```
%{
Jacob Leonard
MATH 467 - Fall 2015
jaleonar@usc.edu
Revision History
Date                 Changes            Programmer
-------------------------------------------------
11/15/2015           Original            Jacob Leonard
%}

%this script is for the fixed step size gradient method
%formula but for z1 and z2 - the simplified quadratic function of x and y

%define the values of x and y from 0 to 2, increasing by 1/50, for 101
%values -> z1 = x^4, z2 = y^4, so z cant be negative although x and y can
for j = 1:201
    x(j) = (-2+(2*(j-1))/100);
    y(j) = (-2+(2*(j-1))/100);
end

z1 = x.^4;
z2 = y.^4;

%define an anonymous function handle for the equations that compose the gradient and the↙
hessian
f = @(x,y) (x^2+4*x^(3/2)*y^(1/2)+6*x*y-2*x+4*x^(1/2)*y^(3/2)+12*x^(1/2)*y^(1/2)+y^2-2*y+1);
G = {@(x,y) (2*x+(6*x^.5*(y)^.5)+(6*(y))-2+(2*x^(-.5)*(y)^(1.5))+(6*x^(-.5)*(y)^.5)),@(x,y) (2*↙
(y)+6*(y)^(.5)*x^.5+6*x-2+2*(y)^(-.5)*x^1.5+6*(y)^(-.5)*x^.5)};
%Gradient = [g{1}(x,y),g{2}(x,y)];
H = {@(x,y) (2+3*x^(-1/2)*y^(1/2)-x^(-3/2)*y^(3/2)-3*x^(-3/2)*y^(1/2)),@(x,y) (3*x^(1/2)*y^(-1↙
/2)+6+3*x^(-1/2)*y^(1/2)+3*x^(-1/2)*y^(-1/2));@(x,y) (3*x^(1/2)*y^(-1/2)+6+3*x^(-1/2)*y^(1/2)↙
+3*x^(-1/2)*y^(-1/2)),@(x,y) (2+3*y^(-1/2)*x^(1/2)-y^(-3/2)*x^(3/2)-3*x^(-3/2)*y^(1/2))};
%Hessian = [H{1}(x,y),H{2}(x,y);H{3}(x,y),H{4}(x,y)];

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

%this matrix shows the number of iterations for matlab to think the value
%is zero, or within the desired tolerance
FixedStep = zeros(201,201);
%this is the value of the function at the point the algorithm terminated
FixedStepValues = zeros(201,201);

%the algorithm is getting stuck at places where, after following the
%gradient, and the rules of g=0, it eventually gets sent to (0,0) and then
%never completes

%create vector graphs of the trouble areas like (0,0) and (1,0) and (0,1)
%to show how the gradient is behaving



for i = 1:201
    for j = 1:201
        %when calculating the graident, matlab can't divide by zero and
        %will say the gradient is NaN, and these values must be replaced by
        %changing the graident
        if z1(i) == 0 && z2(j) == 0
            G = {@(x,y) (2*x+6*y-2),@(x,y) (2*y+6*x-2)};
```

```
        end
        if z1(i) == 0 && z2(j) ~= 0
            G = {@(x,y) (2*x+6*x^.5*(y)^.5+6*(y)-2+2*x^(.5)*(y)^(1.5)+6*x^(.5)*(y)^.5),@(x,y)↙
(2*(y)+6*(y)^.5*x^.5+6*x-2+2*(y)^-.5*x^1.5+6*(y)^-.5*x^.5)};
        end
        if z1(i) ~= 0 && z2(j) == 0
            G = {@(x,y) (2*x+6*x^.5*(y)^.5+6*(y)-2+2*x^-.5*(y)^(1.5)+6*x^-.5*(y)^.5),@(x,y) (2*↙
(y)+6*(y)^.5*x^.5+6*x-2+2*(y)^.5*x^1.5+6*(y)^.5*x^.5)};
        else
            G = {@(x,y) (2*x+6*x^.5*(y)^.5+6*(y)-2+2*x^-.5*(y)^(1.5)+6*x^-.5*(y)^.5),@(x,y) (2*↙
(y)+6*(y)^.5*x^.5+6*x-2+2*(y)^-.5*x^1.5+6*(y)^-.5*x^.5)};
        end
        Z(:,:,1) = [z1(i);z2(j)];
        g(:,:,1) = [G{1}(z1(i),z2(j)),G{2}(z1(i),z2(j))];
        gT(:,:,1) = transpose(g(:,:,1));
        %the Q matrix for this function is [1,35;35,1) and determines the
        %range for alpha
        alpha = .45;
        %begin the iterations for steps
        for k = 2:10000
            p=1;
            q=1;
            %if the values go negative the algorithm has gone too far, and
