MECH 579
Multidisciplinary Design Optimization
Siva Nadarajah

Project 1: Unconstrained Optimization

16/20

Dritan Harizaj
260426327

September 26, 2013

Table of Contents

Introduction

The purpose of this project is to minimize the Rosenbrock function using 4 different unconstrained optimization algorithms: Steepest Descent, Nonlinear Conjugate Gradient, Newton, and Quasi-Newton. The Fletcher-Reeves modification was chosen for the Conjugate Gradient method, and the DFP modification was chosen for the Quasi-Newton method. A fixed step-size was used in all 4 cases in order to better control the convergence rate. Backtracking was only introduced to the Conjugate Gradient method for comparison with the fixed step-size.

Backtracking was supposed to be used with all the cases.

(-3)

The starting point in all 4 cases was set to [x , y] = [3 , 4], while the tolerance was kept at $\varepsilon = 10^{-2}$.

Steepest Descent Algorithm

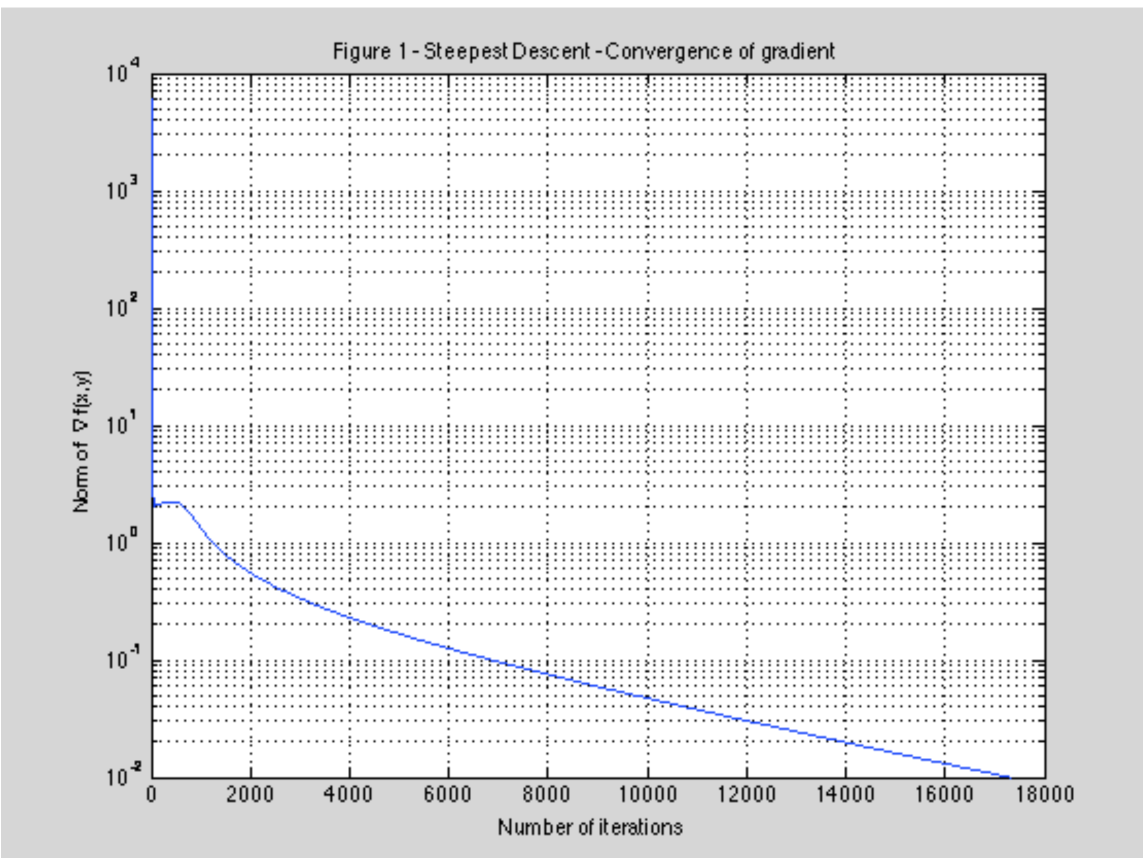
For this algorithm, $\alpha = 0.0005$

Iterations to convergence = 17315

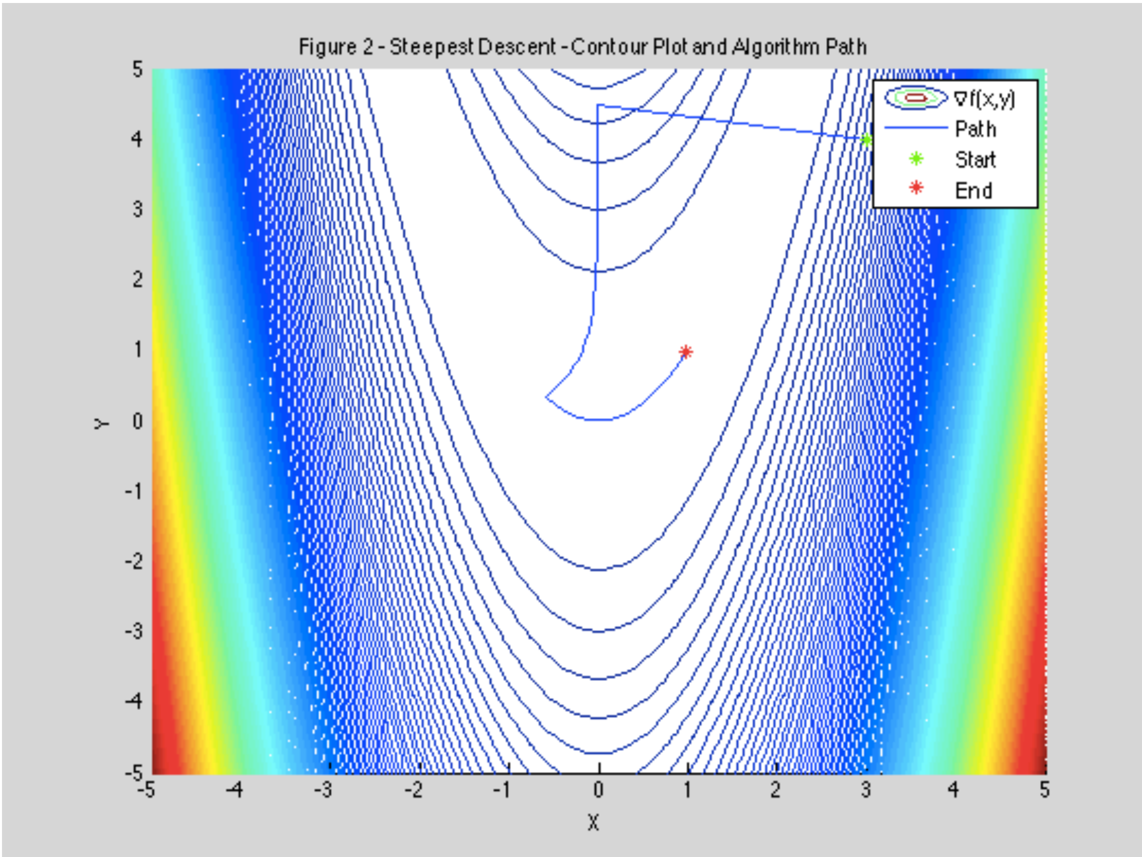
Final value [x , y] = [0.9889 , 0.9779]

