MECH 579
Multidisciplinary Design Optimization
Siva Nadarajah

Project 5: MDO via MDF

20/20

Dritan Harizaj
260426327

November 29, 2013

Introduction

We are asked to minimize the following function with respect to x_1, x_2 using an MDF approach.

$$f(x,y) = -20e^{-[(x_1-1)^2+0.25(x_2-1)^2]} + y_1 + \cos(y_2)$$

The function is subject to the following governing equations.

$$R_1\big(x, y_1(x,y_2)\big): y_1 = -3e^{-[(x_1+1)^2+0.25(x_2+1)^2]} + \sin(y_2)$$
$$R_2\big(x, y_2(x,y_1)\big): y_2 = -3e^{-[5(x_1-3)^2+0.25(x_2-3)^2]} + e^{-y_1}$$

The starting point is (x_1 , x_2) = (-0.1 , -1)

The MDF approach requires for the coupling variables y_1 and y_2 to have converged before running them through the optimizer. Since we cannot implicitly solve for these 2 variables, an initial guess $y_2 = 1$ was chosen, which allows y_1 to be computed from the first governing equation, which is then fed back to the second governing equation to update y_2. This process was repeated until both values converged.

The derivatives of the main function are:

$$\frac{df}{dx_1} = \frac{\partial f}{\partial x_1} + \frac{\partial f}{\partial y_1}\frac{dy_1}{dx_1} + \frac{\partial f}{\partial y_2}\frac{dy_2}{dx_1}$$

$$\frac{df}{dx_2} = \frac{\partial f}{\partial x_2} + \frac{\partial f}{\partial y_1}\frac{dy_1}{dx_2} + \frac{\partial f}{\partial y_2}\frac{dy_2}{dx_2}$$

where

$$\frac{\partial f}{\partial x_1} = (2x_1 - 2)20e^{-[(x_1-1)^2+0.25(x_2-1)^2]}$$

$$\frac{\partial f}{\partial x_2} = \left(\frac{x_2 - 1}{2}\right)20e^{-[(x_1-1)^2+0.25(x_2-1)^2]}$$

$$\frac{\partial f}{\partial y_1} = 1$$

$$\frac{\partial f}{\partial y_2} = -\sin(y_2)$$

To compute $\frac{dy_i}{dx_n}$, we must use the derivatives of the governing equations with respect to the design variables.

$$\frac{dR_1}{dx_1} = \frac{\partial R_1}{\partial x_1} + \frac{\partial R_1}{\partial y_1}\frac{dy_1}{dx_1} + \frac{\partial R_1}{\partial y_2}\frac{dy_2}{dx_1} = 0$$

$$\frac{dR_1}{dx_2} = \frac{\partial R_1}{\partial x_2} + \frac{\partial R_1}{\partial y_1}\frac{dy_1}{dx_2} + \frac{\partial R_1}{\partial y_2}\frac{dy_2}{dx_2} = 0$$

$$\frac{dR_2}{dx_1} = \frac{\partial R_2}{\partial x_1} + \frac{\partial R_2}{\partial y_1}\frac{dy_1}{dx_1} + \frac{\partial R_2}{\partial y_2}\frac{dy_2}{dx_1} = 0$$

$$\frac{dR_2}{dx_2} = \frac{\partial R_2}{\partial x_2} + \frac{\partial R_2}{\partial y_1}\frac{dy_1}{dx_2} + \frac{\partial R_2}{\partial y_2}\frac{dy_2}{dx_2} = 0$$

It can be shown that

$$\frac{\partial R_1}{\partial x_1} = -(2x_1 + 2)3e^{-[(x_1+1)^2+0.25(x_2+1)^2]}$$

$$\frac{\partial R_1}{\partial x_2} = -\left(\frac{x_2 + 1}{2}\right)3e^{-[(x_1+1)^2+0.25(x_2+1)^2]}$$

$$\frac{\partial R_2}{\partial x_1} = -(10x_1 - 30)3e^{-[5(x_1-3)^2+0.25(x_2-3)^2]}$$

$$\frac{\partial R_2}{\partial x_2} = -\left(\frac{x_2 - 3}{2}\right)3e^{-[5(x_1-3)^2+0.25(x_2-3)^2]}$$

$$\frac{\partial R_1}{\partial y_1} = 1$$

$$\frac{\partial R_1}{\partial y_2} = -\cos(y_2)$$

$$\frac{\partial R_2}{\partial y_1} = e^{-y_1}$$

$$\frac{\partial R_2}{\partial y_2} = 1$$

Plugging back into the above equations:

$$-(2x_1+2)3e^{-[(x_1+1)^2+0.25(x_2+1)^2]}+1*\frac{dy_1}{dx_1}-\cos(y_2)*\frac{dy_2}{dx_1}=0$$

$$-\left(\frac{x_2+1}{2}\right)3e^{-[(x_1+1)^2+0.25(x_2+1)^2]}+1*\frac{dy_1}{dx_2}-\cos(y_2)*\frac{dy_2}{dx_2}=0$$

$$-(10x_1-30)3e^{-[5(x_1-3)^2+0.25(x_2-3)^2]}+e^{-y_1}*\frac{dy_1}{dx_1}+1*\frac{dy_2}{dx_1}=0$$

$$-\left(\frac{x_2-3}{2}\right)3e^{-[5(x_1-3)^2+0.25(x_2-3)^2]}+e^{-y_1}*\frac{dy_1}{dx_2}+1*\frac{dy_2}{dx_2}=0$$

Arranging in matrix form:

$$\begin{bmatrix}1 & 0 & -\cos(y_2) & 0\\ 0 & 1 & 0 & -\cos(y_2)\\ e^{-y_1} & 0 & 1 & 0\\ 0 & e^{-y_1} & 0 & 1\end{bmatrix}\begin{bmatrix}\frac{dy_1}{dx_1}\\ \frac{dy_1}{dx_2}\\ \frac{dy_2}{dx_1}\\ \frac{dy_2}{dx_2}\end{bmatrix}=\begin{bmatrix}(2x_1+2)3e^{-[(x_1+1)^2+0.25(x_2+1)^2]}\\ \left(\frac{x_2+1}{2}\right)3e^{-[(x_1+1)^2+0.25(x_2+1)^2]}\\ (10x_1-30)3e^{-[5(x_1-3)^2+0.25(x_2-3)^2]}\\ \left(\frac{x_2-3}{2}\right)3e^{-[5(x_1-3)^2+0.25(x_2-3)^2]}\end{bmatrix}$$

At this point, there are 2 main methods for solving the problem: direct and adjoint.

