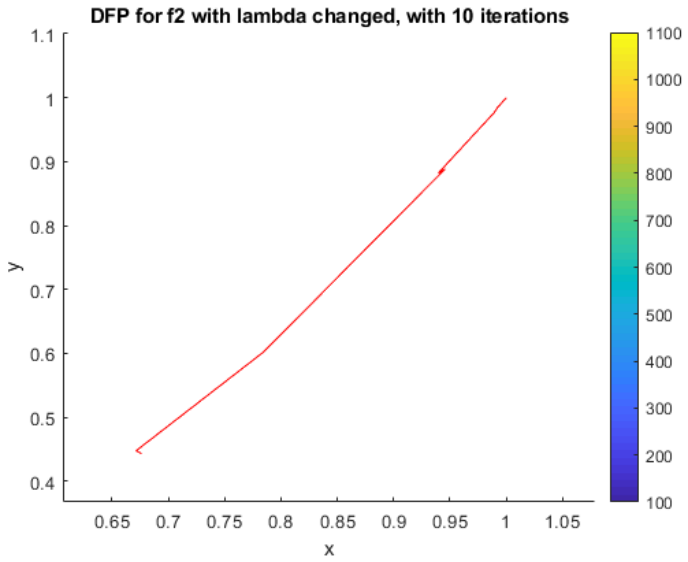

Figure 8,2: DFP algorithm for f_2, with λ^* changed and 10 iterations

We obtain Figure 8.2 by changing the first three λ^*'s from $0.0022, 0.4900, 5.2600$ to $0.0020, 0.4850, 5.8180$. Once again, the iteration path depends on, but not heavily, on the choice of λ^*'s.

Question 9:

We compare the performance of the three methods.

The Steepest Descent method is the easiest to implement. But it suffers from numerical instability and the local 'flatness' around the minimum. The rate of convergence of the Steepest Descent method is slow.

The Conjugate Gradient Descent method is based on the Steepest Descent method, and it overcomes the two issues of the Steepest Descent method. The rate of convergence of the Conjugate Gradient Descent method is faster than the Steepest Descent method.

The DFP algorithm is the best among the three methods: it doesn't suffer from numerical instability too much, and it ignores the local 'flatness' around the minimum. Moreover, the rate of convergence for the DFP algorithm is the fastest among the three methods.

Source Code

```

function v=f1(x,y)
v = x+y+x^2/4-y^2+(y^2-x/2)^2;
end
```

```
function v = f2(x,y)
v = (1-x)^2+80*(y-x^2)^2;
end
```

```
function v = f3(x,y,z)
v = 0.4*x^2+0.2*y^2+z^2+x*z;
end
```

```
function grad = gradf1(x,y)
grad = zeros(1,2);
grad(1) = 1+x-y^2;
grad(2) = 1-2*y-2*x*y+4*y^3;
end
```

```
function grad = gradf2(x,y)
grad = zeros(1,2);
grad(1) = -2+2*x-320*x*y+320*x^3;
grad(2) = 160*y-160*x^2;
end
```

```
function gradf3 = gradf3(x,y,z)
gradf3 = zeros(3,1);
gradf3(1) = 0.8*x+z;
gradf3(2) = 0.4*y;
gradf3(3) = 2*z+x;
end
```

```
function [x_val_2,y_val_2,f_val_2] = ...
    conjugateGSf1(x_ini,y_ini,x_val_1,y_val_1,f_val_1,n)
grad_0 = gradf1(x_ini,y_ini);
grad_1 = gradf1(x_val_1,y_val_1);
const = ...
    (grad_1(1)*grad_1(1)+grad_1(2)*grad_1(2))/(grad_0(1)*grad_0(1)+grad_0(2)*grad_0(2)
```

```

x_val_2 = x_val_1;
y_val_2 = y_val_1;
f_val_2 = f_val_1;
%we start the conjugate GS.
dir = -grad_1 - const*grad_0;
lambda = 0;
h = 10^(-n);%h stands for step size.