$\nabla f(x,y)$ at convergence = 0.0100

Convergence of gradient



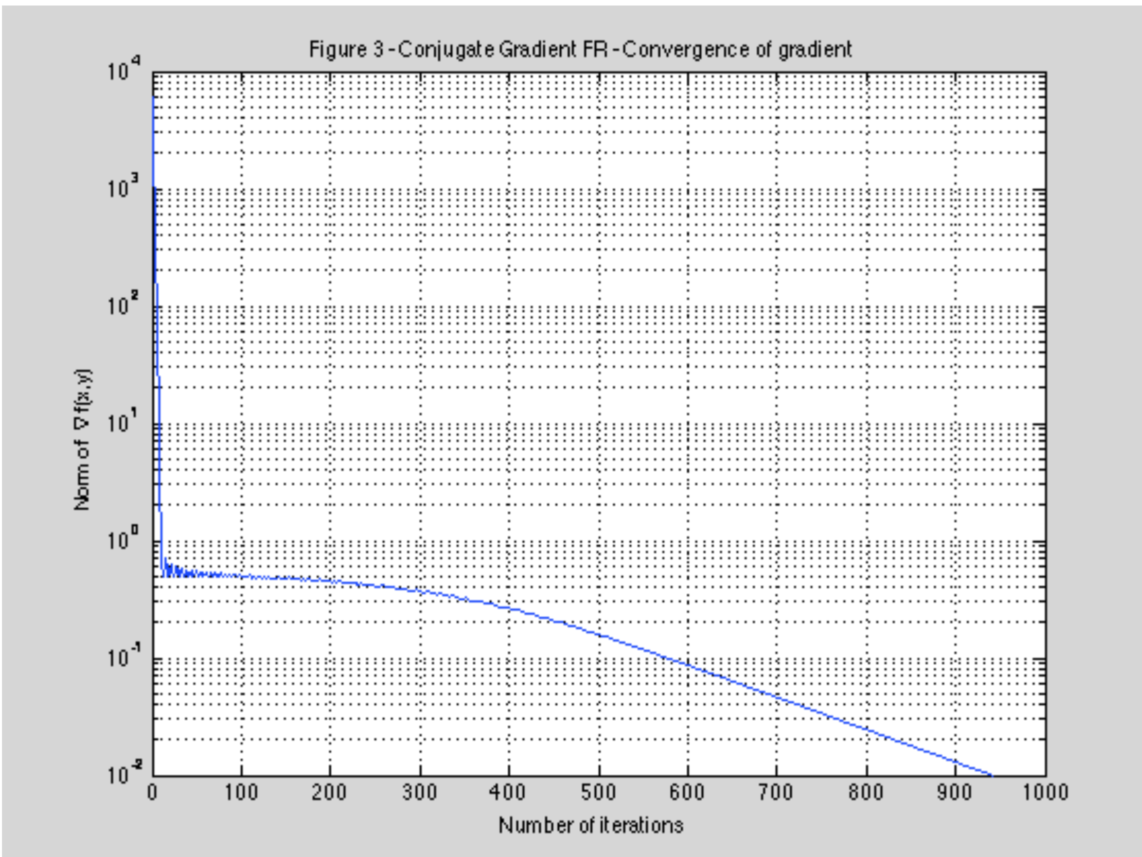
Contour Plot



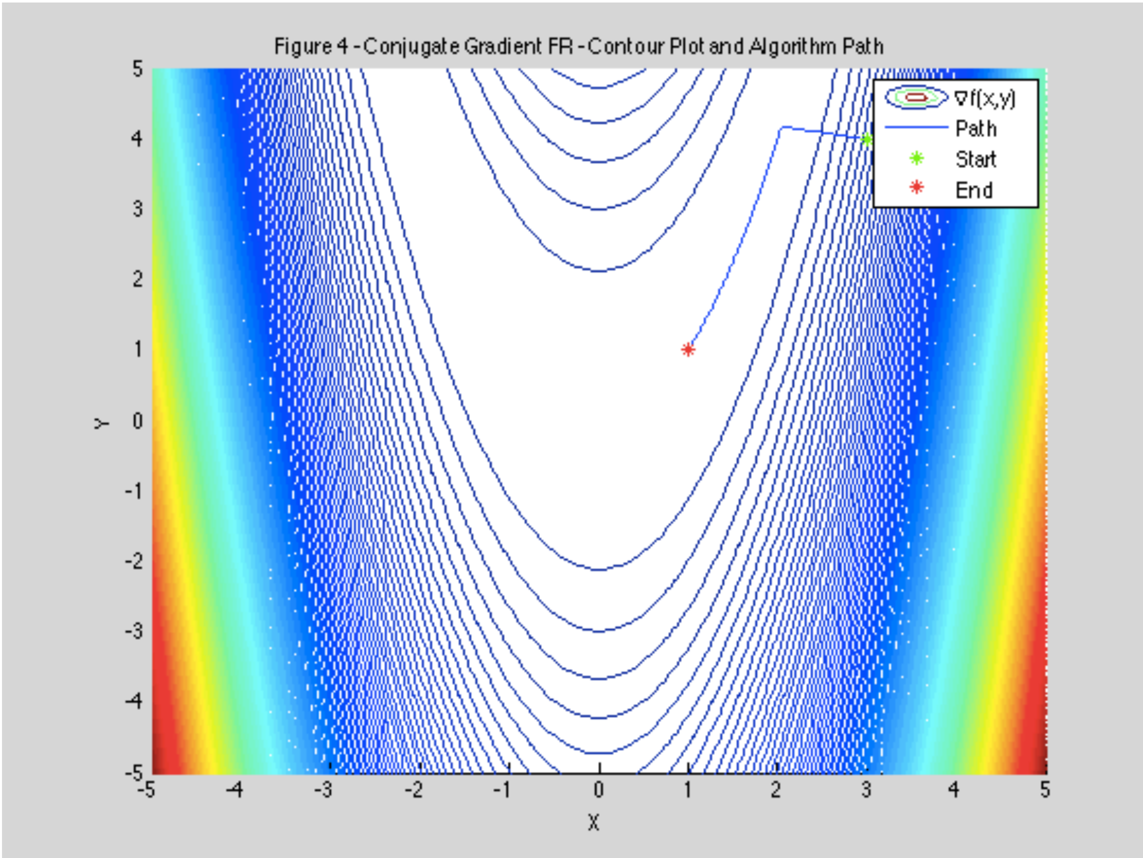
Conjugate Gradient (Fletcher-Reeves) Algorithm

For this algorithm, $\alpha = 0.001$
Iterations to convergence = 942
Final value [x , y] = [1.0112 , 1.0227]
$\nabla f(x,y)$ at convergence = 0.0100