Direct:

$$\begin{bmatrix}\frac{dy_1}{dx_1}\\ \frac{dy_1}{dx_1}\end{bmatrix}=\begin{bmatrix}1 & -\cos(y_2)\\ e^{-y_1} & 1\end{bmatrix}^{-1}\begin{bmatrix}(2x_1+2)3e^{-[(x_1+1)^2+0.25(x_2+1)^2]}\\ (10x_1-30)3e^{-[5(x_1-3)^2+0.25(x_2-3)^2]}\end{bmatrix}$$

$$\begin{bmatrix}\frac{dy_1}{dx_2}\\ \frac{dy_2}{dx_2}\end{bmatrix}=\begin{bmatrix}1 & -\cos(y_2)\\ e^{-y_1} & 1\end{bmatrix}^{-1}\begin{bmatrix}\left(\frac{x_2+1}{2}\right)3e^{-[(x_1+1)^2+0.25(x_2+1)^2]}\\ \left(\frac{x_2-3}{2}\right)3e^{-[5(x_1-3)^2+0.25(x_2-3)^2]}\end{bmatrix}$$

Followed by plugging into the $\frac{df}{dx_2}$ and $\frac{df}{dx_1}$ equations directly.

Adjoint

Define

$$\psi = \begin{bmatrix} \frac{df}{dy_1} \\ \frac{df}{dy_2} \end{bmatrix}^T \begin{bmatrix} \frac{\partial R_1}{\partial y_1} & \frac{\partial R_1}{\partial y_2} \\ \frac{\partial R_2}{\partial y_1} & \frac{\partial R_2}{\partial y_2} \end{bmatrix} = \begin{bmatrix} 1 \\ -sin(y_2) \end{bmatrix} \begin{bmatrix} 1 & -\cos(y_2) \\ e^{-y_1} & 1 \end{bmatrix}$$

(Note that the – sign found in the notes cancels out eventually)

Then

$$\frac{df}{dx_1} = \frac{\partial f}{\partial x_1} + \psi_1 \frac{\partial R_1}{\partial x_1} + \psi_2 \frac{\partial R_2}{\partial x_1}$$

Finite Difference

For the 2 finite-difference cases, the same general derivation shown above was used to obtain the large matrix equation. However, instead of computing each derivative analytically (i.e. $\frac{\partial f}{\partial x_n}, \frac{\partial f}{\partial y_n}, \frac{\partial R_i}{\partial x_n}, \frac{\partial R_i}{\partial y_n}$), the derivatives were obtained by evaluating the function itself, i.e.

1st Order

$$\frac{\partial f}{\partial x_1} = \frac{f(x_1 + h, x_2, y_1, y_2) - f(x_1, x_2, y_1, y_2)}{h}$$

2nd Order

$$\frac{\partial f}{\partial x_1} = \frac{f(x_1 + h, x_2, y_1, y_2) - f(x_1 - h, x_2, y_1, y_2)}{2h}$$

Comparing Derivatives

The derivatives at the initial point (x_1 , x_2) = (-0.1 , -1) were computed using 3 methods: adjoint, finite difference 1st order, and finite difference 2nd order.

Computing df/dx1

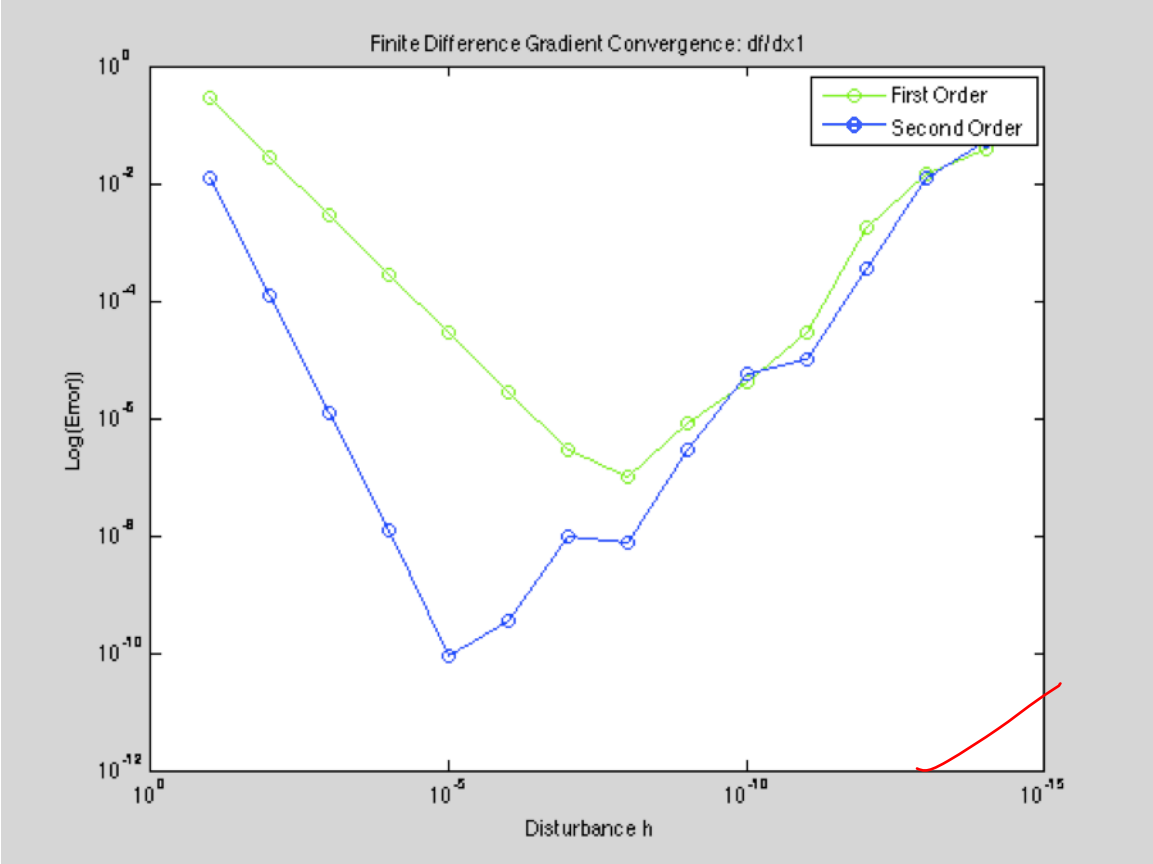
Adjoint

$$\frac{df}{dx_1} = -0.107330726932178$$

Finite Difference

Table 1: Finite Difference df/dx1

Disturbance h	1st Order	2nd Order
10^{-1}	-0.402398113528054	-0.119634645702263
10^{-2}	-0.135252695125478	-0.107454497736981
10^{-3}	-0.110104556213073	-0.107331964713753
10^{-4}	-0.107607923796163	-0.107330739314118
10^{-5}	-0.107358444867239	-0.107330727017862
10^{-6}	-0.107333498885576	-0.107330727293441
10^{-7}	-0.107331006397139	-0.107330717253025
10^{-8}	-0.107330828997226	-0.107330719153013
10^{-9}	-0.107331549733078	-0.107331023989413
10^{-10}	-0.107326556015196	-0.107325009607410
10^{-11}	-0.107360222751202	-0.107320593684495
10^{-12}	-0.109095842544960	-0.107667791836150
10^{-13}	-0.121449811235961	-0.119229365186710
10^{-14}	-0.145455826922180	-0.160938505104399


Finite Difference Gradient Convergence: df/dx1
First Order
Second Order
Log(Error))
Disturbance h
10^0
10^-2
10^-4
10^-6
10^-8
10^-10
10^-12
10^0
10^-5
10^-10
10^-15