for i = 1:200
f_temp = f1(x_val_1+i*h*dir(1),y_val_1+i*h*dir(2));
if f_temp<f_val_1
lambda = i*h;
x_val_2 = x_val_1+lambda*dir(1);
y_val_2 = y_val_1+lambda*dir(2);
f_val_2 = f_temp;
end
end


if n < 10
if lambda ≤ 10^(-n+1)%i ≤ 10, hence only 1 significant digit
[x_val_2,y_val_2,f_val_2] = ...
    conjugateGSf1(x_ini,y_ini,x_val_1,y_val_1,f_val_1,n+1);
end
end



end
```

```

function [x_val_2,y_val_2,f_val_2,lambda] = ...
    conjugateGSf2(x_ini,y_ini,x_val_1,y_val_1,f_val_1,n)
grad_0 = gradf2(x_ini,y_ini);
grad_1 = gradf2(x_val_1,y_val_1);
const = ...
    (grad_1(1)*grad_1(1)+grad_1(2)*grad_1(2))/(grad_0(1)*grad_0(1)+grad_0(2)*grad_0(2)


x_val_2 = x_val_1;
y_val_2 = y_val_1;
f_val_2 = f_val_1;
%we start the conjugate GS.
dir = -grad_1 - const*grad_0;
lambda = 0;
h = 10^(-n);%h stands for step size.


for i = 1:200
f_temp = f2(x_val_1+i*h*dir(1),y_val_1+i*h*dir(2));
```

```
if f_temp<f_val_1
lambda = i*h;
x_val_2 = x_val_1+lambda*dir(1);
y_val_2 = y_val_1+lambda*dir(2);
f_val_2 = f_temp;
end
end


if n < 10
if lambda ≤ 10^(-n+1)%i ≤ 10, hence only 1 significant digit
[x_val_2,y_val_2,f_val_2,lambda] = ...
    conjugateGSf2(x_ini,y_ini,x_val_1,y_val_1,f_val_1,n+1);
end
end



end
%
```

```
function [x_val,y_val,f_val,lambda,hes] = ...
    DFPf1(x_ini,y_ini,hes_ini,n)
s = -hes_ini*transpose(gradf1(x_ini,y_ini));
x_val = x_ini;
y_val = y_ini;
f_val = f1(x_ini,y_ini);
lambda = 0;
h = 10^(-n);% h is the resolution.

for i = 1:2000
lambda_temp = i*h;
f_temp = f1(x_ini+lambda_temp*s(1),y_ini+lambda_temp*s(2));
if f_temp<f_val
lambda = lambda_temp;
x_val = x_ini+lambda_temp*s(1);
y_val = y_ini+lambda_temp*s(2);
f_val = f_temp;
end
end

if n<10
if lambda ≤ 10^(-n+1)
[x_val,y_val,f_val,lambda] = DFPf1(x_ini,y_ini,hes_ini,n+1);
end
end

p = ...
    transpose(gradf1(x_ini+lambda*s(1),y_ini+lambda*s(2))-gradf1(x_ini,y_ini));
q = lambda*s;%so p and q are column vectors.

hes = hes_ini - ...
```

```
    (1/(transpose(p)*hes_ini*p))*hes_ini*p*transpose(p)*hes_ini+q*transpose(q)/(transp
```

```
function [x_val,y_val,f_val,lambda,hes] = ...
    DFPf2(x_ini,y_ini,hes_ini,n)
s = -hes_ini*transpose(gradf2(x_ini,y_ini));
x_val = x_ini;
y_val = y_ini;
f_val = f2(x_ini,y_ini);
lambda = 0;
h = 10^(-n);% h is the resolution.

for i = 1:2000
lambda_temp = i*h;
f_temp = f2(x_ini+lambda_temp*s(1),y_ini+lambda_temp*s(2));
if f_temp<f_val
lambda = lambda_temp;
x_val = x_ini+lambda_temp*s(1);
y_val = y_ini+lambda_temp*s(2);
f_val = f_temp;
end
end

if n<10
if lambda ≤ 10^(-n+1)
[x_val,y_val,f_val,lambda] = DFPf2(x_ini,y_ini,hes_ini,n+1);
end
end

p = ...
    transpose(gradf2(x_ini+lambda*s(1),y_ini+lambda*s(2))-gradf2(x_ini,y_ini));
q = lambda*s;%so p and q are column vectors.

hes = hes_ini - ...
    (1/(transpose(p)*hes_ini*p))*hes_ini*p*transpose(p)*hes_ini+q*transpose(q)/(transp
```

```
function [x_val,y_val,z_val,f_val,lambda,hes] = ...