Convergence of gradient



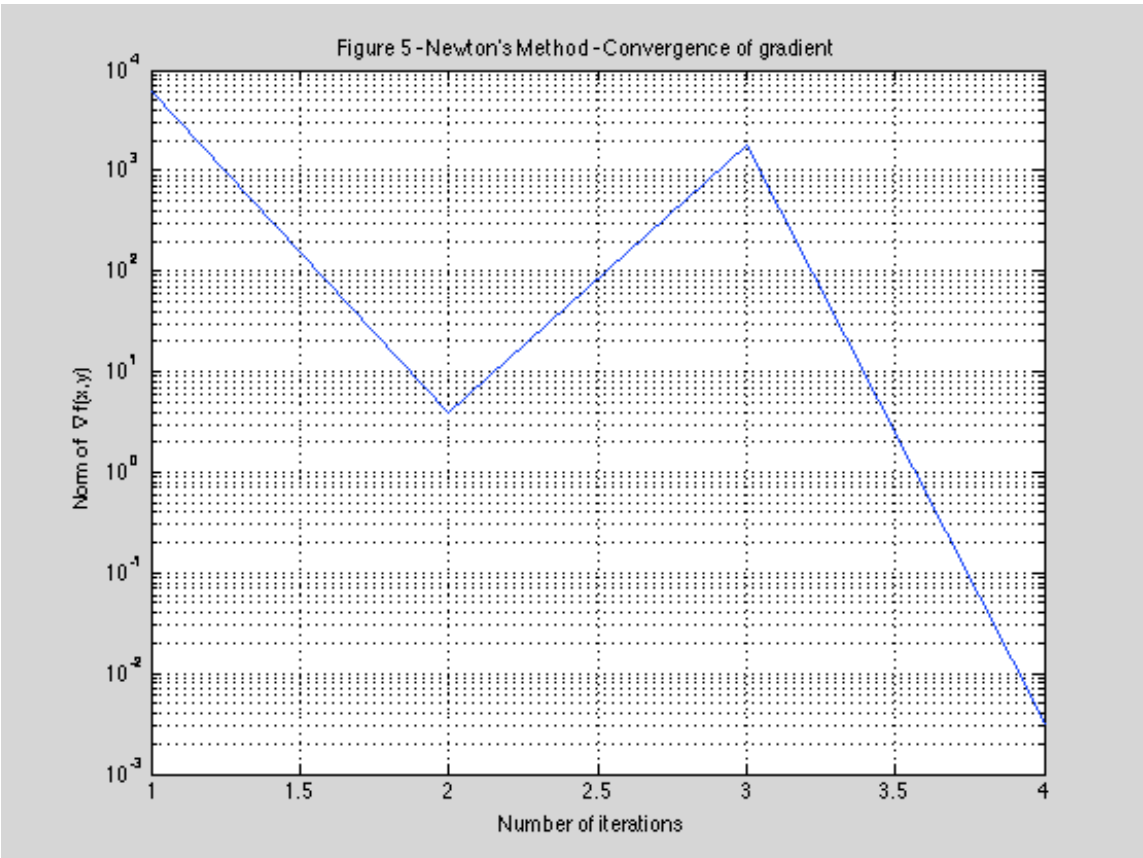
Contour Plot



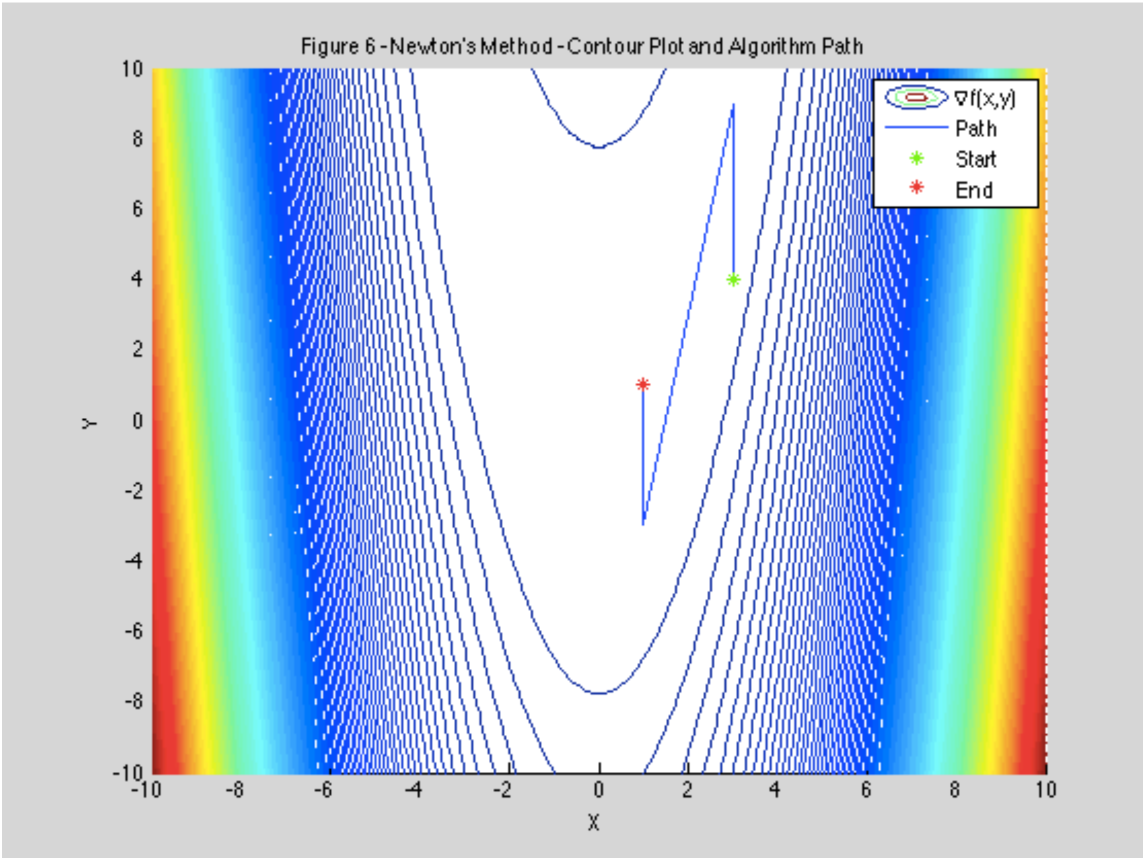
Newton's Method Algorithm

For this algorithm, $\alpha = 0.0005$
Iterations to convergence = 4
Final value [x , y] = [1.0016 , 1.0032]
$\nabla f(x,y)$ at convergence = 0.0032