Computing df/dx2

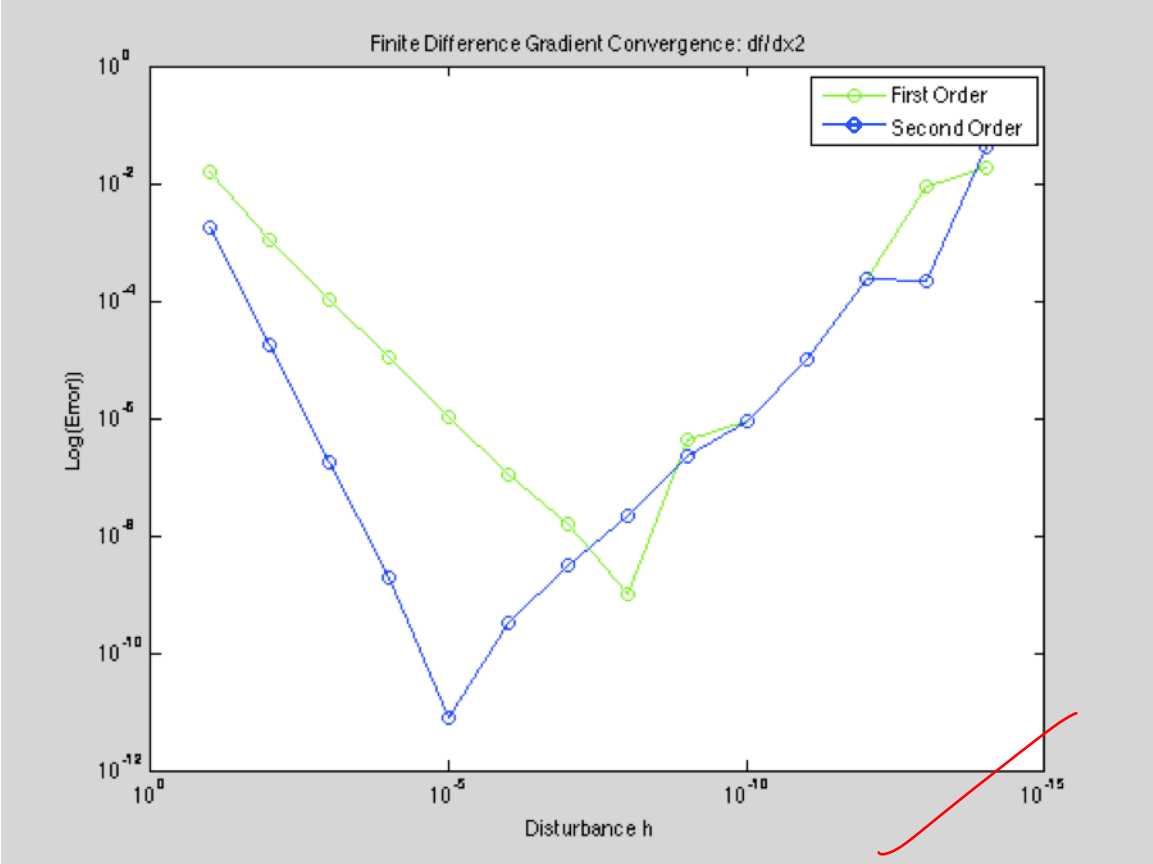
Adjoint

$$\frac{df}{dx_2} = -2.194012970310228$$

Finite Difference

Table 2: Finite Difference df/dx2

Disturbance h	1st Order	2nd Order
10^{-1}	-2.1787481474410**90**	-2.19**2184170659035**
10^{-2}	-2.19**2897499118819**	-2.19**3994686823064**
10^{-3}	-2.19**3905531380776**	-2.19401**2787475774**
10^{-4}	-2.19400**2267484091**	-2.19401**268477210**
10^{-5}	-2.19401**1900504240**	-2.19401297**0302417**
10^{-6}	-2.19401**2862757626**	-2.19401297**0635484**
10^{-7}	-2.19401**2954716136**	-2.19401297**3522064**
10^{-8}	-2.19401297**1301618**	-2.19401294**9097157**
10^{-9}	-2.19401**3415390828**	-2.19401**3193346223**
10^{-10}	-2.19401**3859480037**	-2.19401**3859480037**
10^{-11}	-2.19402**2741264234**	-2.19402**2741264234**
10^{-12}	-2.194**244785869159**	-2.194**244785869159**
10^{-13}	-2.1**84918912462308**	-2.19**3800696659309**
10^{-14}	-2.1**76037128265307**	-2.1**53832667772804**



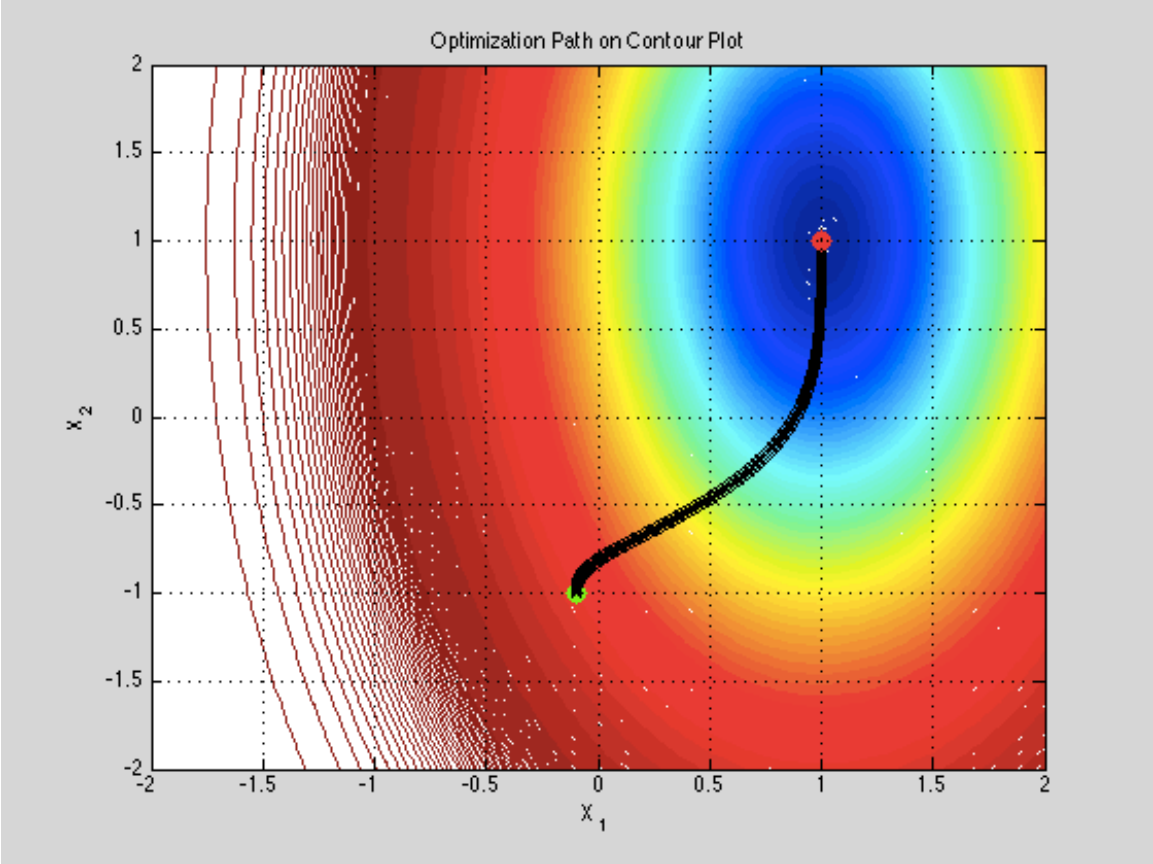
Note that in both convergence graphs, the second order finite difference method converges faster than the first order, as expected. First order convergence rate is $O(h)$, whereas second order convergence is $O(h^2)$. The slope of the second order line (blue) is roughly twice as steep as that of the first order on the log-log plot above for the first section.