    DFPf3(x_ini,y_ini,z_ini,hes_ini,h)
s = -hes_ini*(gradf3(x_ini,y_ini,z_ini));
x_val = x_ini;
y_val = y_ini;
z_val = z_ini;
f_val = f3(x_ini,y_ini,z_ini);
lambda = 0;
for i = 1:50000
lambda_temp = i*h;
f_temp = ...
    f3(x_ini+lambda_temp*s(1),y_ini+lambda_temp*s(2),z_ini+lambda_temp*s(3));
if f_temp<f_val
lambda = lambda_temp;
x_val = x_ini+lambda_temp*s(1);
```

```
y_val = y_ini+lambda_temp*s(2);
z_val = z_ini+lambda_temp*s(3);
f_val = f_temp;
end
end


p = (gradf3(x_val,y_val,z_val)-gradf3(x_ini,y_ini,z_ini));
q = lambda*s;%so p and q are column vectors.

hes = hes_ini - ...
    (1/(transpose(p)*hes_ini*p))*hes_ini*p*transpose(p)*hes_ini+q*transpose(q)/(transp
```

```

function [x_val_2,y_val_2,f_val_2,lambda] = ...
    res_conjugateGSf2(x_ini,y_ini,x_val_1,y_val_1,f_val_1,res)
grad_0 = gradf2(x_ini,y_ini);
grad_1 = gradf2(x_val_1,y_val_1);
const = ...
    (grad_1(1)*grad_1(1)+grad_1(2)*grad_1(2))/(grad_0(1)*grad_0(1)+grad_0(2)*grad_0(2)


x_val_2 = x_val_1;
y_val_2 = y_val_1;
f_val_2 = f_val_1;
%we start the conjugate GS.
dir = -grad_1 - const*grad_0;


for i = 1:(floor(2/res))
f_temp = f2(x_val_1+i*res*dir(1),y_val_1+i*res*dir(2));
if f_temp<f_val_1
lambda = i*res;
x_val_2 = x_val_1+lambda*dir(1);
y_val_2 = y_val_1+lambda*dir(2);
f_val_2 = f_temp;
end
end




end
%
```

```

function [x_val,y_val,f_val,lambda,hes] = ...
    res_DFPf2(x_ini,y_ini,hes_ini,res)
s = -hes_ini*transpose(gradf2(x_ini,y_ini));
x_val = x_ini;
```

```
y_val = y_ini;
f_val = f2(x_ini,y_ini);
lambda = 0;

for i = 1:20000
lambda_temp = i*res;
f_temp = f2(x_ini+lambda_temp*s(1),y_ini+lambda_temp*s(2));
if f_temp<f_val
lambda = lambda_temp;
x_val = x_ini+lambda_temp*s(1);
y_val = y_ini+lambda_temp*s(2);
f_val = f_temp;
end
end


p = ...
    transpose(gradf2(x_ini+lambda*s(1),y_ini+lambda*s(2))-gradf2(x_ini,y_ini));
q = lambda*s;%so p and q are column vectors.

hes = hes_ini - ...
    (1/(transpose(p)*hes_ini*p))*hes_ini*p*transpose(p)*hes_ini+q*transpose(q)/(transp
```

```
function [x_val,y_val,f_val,lambda] = ...
    res_steepestDesf2(x_ini,y_ini,res)%we manually select ...
    the resolution
lambda = 0;
grad = gradf2(x_ini,y_ini);
f_val = f2(x_ini,y_ini);
x_val = x_ini;
y_val = y_ini;

for i = 1:floor(2/res)
f_temp = f2(x_ini-res*i*grad(1),y_ini-res*i*grad(2));
if f_temp<f_val
x_val = x_ini-res*i*grad(1);
y_val = y_ini-res*i*grad(2);
lambda = res*i;
f_val = f_temp;
end
end

end
```

```

function [x_val,y_val,f_val,lambda] = ...
    steepestDesf1(x_ini,y_ini,n)%n is the resolutoin parameter
lambda = 0;
grad = gradf1(x_ini,y_ini);
f_val = f1(x_ini,y_ini);
```

```
x_val = x_ini;
y_val = y_ini;
h = 10^(-n);%h stands for step size.