Convergence of gradient



Contour Plot

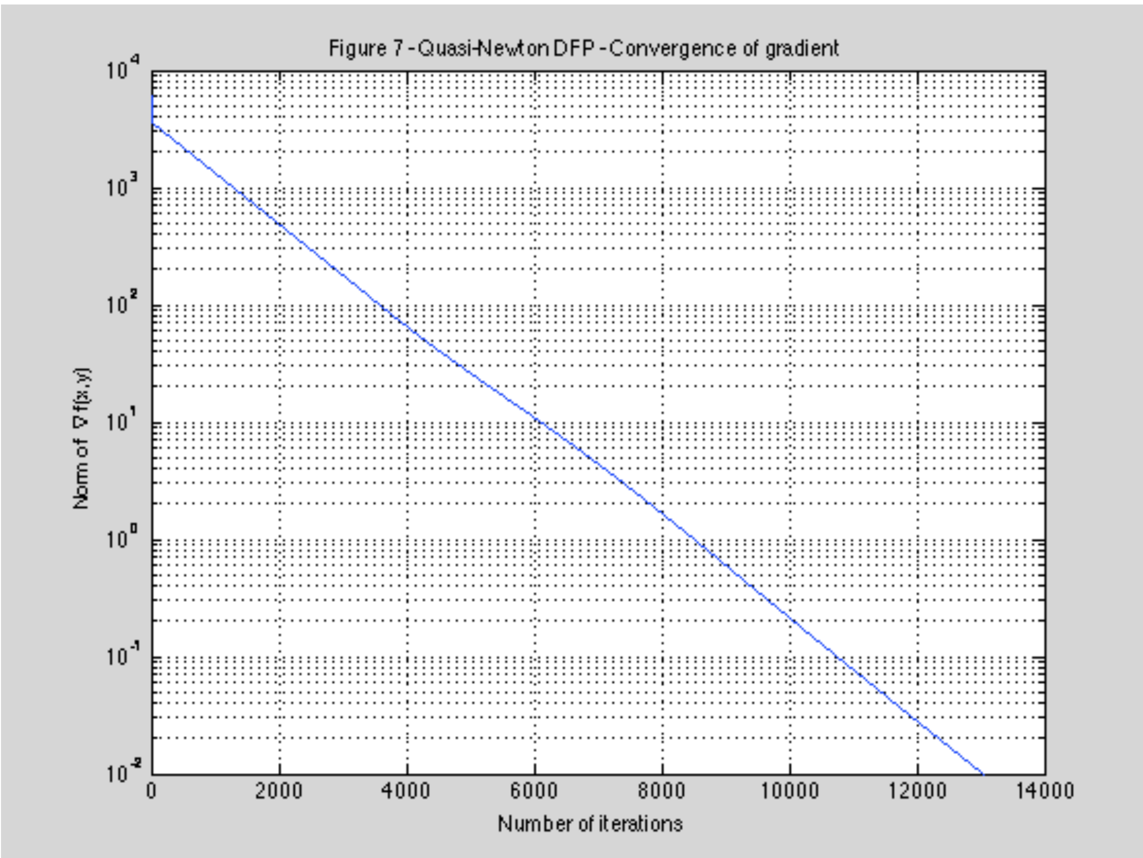


Quasi-Newton (DFP) Algorithm

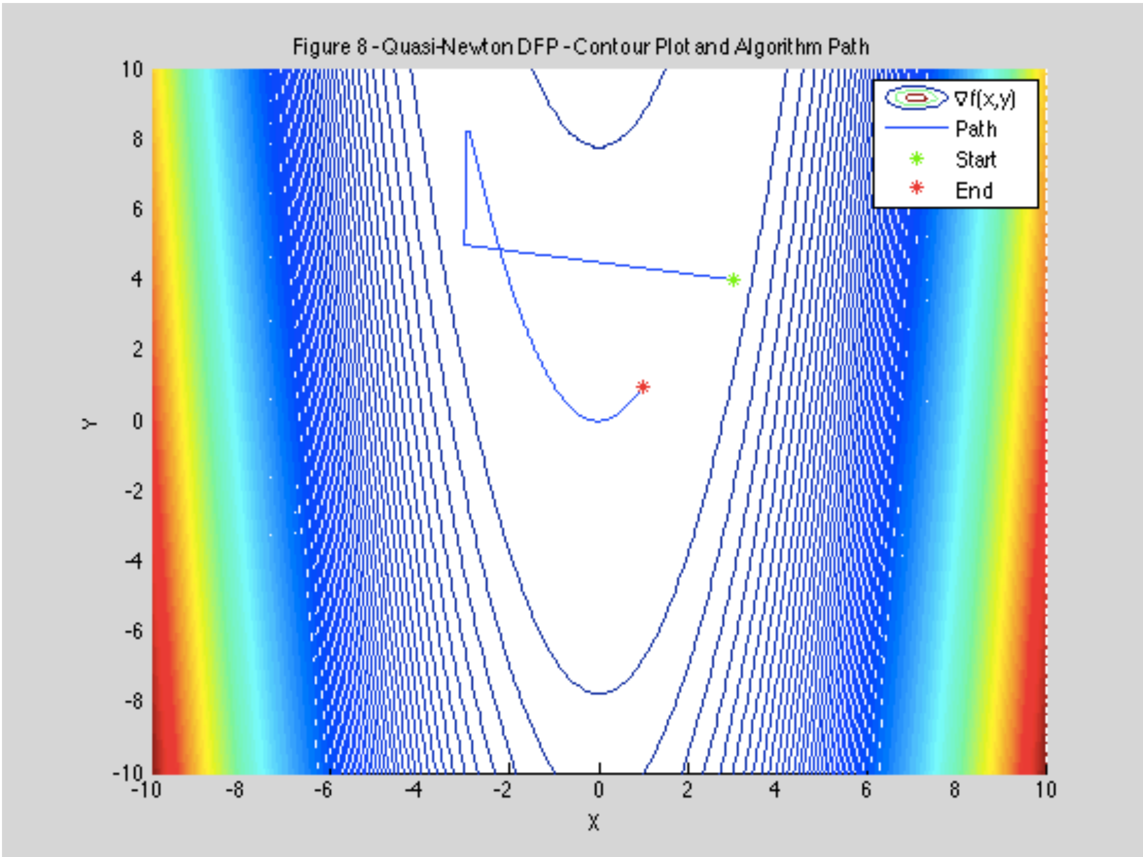
For this algorithm, $\alpha = 0.001$
Iterations to convergence = 13051
Final value [x , y] = [0.9918 , 0.9836]
$\nabla f(x,y)$ at convergence = 0.0100

Quasi-Newton would have been a lot faster, but since you did not use backtracking then there it would be difficult to find the source of the problem.