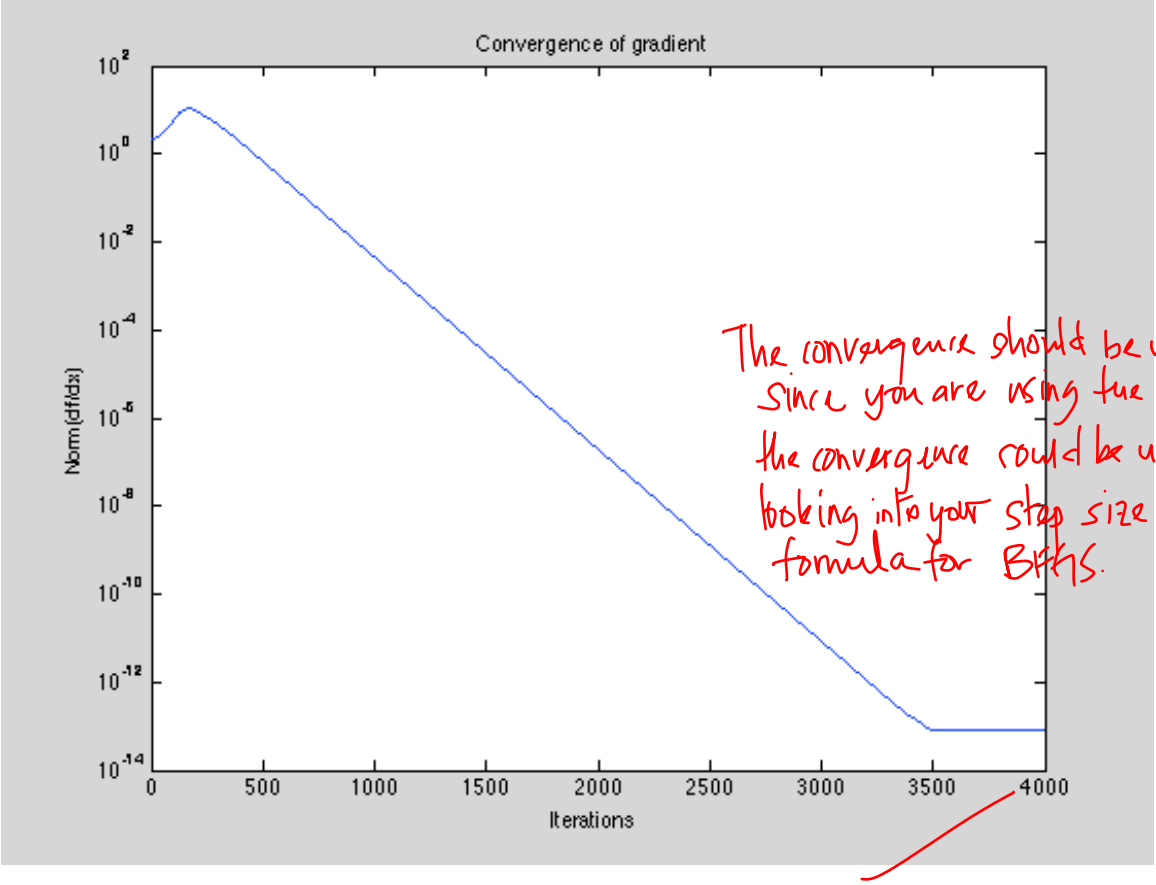
Reducing the disturbance h improves the accuracy of the gradient but only up to a certain point. Reducing h too much causes the difference f(x+h)-f(x) to yield a very small number which is truncated due to the precision of the machine, thereby resulting in an incorrect derivative. The values found using the adjoint method are accurate to 16 digits because they represent the analytical solution

Note that the values used in the y-axis were obtained by subtracting the values found using the finite difference methods (1st or 2nd order) from the adjoint method, taking the absolute value, and taking the logarithm of that error.

Solution

A DFP Quasi-Newton approach with fixed step-length $\alpha = 0.001$ and initial guess $B_k = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ was used. The algorithm is found in page 18 of the Unconstrained Optimization, Part 2 .pdf document (Matlab code included in Appendix).





The convergence should be much faster. Since you are using the exact derivative the convergence could be under 20. Try looking into your step size or the update formula for BFGS. —2

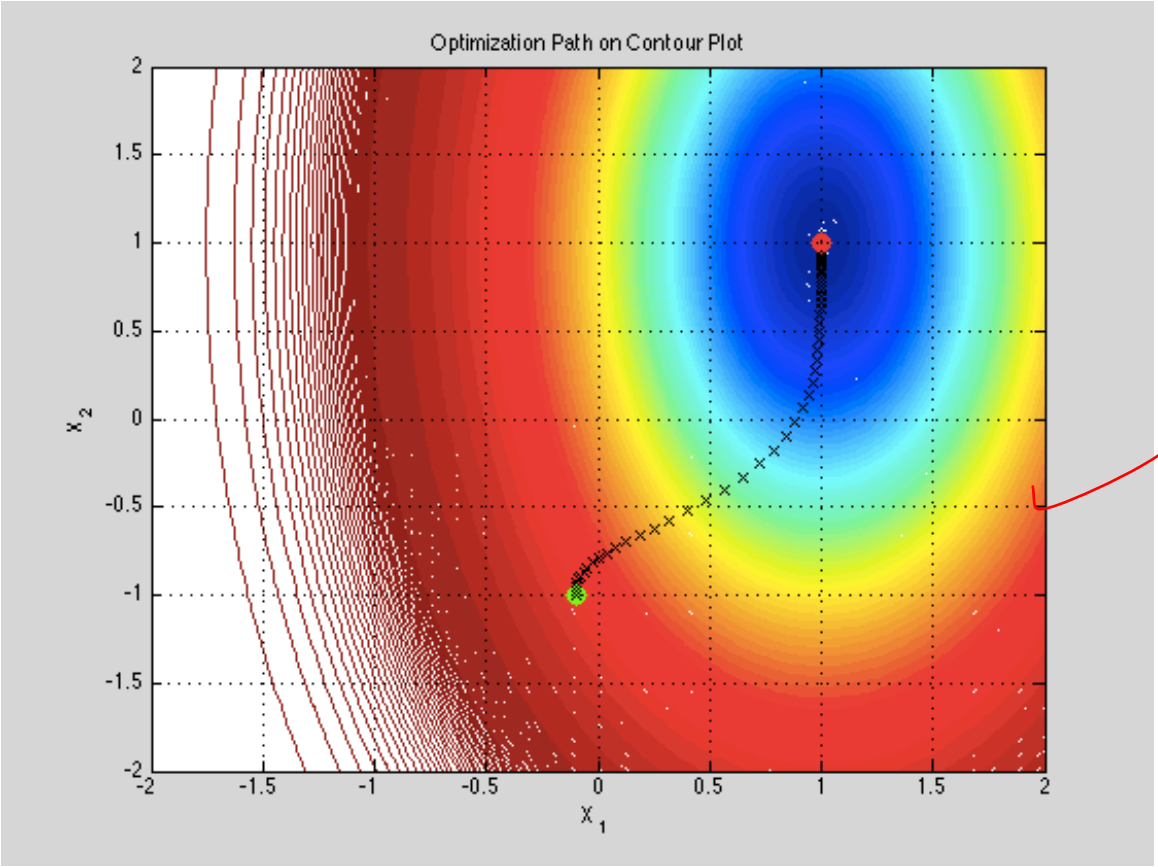
The minimum of (1,1) is reached in approximately 3500 iterations. The gradient used as the convergence criterion was $\begin{bmatrix} \frac{df}{dx_1} \\ \frac{df}{dx_2} \end{bmatrix}$, and upon termination reaches machine zero.