for i = 1:200
f_temp = f1(x_ini-h*i*grad(1),y_ini-h*i*grad(2));
if f_temp<f_val
x_val = x_ini-h*i*grad(1);
y_val = y_ini-h*i*grad(2);
lambda = h*i;
f_val = f_temp;

end
end

if n<10
if lambda ≤ 10^(-n+1)
[x_val,y_val,f_val,lambda] = steepestDesf1(x_ini,y_ini,n+1);
end
end




end
```

```

function [x_val,y_val,f_val,lambda] = ...
    steepestDesf2(x_ini,y_ini,n)%n is the resolution parameter
lambda = 0;
grad = gradf2(x_ini,y_ini);
f_val = f2(x_ini,y_ini);
x_val = x_ini;
y_val = y_ini;
h = 10^(-n);%h stands for step size.


for i = 1:200
f_temp = f2(x_ini-h*i*grad(1),y_ini-h*i*grad(2));
if f_temp<f_val
x_val = x_ini-h*i*grad(1);
y_val = y_ini-h*i*grad(2);
lambda = h*i;
f_val = f_temp;
end
end

if lambda ≤ 10^(-n+1)
[x_val,y_val,f_val,lambda] = steepestDesf2(x_ini,y_ini,n+1);
end

```

```
25
26
27
28 end
```

```
1  f =@(x,y) x + y + x^2/4-y^2+(y^2-x/2)^2;
2  t = fcontour(f,[-1.5 1.5 -1.5 1.5]);
3  t.LevelStep = 0.05;
4  title({'Contour Plot for f1'})
5  xlabel('x')
6  ylabel('y')
7  colorbar;
8
9  g = @(x,y) (1-x)^2+80*(y-x^2)^2;
10 s = fcontour(g,[-1.5 1.5 -1.5 1.5]);
11 s.LevelStep = 5;
12 title({'Contour Plot for f2'})
13 xlabel('x')
14 ylabel('y')
15 colorbar
```

```
1  X = zeros(1,11);
2  Y = zeros(1,11);
3  flist = zeros(1,11);
4  X(1) = -1.0;
5  Y(1) = -1.3;
6  flist(1) = f1(X(1),Y(1));
7  lambdaList = zeros(1,10);
8  for i  = 2:11
9  [x_val,y_val,f_val,lambda] = steepestDesf1(X(i-1),Y(i-1),2);
10 lambdaList(i-1) = lambda;
11 X(i) = x_val;
12 Y(i) = y_val;
13 flist(i) = f_val;
14 decrease = -f_val+flist(i-1)
15 end
16
17 hold on
18
19 f =@(x,y) x + y + x^2/4-y^2+(y^2-x/2)^2;
20 t = fcontour(f,[-1.5 1.5 -1.5 1.5]);
21 t.LevelStep = 0.05;
22 f =@(x,y) x + y + x^2/4-y^2+(y^2-x/2)^2;
23 t = fcontour(f,[-1.5 1.5 -1.5 1.5]);
24 t.LevelStep = 0.05;
25 title({'Steepest Descent for f1'})
26 xlabel('x')
27 ylabel('y')
28 colorbar;
29
```

```
plot(X,Y,'red')
hold off
%analytic x- and y-val is 2^(-2/3)-1,-2^(-1/3).
%analytic fvalue at min is ...
    2^(-2/3)-1-2^(-1/3)+0.25*(2^(-2/3)-1)^2-2^(-2/3)+(0.5*2^(-2/3)+0.5)^2 ...
    = -0.5-2^(-1/3)+2^(-7/3)
```

```
% steepest GS for f2
X = zeros(1,11);
Y = zeros(1,11);
flist = zeros(1,11);
X(1) = 0.676;
Y(1) = 0.443;
flist(1) = f2(X(1),Y(1));
lambdaList = zeros(1,10);
%i = 16: 0.8470,0.7168
%i = 1501: 0.9481,0.9684
for i  = 2:16
[x_val,y_val,f_val,lambda] = steepestDesf2(X(i-1),Y(i-1),2);
lambdaList(i-1) = lambda;
X(i) = x_val;
Y(i) = y_val;
flist(i) = f_val;
decrease = -f_val+flist(i-1)

end

hold on

f =@(x,y) (1-x)^2+80*(y-x^2)^2;
t = fcontour(f,[-1.5 1.5 -1.5 1.5]);
title({'Steepest Descent for f2'})
xlabel('x')
ylabel('y')
colorbar;

plot(X,Y,'red')
hold off

%the analytic min fvalue is 0.