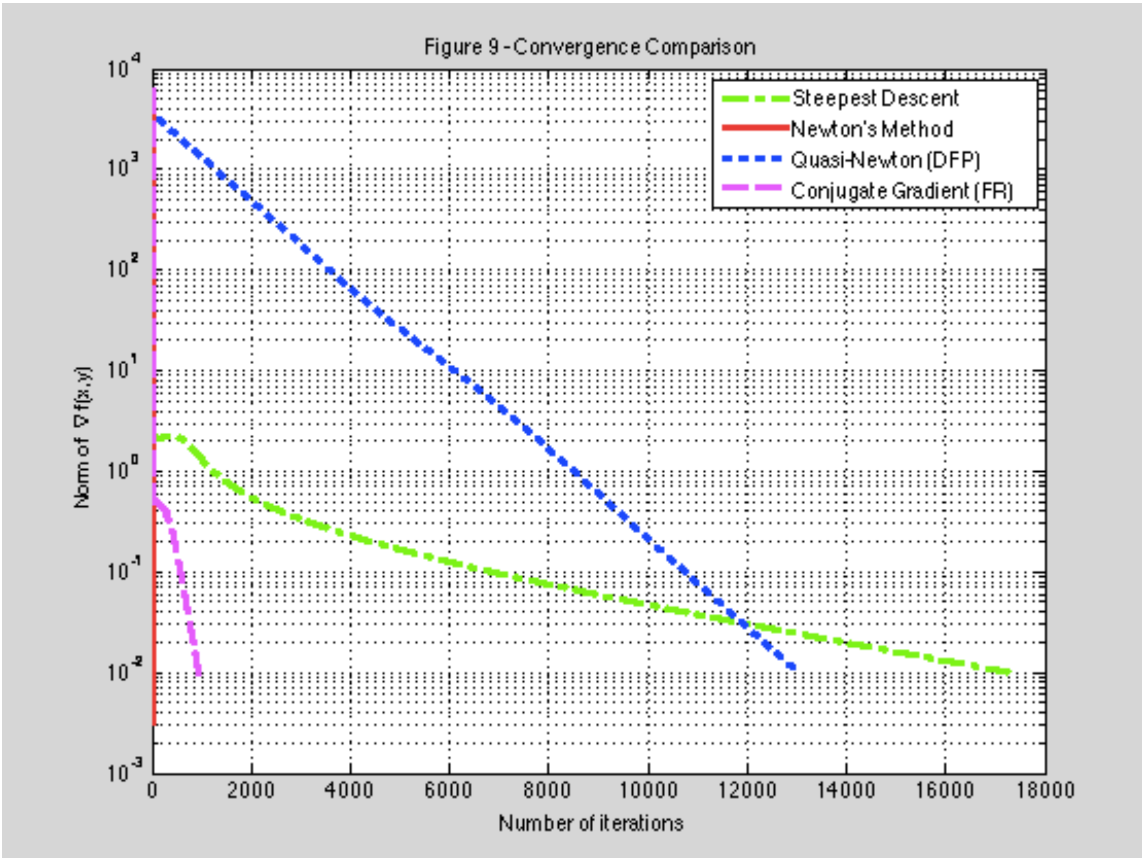
Convergence of gradient



Contour Plot

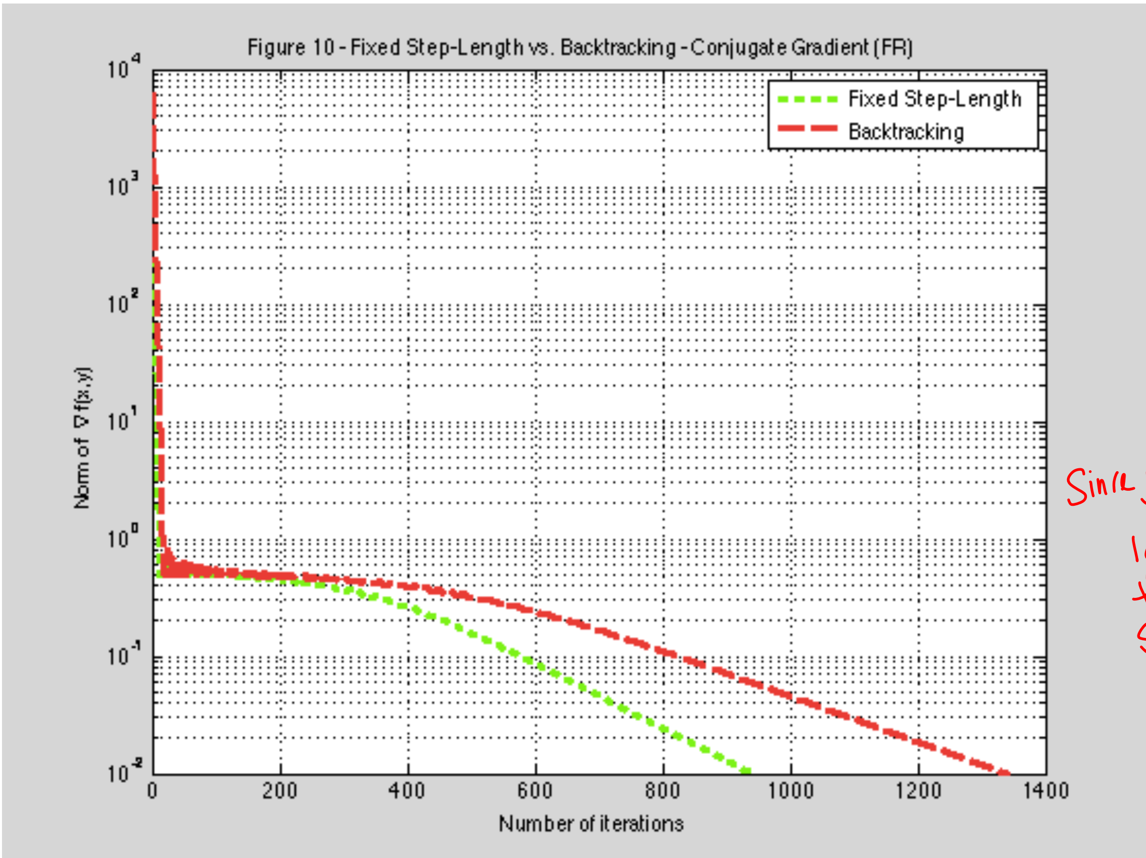


Comparison of Convergences



Newton's method is clearly the most efficient algorithm, although it does not guarantee convergence if the starting point is too far from the answer. The other 3 algorithms require orders of magnitude more iterations than Newton's method. The Steepest Descent method is the slowest to converge, but on the other hand is quite reliable after the bump near the beginning. The Quasi-Newton method ~~converges at a logarithmic pace~~, as demonstrated by the straight line it forms in the semi-log graph. Although for this problem it underperforms Newton's method, for larger problems the Quasi-Newton method would more likely have the advantage because it does not require the Hessian to be computed directly. The Conjugate Gradient method converges quite quickly and is reliable; just like Quasi-Newton techniques, this method scales well with size due to not requiring the Hessian to be computed.

Comparison of Backtracking vs. Fixed Step-Size



Since you are starting from a larger value then backtracking should be faster.

The Conjugate Gradient Method was used to perform the comparison between fixed step-length and backtracking. In both cases, the initial value of $\alpha = 10^{-4}$. For the backtracking case, $c = 10^{-4}$ and $\rho = 0.5$. Did you try a larger value.