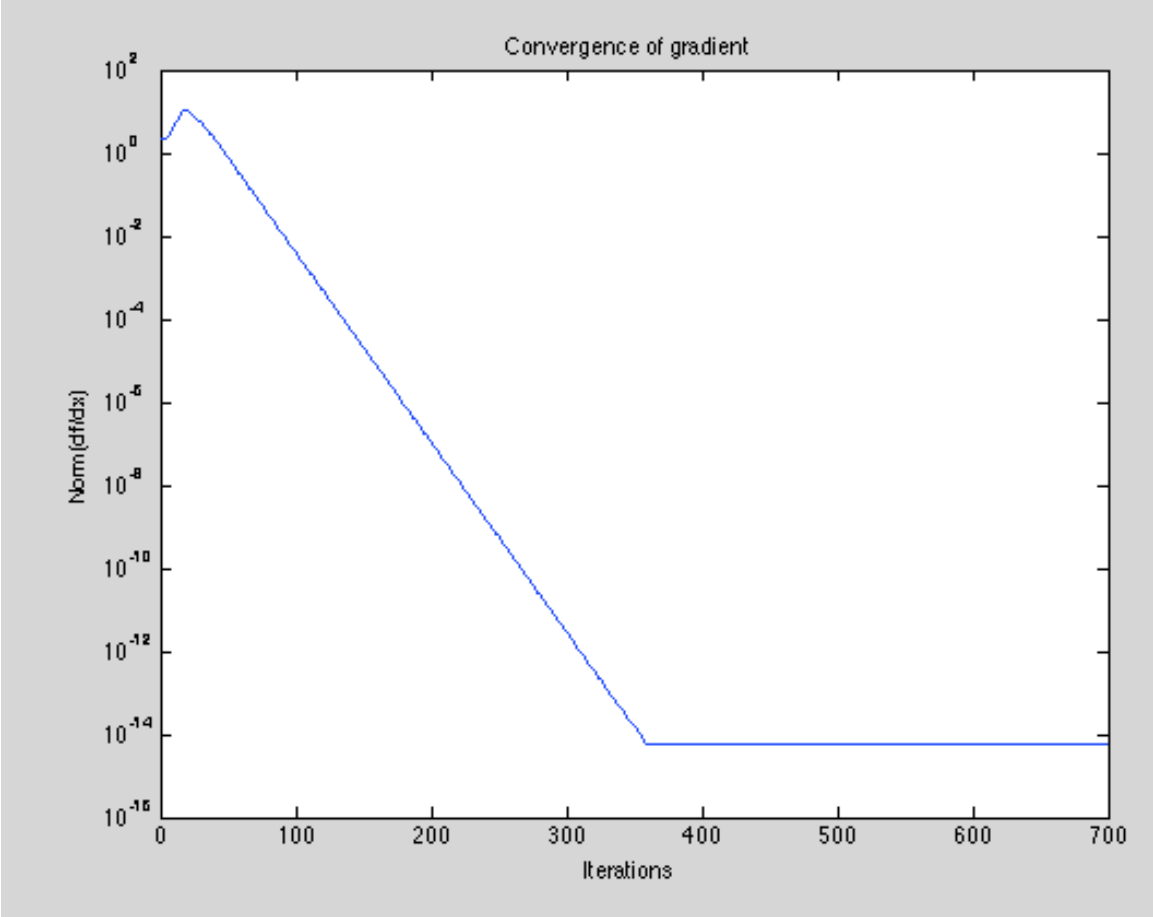
The direct method was used to compute the gradient, and the Quasi-Newton DFP method was used to update the design variables.