%consider f2, with resolution manually set
X = zeros(1,11);
Y = zeros(1,11);
flist = zeros(1,11);
X(1) = 0.676;
Y(1) = 0.443;
flist(1) = f2(X(1),Y(1));
lambdaList = zeros(1,10);
res = 0.001;%%%resolution!!!
for i  = 2:16
```

```
[x_val,y_val,f_val,lambda] = ...
    res_steepestDesf2(X(i-1),Y(i-1),res);%to investigate ...
    effect of resolution of lambda we introduce the ...
    variable res
X(i) = x_val;
lambdaList(i-1) = lambda;
Y(i) = y_val;
flist(i) = f_val;
decrease = flist(i-1) - flist(i)

end

hold on

f =@(x,y) (1-x)^2+80*(y-x^2)^2;
t = fcontour(f,[-1.5 1.5 -1.5 1.5]);
title({'Steepest Descent for f2, with lambda changed, with ...
    15 iterations'})
xlabel('x')
ylabel('y')
colorbar;

plot(X,Y,'red')
hold off
```

```
%conjugate for f1

X = zeros(1,11);
Y = zeros(1,11);
flist = zeros(1,11);
X(1) = -1.0;
Y(1) = -1.3;
flist(1) = f1(X(1),Y(1));
for i  = 2:11
[x_val_1,y_val_1,f_val_1] = steepestDesf1(X(i-1),Y(i-1),2);
[X(i),Y(i),flist(i)] = ...
    conjugateGSf1(X(i-1),Y(i-1),x_val_1,y_val_1,f_val_1,2);
decrease = flist(i-1) - flist(i)
end

hold on

f =@(x,y) x + y + x^2/4-y^2+(y^2-x/2)^2;
t = fcontour(f,[-1.5 1.5 -1.5 1.5]);
t.LevelStep = 0.05;
title({'Conjugate Gradient Descent for f1 with 10 iterations'})
xlabel('x')
ylabel('y')
colorbar;

plot(X,Y,'red')
hold off
```

```
27
28 %analytic fvalue at min is ...
       2^(-2/3)-1-2^(-1/3)+0.25*(2^(-2/3)-1)^2-2^(-2/3)+(0.5*2^(-2/3)+0.5)^2
```

```
1
2  %conjugate for f2
3
4  X = zeros(1,11);
5  Y = zeros(1,11);
6  flist = zeros(1,11);
7  X(1) = 0.676;
8  Y(1) = 0.443;
9  flist(1) = f2(X(1),Y(1));
10 lambdaList = zeros(1,10);
11 for i  = 2:11
12 [x_val_1,y_val_1,f_val_1] = steepestDesf2(X(i-1),Y(i-1),2);
13 [X(i),Y(i),flist(i),lambdaList(i-1)] = ...
       conjugateGSf2(X(i-1),Y(i-1),x_val_1,y_val_1,f_val_1,2);
14 decrease = flist(i-1) - flist(i)
15 end
16
17 hold on
18
19 f =@(x,y) (1-x)^2+80*(y-x^2)^2;
20 t = fcontour(f,[-1.5 1.5 -1.5 1.5]);
21 title({'Conjugate Gradient Descent for f2, with 10 ...
       iterations'})
22 xlabel('x')
23 ylabel('y')
24 colorbar;
25 plot(X,Y,'red')
26
27 hold off
28
29 %the analytic min fvalue is 0.
30
31
32 %consider f2, with resolution manually set
33
34 X = zeros(1,11);
35 Y = zeros(1,11);
36 flist = zeros(1,11);
37 X(1) = 0.676;
38 Y(1) = 0.443;
39 flist(1) = f2(X(1),Y(1));
40 lambdaList = zeros(1,10);
41 res = 0.001;%resolution!!
42 for i  = 2:11
43 [x_val_1,y_val_1,f_val_1] = ...
       res_steepestDesf2(X(i-1),Y(i-1),res);
44 [X(i),Y(i),flist(i),lambdaList(i-1)] = ...
       res_conjugateGSf2(X(i-1),Y(i-1),x_val_1,y_val_1,f_val_1,res);
```

```
decrease = flist(i-1) - flist(i)
end

hold on

f =@(x,y) (1-x)^2+80*(y-x^2)^2;
t = fcontour(f,[-1.5 1.5 -1.5 1.5]);
title({'Conjugate Gradient Descent for f2, with lambda ...
    changed, with 10 iterations'})
xlabel('x')
ylabel('y')
colorbar;
plot(X,Y,'red')

hold off
```

```
X = zeros(1,11);
Y = zeros(1,11);
Z = zeros(1,11);
flist = zeros(1,11);
X(1) = 1;
Y(1) = 1;
Z(1) = 1;
flist(1) = f3(X(1),Y(1),Z(1));
lambdaList = zeros(1,10);%lambda will be valuated to the ...
    desire value
hes = eye(3);
h = 0.0001;
for i  = 2:4
[X(i),Y(i),Z(i),flist(i),lambdaList(i-1),hes] = ...