The fixed step-size approach converges in 942 iterations, whereas the backtracking approach converges in 1341 iterations

The final value of α for the backtracking case was $5 * 10^{-5}$, which implies that only one backtracking iteration was necessary to ensure that α was small enough. Thus, using a value of α carefully chosen through trial and error yields a faster convergence than using backtracking. In general, backtracking tends to lower α too much, which is ironic because the purpose of backtracking is to prevent precisely that from happening.

Appendix – Matlab Code

1. Steepest Descent

```
THEX = 0;
THEY = 0;
CONV = 0;
x = 1;
y = 1;

k=0;
limit = 100000;
tol = 0.01;
alpha = 0.0005;
theC = 10^(-4);
rho = 0.9;

x = 3;
y = 4;
    p_k = [400*x*(- x^2 + y) - 2*x + 2;  200*x^2 - 200*y];

while(1)
    epsilon = norm([2*x - 400*x*(- x^2 + y) - 2 ; - 200*x^2 + 200*y]);

    k=k+1;

    CONV(k) = epsilon;
    THEX(k) = x;
    THEY(k) = y;

    if (epsilon < tol)
        break;
    end

    if (k==limit)
        break;
    end

    x_old = x;
    y_old = y;

    x = x_old + alpha * p_k(1);
    y = y_old + alpha * p_k(2);
  p_k = [400*x*(- x^2 + y) - 2*x + 2;  200*x^2 - 200*y];

end

semilogy(CONV)
xlabel('Number of iterations')
ylabel('Norm of \nablaf(x,y)')
title('Steepest Descent - Convergence of Algorithm')
grid on
size = 5;
a = linspace(-size,size);
b = linspace(-size,size);
[A,B] = meshgrid(a,b);
```

```
C = (1-A).^2+100*((B-(A.^2)).^2);
levels = 100:100:100;
figure
hold on
contour(A,B,C,200)
plot(THEX,THEY)
plot(3,4,'g*')
plot(x,y,'r*')
legend('\nablaf(x,y)','Path','Start','End')
xlabel('X')
ylabel('Y')
title('Steepest Descent - Contour Plot and Algorithm Path')

dlmwrite('st_CONV.txt',CONV);
```

2.a. Conjugate Gradient – Fletcher-Reeves: Fixed Step-Size

```
THEX = 0;
THEY = 0;
CONV = 0;
x = 1;
y = 1;

k=0;
limit = 100000;
tol = 0.01;
alpha = 0.0001;
theC = 10^(-4);
rho = 0.5;
x_old = 0;
y_old = 0;

x = 3;
y = 4;
p_k  = [400*x*(- x^2 + y) - 2*x + 2;  200*x^2 - 200*y];

while(1)
    epsilon = norm([2*x - 400*x*(- x^2 + y) - 2 ; - 200*x^2 + 200*y]);

    k=k+1;

    CONV(k) = epsilon;
    THEX(k) = x;
    THEY(k) = y;

    if (epsilon < tol)
        break;
    end

    if (k==limit)
        break;
    end

    x_old = x;
    y_old = y;
```

```
    x = x_old + alpha * p_k(1);
    y = y_old + alpha * p_k(2);
    r_k = [2*x_old - 400*x_old*(- x_old^2 + y_old) - 2 ; - 200*x_old^2
+ 200*y_old];
    r_k1 = [2*x - 400*x*(- x^2 + y) - 2 ; - 200*x^2 + 200*y];

    % Fletcher-Reeves definition
    sigma = (r_k1'*r_k1)/(r_k'*r_k);

    p_k_old = p_k;
    p_k = -r_k1 + sigma*p_k_old;

end

semilogy(CONV)
xlabel('Number of iterations')
ylabel('Norm of \nablaf(x,y)')
title('Conjugate Gradient FR - Convergence of Algorithm')
grid on
size = 5;
a = linspace(-size,size);
b = linspace(-size,size);
[A,B] = meshgrid(a,b);
C = (1-A).^2+100*((B-(A.^2)).^2);
levels = 100:100:100;
figure
hold on
contour(A,B,C,200)
plot(THEX,THEY)
plot(3,4,'g*')
plot(x,y,'r*')
legend('\nablaf(x,y)','Path','Start','End')
xlabel('X')
ylabel('Y')
title('Conjugate Gradient FR - Contour Plot and Algorithm Path')

dlmwrite('cg_CONV.txt',CONV);
```

2.b. Conjugate Gradient – Fletcher-Reeves: Backtracking

```
THEX = 0;
THEY = 0;
CONV = 0;
x = 1;
y = 1;

k=0;
limit = 100000;
tol = 0.01;
alpha = 0.0001;
theC = 10^(-4);
rho = 0.5;

x = 3;
y = 4;
```

```
p_k  = [400*x*(- x^2 + y) - 2*x + 2;  200*x^2 - 200*y];

while(1)
    epsilon = norm([2*x - 400*x*(- x^2 + y) - 2 ; - 200*x^2 + 200*y]);

    k=k+1;

    CONV(k) = epsilon;
    THEX(k) = x;
    THEY(k) = y;

    if (epsilon < tol)
        break;
    end

    if (k==limit)
        break;
    end

    % Backtracking Start
    f_new = (1-x)^2+100*((y-(x^2))^2);
    f_old = (1-x_old)^2+100*((y_old-(x_old^2))^2);
    grad_old = [2*x_old - 400*x_old*(- x_old^2 + y_old) - 2 ; -
200*x_old^2 + 200*y_old];
    p_k_old = [400*x_old*(- x_old^2 + y_old) - 2*x_old + 2;
200*x_old^2 - 200*y_old];
    RHS = f_old + theC * alpha * (grad_old' * p_k_old);

    tempx = x;
    tempy = y;
    while (f_new > RHS)
        alpha = alpha * rho;
        tempx = x_old + alpha * p_k_old(1);
        tempy = y_old + alpha * p_k_old(2);
        f_new = (1-tempx)^2+100*((tempy-(tempx^2))^2);
    end
    % Backtracking End

    x_old = x;
    y_old = y;

    x = x_old + alpha * p_k(1);
    y = y_old + alpha * p_k(2);
    r_k = [2*x_old - 400*x_old*(- x_old^2 + y_old) - 2 ; - 200*x_old^2
+ 200*y_old];
    r_k1 = [2*x - 400*x*(- x^2 + y) - 2 ; - 200*x^2 + 200*y];

    % Fletcher-Reeves definition
    sigma = (r_k1'*r_k1)/(r_k'*r_k);

    p_k_old = p_k;
    p_k = -r_k1 + sigma*p_k_old;

end

dlmwrite('cg_CONV_back.txt',CONV);
```