Parameter Variation

Changing α

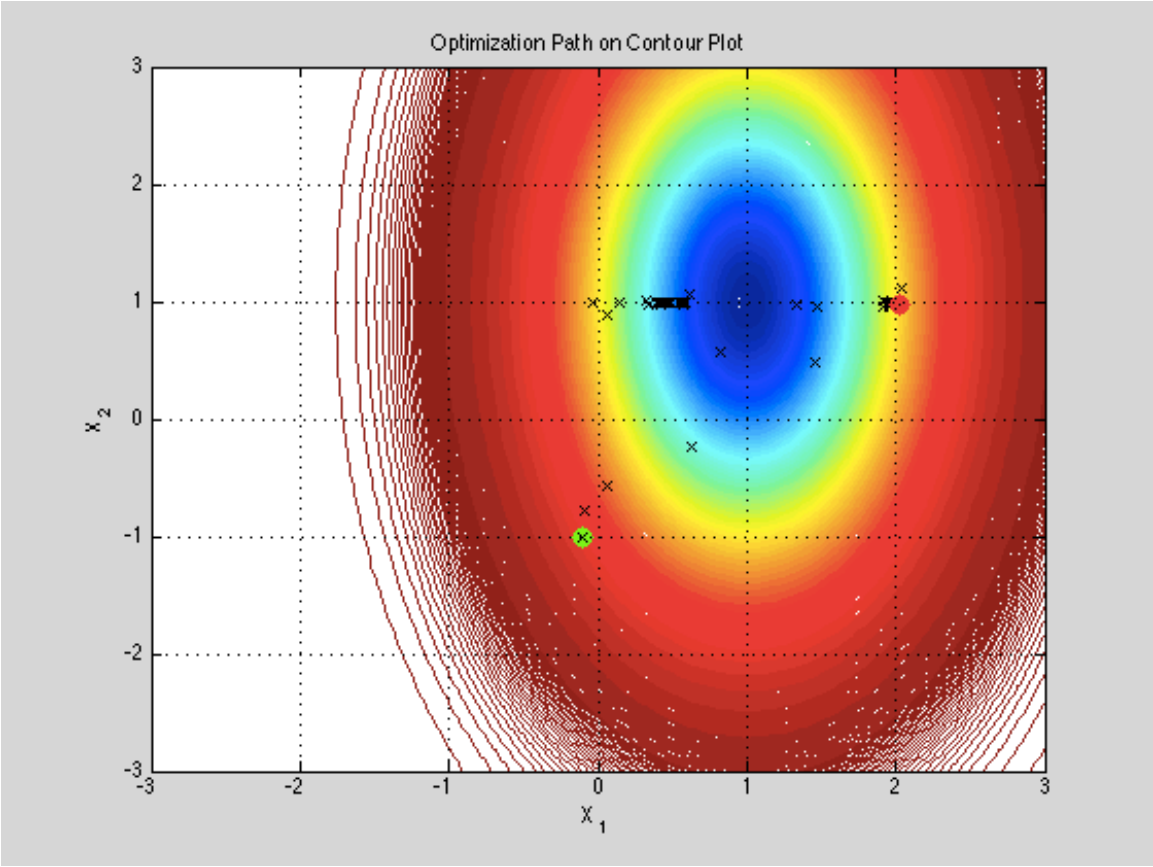
Setting α to 0.01 instead of 0.001

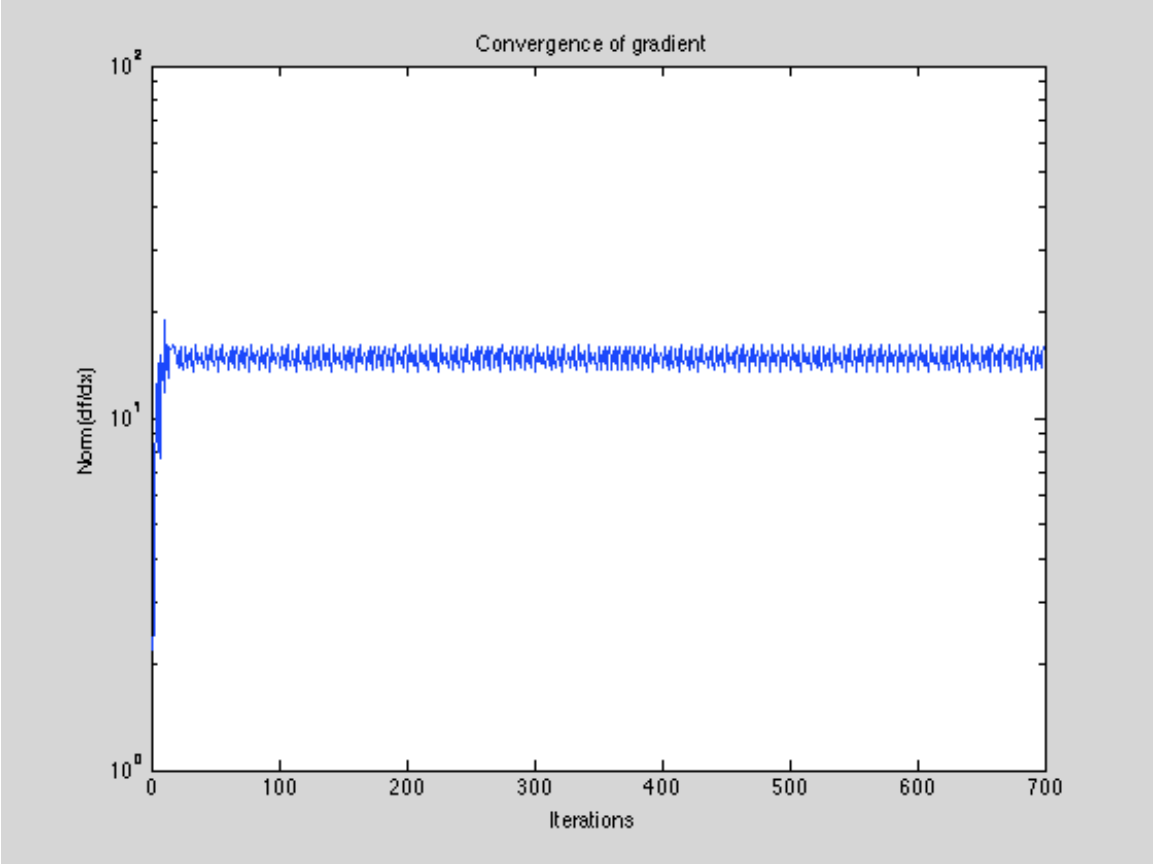




As expected, the gradient converges much faster by increasing the step length. Just as α was increased ten times, the convergence occurred approximately ten times faster, at just over 350 iterations. Also notice how much sparser the intermediate points are on the contour plot.

