```
%{
Jacob Leonard
MATH 467 - Fall 2015
jaleonar@usc.edu
Revision History
Date                    Changes                      Programmer
-------------------------------------------------------------------
11/6/2015               Original                   Jacob Leonard
11/7/2015          Developed Derivatives           Jacob Leonard
11/10/2015        Developed Algorithm Body         Jacob Leonard
11/12/2015        Developed Backtracking           Jacob Leonard
11/13/2015          Troubleshooting                Jacob Leonard
11/14/2015          Developed Z Function           Jacob Leonard

%}

%this script is for newtons method with back-tracking

%determine x(0) and y(0) for the start of the methods
for j = 1:101
    x(j) = (-2)+((4*(j-1))/100);
    y(j) = (-2)+((4*(j-1))/100);
end


%define an anonymous function handle for the equations that compose the gradient and the↙
hessian
f = @(x,y) ((x^4+y^4-6*x^2*y^2-1)^2+(4*x^3*y-4*x*y^3)^2);
G = {@(x,y) (8*x*(x^6+3*x^4*y^2+x^2*(3*y^4-1)+y^2*(y^4+3))),@(x,y) (8*y*(x^6+3*x^4*y^2+3*x^2*↙
(y^4+1)+y^2*(y^4-1)))};
%Gradient = [g{1}(x,y),g{2}(x,y)];
%when the
H = {@(x,y) 8*(7*x^6+15*x^4*y^2+x^2*(9*y^4-3)+y^2*(y^4+3)),@(x,y) 48*x*y*(x^4+2*x^2*y^2+y^4+1);↙
@(x,y) 48*x*y*(x^4+2*x^2*y^2+y^4+1),@(x,y) 8*(x^6+9*x^4*y^2+3*x^2*(5*y^4+1)+y^2*(7*y^4-3))};
%Hessian = [H{1}(x,y),H{2}(x,y);H{3}(x,y),H{4}(x,y)];

%this matrix defines the size of the final graph to be plotted for
%iterations
Newtons = zeros(101,101);

%desired level of accuracy
tolerance = 10^(-7);

%lowest value we wish to divide by
epsilon = 10^(-14);

%given each initial value, newtons method will iterate without the need to
%evaluate any points
k=1;
for i = 1:101
    for j = 1:101
        X(:,:,k) = [x(i);y(j)];
        g(:,:,k) = [G{1}(x(i),y(j)),G{2}(x(i),y(j))];
        h(:,:,k) = [H{1}(x(i),y(j)),H{2}(x(i),y(j));H{3}(x(i),y(j)),H{4}(x(i),y(j))];
        I(:,:,k) = inv(h(:,:,k));
        gT(:,:,k) = transpose(g(:,:,k));
        d(:,:,k)=(I(:,:,k)*gT(:,:,k));
        B = .8;
        A = .4;
```

```
        t=1;
        for k = 2:100
            %add backtracking to the line search
            s = 0;
            while s == 0
                c = f(X(1,1,k-1)+t*d(1,1,k-1),X(2,1,k-1)+t*d(2,1,k-1));
                m = f(X(1,1,k-1),(X(2,1,k-1)))+A*t*g(:,:,k-1)*d(:,:,k-1);
                if c>m
                    t = B*t;
                end
                if m>c
                    s=1;
                end
            end
            X(:,:,k) = X(:,:,k-1)-t*d(:,:,k-1);
            g(:,:,k) = [G{1}(X(1,1,k),X(2,1,k)),G{2}(X(1,1,k),X(2,1,k))];
            h(:,:,k) = [H{1}(X(1,1,k),X(2,1,k)),H{2}(X(1,1,k),X(2,1,k));H{3}(X(1,1,k),X(2,1,↙
k)),H{4}(X(1,1,k),X(2,1,k))];
            I(:,:,k) = inv(h(:,:,k));
            gT(:,:,k) = transpose(g(:,:,k));
            d(:,:,k)=(I(:,:,k)*gT(:,:,k));
            if (f(X(1,1,k),X(2,1,k))-f(X(1,1,k-1),X(2,1,k-1)))<tolerance
                Newtons(i,j) = k-1;
                break
            end
        end
    end
end

for i = 1:101
    for j = 1:101
        Z(i,j) = f(x(i),y(j));
    end
end
```