    DFPf3(X(i-1),Y(i-1),Z(i-1),hes,h);
decrease = flist(i-1)-flist(i)
hes
end

%investigate stability
X = zeros(1,11);
Y = zeros(1,11);
Z = zeros(1,11);
flist = zeros(1,11);
X(1) = 1;
Y(1) = 1;
Z(1) = 1;
flist(1) = f3(X(1),Y(1),Z(1));
lambdaList = zeros(1,10);%lambdalist = ...
    [0.394000000000000,2.55200000000000,4.18400000000000]
hes = eye(3);
h = 0.001;

for i  = 2:4
[X(i),Y(i),Z(i),flist(i),lambdaList(i-1),hes] = ...
    DFPf3(X(i-1),Y(i-1),Z(i-1),hes,h);
```

```
decrease = flist(i-1)-flist(i)
hes
end
```

```

X = zeros(1,11);
Y = zeros(1,11);
flist = zeros(1,11);
X(1) = -1.0;
Y(1) = -1.3;
flist(1) = f1(X(1),Y(1));
lambdaList = zeros(1,10);
hes = eye(2);
for i  = 2:11
[x_val,y_val,f_val,lambda,hes] = DFPf1(X(i-1),Y(i-1),hes,2);
lambdaList(i-1) = lambda;
X(i) = x_val;
Y(i) = y_val;
flist(i) = f_val;
decrease = -f_val+flist(i-1)
hes
end

hold on

f =@(x,y) x + y + x^2/4-y^2+(y^2-x/2)^2;
t = fcontour(f,[-1.5 1.5 -1.5 1.5]);
t.LevelStep = 0.05;
title({'DFP for f1 with 10 iterations'})
xlabel('x')
ylabel('y')
colorbar;

plot(X,Y,'red')
hold off

%analytic fvalue at min is ...
    2^(-2/3)-1-2^(-1/3)+0.25*(2^(-2/3)-1)^2-2^(-2/3)+(0.5*2^(-2/3)+0.5)^2
```

```

X = zeros(1,11);
Y = zeros(1,11);
flist = zeros(1,11);
X(1) = 0.676;
Y(1) = 0.443;
lambdaList = zeros(1,10);
flist(1) = f2(X(1),Y(1));
hes = eye(2);
for i  = 2:11
[x_val,y_val,f_val,lambda,hes] = DFPf2(X(i-1),Y(i-1),hes,2);
```

```
lambdaList(i-1) = lambda;
X(i) = x_val;
Y(i) = y_val;
flist(i) = f_val;
decrease = -f_val+flist(i-1)
hes
end

hold on

f =@(x,y) (1-x)^2+80*(y-x^2)^2;
t = fcontour(f,[-1.5 1.5 -1.5 1.5]);
title({'DFP for f2 with 10 iterations'})
xlabel('x')
ylabel('y')
colorbar;
plot(X,Y,'red')

hold off

%the analytic min fvalue is 0.



%consider f2, with resolution manually set

X = zeros(1,11);
Y = zeros(1,11);
flist = zeros(1,11);
X(1) = 0.676;
Y(1) = 0.443;
flist(1) = f2(X(1),Y(1));
res = 0.001;%resolution!!
hes = eye(2);
for i  = 2:11
[x_val,y_val,f_val,lambda,hes] = ...
    res_DFPf2(X(i-1),Y(i-1),hes,res);
lambdaList(i-1) = lambda;
X(i) = x_val;
Y(i) = y_val;
flist(i) = f_val;
decrease = -f_val+flist(i-1)
hes
end

hold on

f =@(x,y) (1-x)^2+80*(y-x^2)^2;
t = fcontour(f,[-1.5 1.5 -1.5 1.5]);
colorbar;
title({'DFP for f2 with lambda changed, with 10 iterations'})
xlabel('x')
ylabel('y')
plot(X,Y,'red')
```

```
65
66 hold off
```