Setting α to 0.1 instead of 0.001

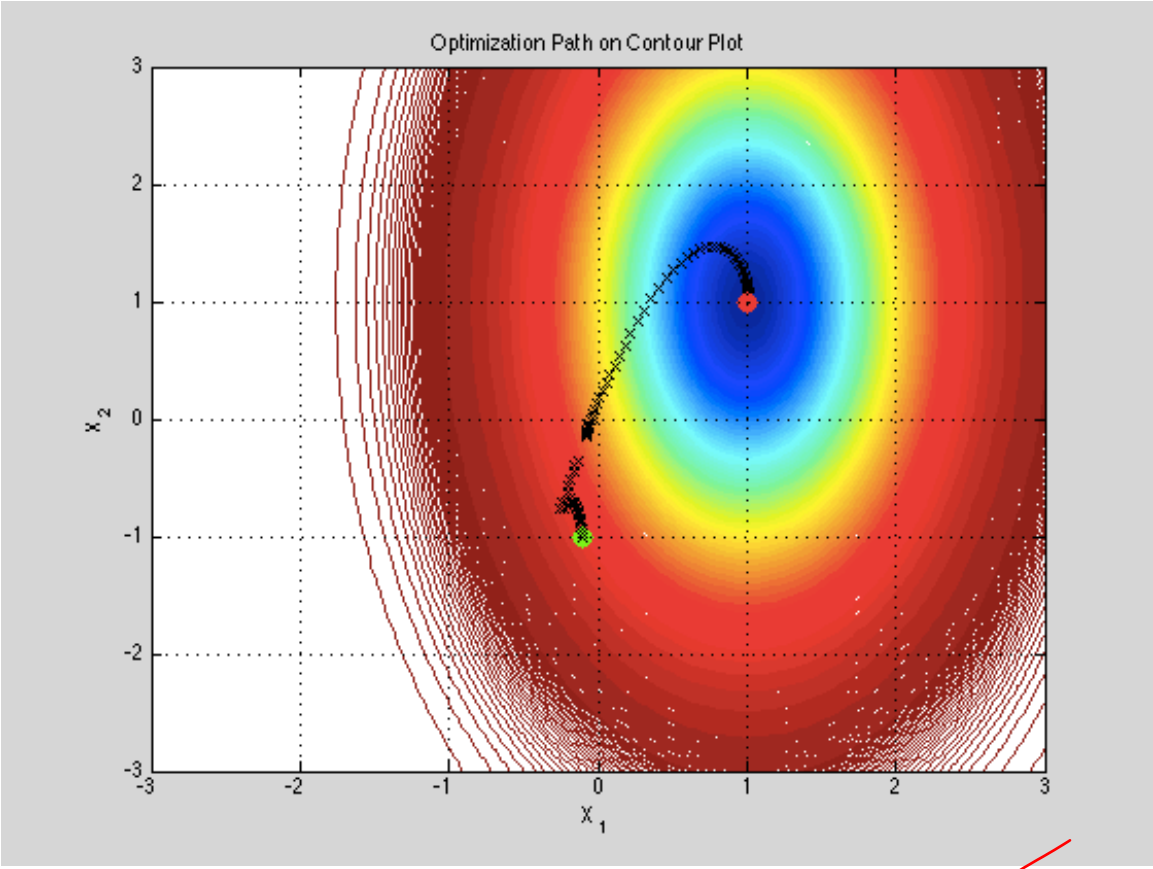


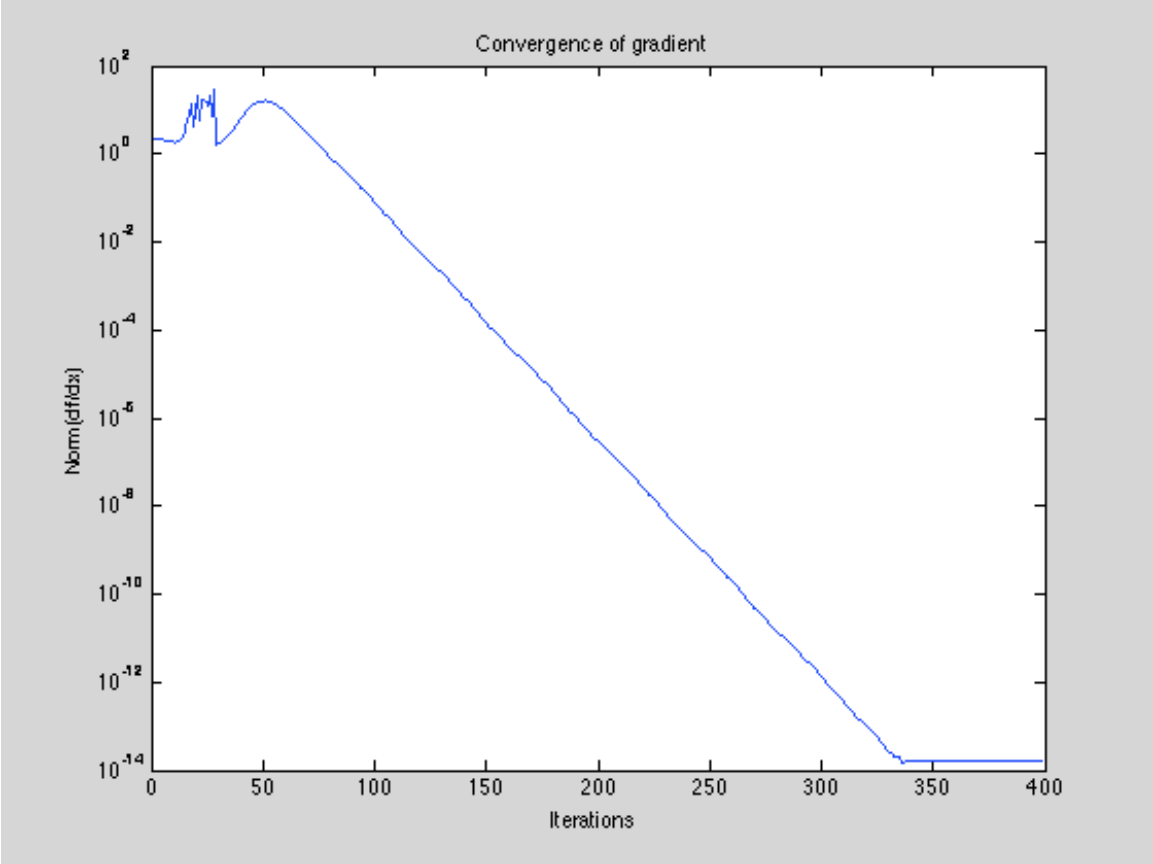


In this case, the value of α was increased too much, to the point where convergence becomes impossible. Notice the cluster of points to the left and to the right of the global minimum of (1,1) on the contour plot. Coupled with the near constant value of the gradient norm, it is clear that the search direction is repeatedly computed correctly, but that the minimum cannot be reached due to the large step size.

Changing B_k

Setting to $B_k = \begin{bmatrix} 3 & -2 \\ 6 & 12 \end{bmatrix}$ instead of $B_k = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$





The pseudo-Hessian approximation dictates the search direction, as can be shown from the optimization path on the contour plot. Convergence is actually much quicker using this search direction than using the identity matrix. The number of iterations to convergence improved from 3500 to fewer than 350.

Conclusion

Using an MDF approach integrated into a DFP Quasi-Newton framework, the global minimum was found to be (x_1 , x_2) = (1 , 1). The number of iterations to convergence improves considerably by judiciously selecting the value of the step length α and the initial value of the pseudo-Hessian B_k.

The derivatives found from the adjoint method are accurate up to 16 digits because they represent the analytical solution, but the finite difference methods (both first and second order), suffer from inaccuracies due to truncation errors when the disturbance is reduced too much.

Appendix

Matlab Code

Main Solver: test5.m

```
clear
clc

size=3;
x=linspace(-size,size);
y=linspace(-size,size);
[X,Y]=meshgrid(x,y);
y1=-0.346765878359960;
y2=1.414485524930436;
Z=y1 + cos(y2) - 20*exp(- (X - 1).^2 - (Y - 1).^2/4);
contour(X,Y,Z,2000)
xlabel('X_1')
ylabel('X_2')
title('Optimization Path on Contour Plot')

x1=-0.1;
x2=-1;
dfdx=feval('compute',x1,x2);
hold on
plot(x1,x2,'go','LineWidth',5)

alpha=0.001;
Hk=eye(2);
Hk=[3 -2;6 12];
i=0;

grid on
while(1)
i=i+1;
if i==400
    plot(x1,x2,'ro','LineWidth',5)
    break
end
bigmat(i)=norm(dfdx);

dk=-Hk*dfdx;
plot(x1,x2,'Color',[0 0 0],'Marker','x')
x1new=x1+alpha*dk(1);
x2new=x2+alpha*dk(2);

dfdxnew=feval('compute',x1new,x2new);

deltaX=alpha*dk;
deltaG=dfdxnew-dfdx;

ranktwo=((deltaX*deltaX')/(deltaX'*deltaG)) -
((Hk*deltaG)*(Hk*deltaG)')/(deltaG'*Hk*deltaG);
Hknew = Hk + ranktwo;
```

```
x1=x1new;
x2=x2new;
dfdx=dfdxnew;

end

hold off
figure
semilogy(bigmat)
xlabel('Iterations')
ylabel('Norm(df/dx)')
title('Convergence of gradient')
```