3. Newton's Method

```
THEX = 0;
THEY = 0;
CONV = 0;
syms x y;
syms p_k;
syms grad;
f(x,y) = (1-x)^2+100*((y-(x^2))^2);
grad(x,y) = gradient(f);
hess(x,y) = hessian(f);
p_k(x,y) = -gradient(f);

i=0;
limit = 10;
tol = 0.01;
alpha = 0.0005;
theC = 10^(-4);
rho = 0.9;

x_old = 3;
y_old = 4;
x = 3;
y = 4;

while(1)
    epsilon = norm(double(grad(x,y)));

    i=i+1;

    CONV(i) = epsilon;
    THEX(i) = x;
    THEY(i) = y;

    if epsilon < tol
        break;
    end

    if (i==limit)
        break;
    end

    x_old = x;
    y_old = y;

    delta = -inv(hess(x,y))*grad(x,y);
    delta = double(delta);
    x = x_old + delta(1);
    y = y_old + delta(2);
end

semilogy(CONV)
xlabel('Number of iterations')
ylabel('Norm of \nablaf(x,y)')
```

```
title('Newton''s Method - Convergence of Algorithm')
grid on
size = 10;
a = linspace(-size,size);
b = linspace(-size,size);
[A,B] = meshgrid(a,b);
C = (1-A).^2+100*((B-(A.^2)).^2);
levels = 100:100:100;
figure
hold on
contour(A,B,C,200)
plot(THEX,THEY)
plot(3,4,'g*')
plot(x,y,'r*')
legend('\nablaf(x,y)','Path','Start','End')
xlabel('X')
ylabel('Y')
title('Newton''s Method - Contour Plot and Algorithm Path')

dlmwrite('ne_CONV.txt',CONV);
```

4. Quasi-Newton – DFP

```
THEX = 0;
THEY = 0;
CONV = 0;

k=0;
limit = 100000;
tol = 0.01;
alpha = 0.001;
theC = 10^(-4);
rho = 0.9;

H_new = eye(2);
x = 3;
y = 4;

while(1)
    epsilon = norm([2*x - 400*x*(- x^2 + y) - 2 ; - 200*x^2 + 200*y]);

    k=k+1;

    CONV(k) = epsilon;
    THEX(k) = x;
    THEY(k) = y;

    if (epsilon < tol)
        break;
    end

    if (k==limit)
        break;
    end
```

```
    x_old = x;
    y_old = y;
    H_old = H_new;

    temp = -H_old * [2*x - 400*x*(- x^2 + y) - 2 ; - 200*x^2 + 200*y];
    x = x_old + alpha * temp(1);
    y = y_old + alpha * temp(2);


    deltaX = alpha * temp;
    %deltaG = grad(x,y) - grad(x_old,y_old);
    deltaG= [2*x - 400*x*(- x^2 + y) - 2 ; - 200*x^2 + 200*y] - ...
        [2*x_old - 400*x_old*(- x_old^2 + y_old) - 2 ; - 200*x_old^2 +
200*y_old];

    H_new = H_old + ((deltaX * deltaX')/(deltaX' * deltaG)) - ...
        (H_old * deltaG * deltaG' * H_old )/(deltaG' * H_old * deltaG);


end

semilogy(CONV)
xlabel('Number of iterations')
ylabel('Norm of \nablaf(x,y)')
title('Quasi-Newton DFP - Convergence of Algorithm')
grid on
size = 10;
a = linspace(-size,size);
b = linspace(-size,size);
[A,B] = meshgrid(a,b);
C = (1-A).^2+100*((B-(A.^2)).^2);
levels = 100:100:100;
figure
hold on
contour(A,B,C,200)
plot(THEX,THEY)
plot(3,4,'g*')
plot(x,y,'r*')
legend('\nablaf(x,y)','Path','Start','End')
xlabel('X')
ylabel('Y')
title('Quasi-Newton DFP - Contour Plot and Algorithm Path')

dlmwrite('qu_CONV.txt',CONV);
```

5. Compare Convergence

```
st_CONV = dlmread('st_CONV.txt');
ne_CONV = dlmread('ne_CONV.txt');
qu_CONV = dlmread('qu_CONV.txt');
cg_CONV = dlmread('cg_CONV.txt');

semilogy(st_CONV,'-.g','LineWidth',2)
hold on
grid on
```

```
semilogy(ne_CONV,'r','LineWidth',2)
semilogy(qu_CONV,':b','LineWidth',2)
semilogy(cg_CONV,'--m','LineWidth',2)
legend('Steepest Descent','Newton''s Method','Quasi-Newton
(DFP)','Conjugate Gradient (FR)')
xlabel('Number of iterations')
ylabel('Norm of \nablaf(x,y)')
title('Convergence Comparison')
```

6. Compare Backtracking

```
cg_CONV = dlmread('cg_CONV.txt');
cg_CONV_back = dlmread('cg_CONV_back.txt');

semilogy(cg_CONV,':g','LineWidth',2)
hold on
grid on
semilogy(cg_CONV_back,'--r','LineWidth',2)
legend('Fixed Step-Length','Backtracking')
xlabel('Number of iterations')
ylabel('Norm of \nablaf(x,y)')
title('Fixed Step-Length vs. Backtracking - Conjugate Gradient (FR)')
```