            %is outside the given values
            Z(:,:,k) = Z(:,:,k-1)-alpha*(gT(:,:,k-1));
            %if the algorithm reaches a value less than the tolerance in
            %the sequence, for either x or y, then the value goes to zero,
            %and the gradient follows the other variable
            if Z(1,1,k) < tolerance
                p = Z(1,1,k);
                Z(1,1,k:10000) = 0;
            end
            if Z(2,1,k) < tolerance
                q = Z(2,1,k);
                Z(2,1,k:10000) = 0;
            end
            %the gradient returns NaN if either of the values is 0, if the
            %new Z values contain 0, the graident must be changed
            if Z(1,1,k) == 0 && Z(2,1,k) == 0
                G = {@(x,y) (2),@(x,y) (2)};
            end
            if Z(1,1,k) == 0 && Z(2,1,k) ~= 0
                G = {@(x,y) (2*x+6*x^.5*(y)^.5+6*(y)-2+2*x^(.5)*(y)^(1.5)+6*x^(.5)*(y)^.5),@(x,↙
y) (2*(y)+6*(y)^.5*x^.5+6*x-2+2*(y)^-.5*x^1.5+6*(y)^-.5*x^.5)};
            end
            if Z(1,1,k) ~= 0 && Z(2,1,k) == 0
                G = {@(x,y) (2*x+6*x^.5*(y)^.5+6*(y)-2+2*x^-.5*(y)^(1.5)+6*x^-.5*(y)^.5),@(x,y)↙
(2*(y)+6*(y)^.5*x^.5+6*x-2+2*(y)^.5*x^1.5+6*(y)^.5*x^.5)};
            else
                G = {@(x,y) (2*x+6*x^.5*(y)^.5+6*(y)-2+2*x^-.5*(y)^(1.5)+6*x^-.5*(y)^.5),@(x,y)↙
(2*(y)+6*(y)^.5*x^.5+6*x-2+2*(y)^-.5*x^1.5+6*(y)^-.5*x^.5)};
            end
            %find the new graident for the updated value
            g(:,:,k) = [G{1}(Z(1,1,k),Z(2,1,k)),G{2}(Z(1,1,k),Z(2,1,k))];
            %if the z1 value is less than the tolerance, then the z1 value
            %is set equal to 0, and the algorithm follows the other
            %gradient direction
            if p<tolerance
                g(1,1,k)=0;
```

```
            end
            %if the z2 value is less than the tolerance, then the z2 value
            %is set equal to 0, and the algorithm follows the other
            %gradient direction
            if q<tolerance
                g(1,2,k)=0;
            end
            gT(:,:,k) = transpose(g(:,:,k));
            %if the function value dips below the tolerance, then it is
            %considered to have converged to the optimal value
            if f(Z(1,1,k),Z(2,1,k))<tolerance;
                FixedStep(i,j) = k;
                FixedStepValues(i,j) = 0;
                break
            end
            if Z(1,1,k) == 0 && Z(2,1,k) == 0
                FixedStep(i,j) = k;
                FixedStepValues(i,j) = 1;
                break
            end
        end
    end
end
xAxis = linspace(-2,2,201);
yAxis = linspace(-2,2,201);

%this plot will look at the number of steps it took for the algorithm to finish, the real↙
values of the function over the interval
%for which the algorithm finished, and the  imaginary values for which the
%algorithm finished

subplot(2,2,1:2)
%this plot will show the number of iterations it took
contourf(xAxis,yAxis,FixedStep);
xlabel('x');
ylabel('y');
title('Number of steps for Fixed Step Size Method to Converge,Alpha=.45');
colorbar;
subplot(2,2,3:4)
FixedStepValuesReal = real(FixedStepValues);
contourf(xAxis,yAxis,FixedStepValuesReal);
xlabel('x');
ylabel('y');
title('Values After Convergence x=[-2:2], z2=[-2:2],Alhpa=.45');
colorbar;
```