Derivatives: derivatives.m

```
function [dfdx_first,dfdx_second]=derivatives(h)

y2=1;
x1=-0.1;
x2=-1;
i=0;
while(1)
   i=i+1;
   if i==100
       break
   end
   y1=-3*exp(-(x1+1)^2 + 0.25* (x2+1)^2) + sin(y2);
   y2=-3*exp(-5*(x1-3)^2 + 0.25* (x2-3)^2) + exp(-y1);
end

% FIRST ORDER
dfdx1_first=(feval('testfunction',x1+h,x2,y1,y2)-
feval('testfunction',x1,x2,y1,y2))/h;
dfdx2_first=(feval('testfunction',x1,x2+h,y1,y2)-
feval('testfunction',x1,x2,y1,y2))/h;
dfdy1_first=(feval('testfunction',x1,x2,y1+h,y2)-
feval('testfunction',x1,x2,y1,y2))/h;
dfdy2_first=(feval('testfunction',x1,x2,y1,y2+h)-
feval('testfunction',x1,x2,y1,y2))/h;

dR1x1_first=(feval('governing1',x1+h,x2,y1,y2)-
feval('governing1',x1,x2,y1,y2))/h;
dR1x2_first=(feval('governing1',x1,x2+h,y1,y2)-
feval('governing1',x1,x2,y1,y2))/h;
dR2x1_first=(feval('governing2',x1+h,x2,y1,y2)-
feval('governing2',x1,x2,y1,y2))/h;
dR2x2_first=(feval('governing2',x1,x2+h,y1,y2)-
feval('governing2',x1,x2,y1,y2))/h;

dR1y1_first=(feval('governing1',x1,x2,y1+h,y2)-
feval('governing1',x1,x2,y1,y2))/h;
dR1y2_first=(feval('governing1',x1,x2,y1,y2+h)-
feval('governing1',x1,x2,y1,y2))/h;
dR2y1_first=(feval('governing2',x1,x2,y1+h,y2)-
```

```
feval('governing2',x1,x2,y1,y2))/h;
dR2y2_first=(feval('governing2',x1,x2,y1,y2+h)-
feval('governing2',x1,x2,y1,y2))/h;

Arow1_1 = [dR1y1_first 0 dR1y2_first 0];
Arow2_1 = [0 dR1y1_first 0 dR1y2_first];
Arow3_1 = [dR2y1_first 0 dR2y2_first 0];
Arow4_1 = [0 dR2y1_first 0 dR2y2_first];

A_1 = [Arow1_1;Arow2_1;Arow3_1;Arow4_1];

brow1_1=-dR1x1_first;
brow2_1=-dR1x2_first;
brow3_1=-dR2x1_first;
brow4_1=-dR2x2_first;

b_1 = [brow1_1;brow2_1;brow3_1;brow4_1];

% | y1x1 |
% | y1x2 |
% | y2x1 |
% | y2x2 |

dydx_first=inv(A_1)*b_1;
dfrow1_1=dfdx1_first+dfdy1_first*dydx_first(1)+dfdy2_first*dydx_first(3
);
dfrow2_1=dfdx2_first+dfdy1_first*dydx_first(2)+dfdy2_first*dydx_first(4
);
dfdx_first=[dfrow1_1;dfrow2_1];

% SECOND ORDER
dfdx1_second=(feval('testfunction',x1+h,x2,y1,y2)-
feval('testfunction',x1-h,x2,y1,y2))/(2*h);
dfdx2_second=(feval('testfunction',x1,x2+h,y1,y2)-
feval('testfunction',x1,x2-h,y1,y2))/(2*h);
dfdy1_second=(feval('testfunction',x1,x2,y1+h,y2)-
feval('testfunction',x1,x2,y1-h,y2))/(2*h);
dfdy2_second=(feval('testfunction',x1,x2,y1,y2+h)-
feval('testfunction',x1,x2,y1,y2-h))/(2*h);

dR1x1_second=(feval('governing1',x1+h,x2,y1,y2)-feval('governing1',x1-
h,x2,y1,y2))/(2*h);
dR1x2_second=(feval('governing1',x1,x2+h,y1,y2)-
feval('governing1',x1,x2-h,y1,y2))/(2*h);
dR2x1_second=(feval('governing2',x1+h,x2,y1,y2)-feval('governing2',x1-
h,x2,y1,y2))/(2*h);
dR2x2_second=(feval('governing2',x1,x2+h,y1,y2)-
feval('governing2',x1,x2-h,y1,y2))/(2*h);

dR1y1_second=(feval('governing1',x1,x2,y1+h,y2)-
feval('governing1',x1,x2,y1-h,y2))/(2*h);
dR1y2_second=(feval('governing1',x1,x2,y1,y2+h)-
feval('governing1',x1,x2,y1,y2-h))/(2*h);
dR2y1_second=(feval('governing2',x1,x2,y1+h,y2)-
feval('governing2',x1,x2,y1-h,y2))/(2*h);
dR2y2_second=(feval('governing2',x1,x2,y1,y2+h)-
```

```
feval('governing2',x1,x2,y1,y2-h))/(2*h);


Arow1_2 = [dR1y1_second 0 dR1y2_second 0];
Arow2_2 = [0 dR1y1_second 0 dR1y2_second];
Arow3_2 = [dR2y1_second 0 dR2y2_second 0];
Arow4_2 = [0 dR2y1_second 0 dR2y2_second];

A_2 = [Arow1_2;Arow2_2;Arow3_2;Arow4_2];

brow1_2=-dR1x1_second;
brow2_2=-dR1x2_second;
brow3_2=-dR2x1_second;
brow4_2=-dR2x2_second;

b_2 = [brow1_2;brow2_2;brow3_2;brow4_2];

% | y1x1 |
% | y1x2 |
% | y2x1 |
% | y2x2 |

dydx_second=inv(A_2)*b_2;
dfrow1_2=dfdx1_second+dfdy1_second*dydx_second(1)+dfdy2_second*dydx_sec
ond(3);
dfrow2_2=dfdx2_second+dfdy1_second*dydx_second(2)+dfdy2_second*dydx_sec
ond(4);
dfdx_second=[dfrow1_2;dfrow2_2];

end
```

Comparing derivatives – compare.m

```
clear
clc

format long
i=0;
derr=zeros(14,2);
purp=zeros(14);
while(1)
    i=i+1;
    if i==15
        break
    end
    h=10^(-i);
    purp(i)=h;
    [dfdx_first,dfdx_second]=derivatives(h);
    derr(i,1)=dfdx_first(2);
    derr(i,2)=dfdx_second(2);

end

    h=0.0000001;
```

```
    [dfdx_first,dfdx_second]=derivatives(h);
    dfdx_first;
    dfdx_second;
% direct/adjoint:
dfdx_direct=feval('compute',-0.1,-1);
derr;

i=0;
while(1)
    i=i+1;
    if i==15
        break
    end

derr(i,1)=derr(i,1)-dfdx_direct(2);
derr(i,2)=derr(i,2)-dfdx_direct(2);
end

derr=abs(derr);

h1=loglog(purp,derr(:,1),'Color','green','Marker','o');
hold on
h2=loglog(purp,derr(:,2),'Color','blue','Marker','o');
hold off
xlabel('Disturbance h')
ylabel('Log(Error))')
title('Finite Difference Gradient Convergence: df/dx2')
set(gca, 'XDir', 'reverse')
legend('First Order','Second Order')
```