```
function proj06
% proj06
% solves the steepest descent and simulated annealing problems
 outlined in
% the assignment
% Inputs: None
% Outputs: None
% Quan Le, CAAM 210, Fall 2019, Project 06
% Last Modified: October 26, 2019


steepest_descent_driver
simulated_annealing_method_driver

% comments:

% From the figure, notice that the behavior of each function as it
 reaches
% the minimizer is linear. In addition, the behavior of each function
 seems
% to be strongly correlated with the step size, alpha. When alpha for
 each
% function was kept at 1/34, the plots all quickly converge.

% Therefore, it can be concluded that the number of iterations are
% inversely dependent on alpha.
end



function steepest_descent_driver
% steepest_descent_driver
% solves the steepest descent problem, plots the iterations vs the
% gradient norms
% Inputs: None
% Outputs: None


close all % clears old graphs

% create symbolic functions
x1 = sym('x1');
x2 = sym('x2');
fset = [];
for i = 1:4
    fset = [fset x1^2 + (2^i)*x2^2];
end

% create symbolic gradients
Fset = [];
for func = fset
    Fset = [Fset gradient(func)];
```

```
end

a = [1/6; 1/10; 1/18; 1/34]; % variable alpha
xmins = [];

% calls steepest_descent for each function
for i = 1:4
    F = matlabFunction(Fset(:,i)); % gets the anonymous function
    [min,norms] = steepest_descent(F,a(i),1);
    xmins = [xmins min]; % creates set of calculated minimums
    semilogy(norms) % plots norms on log graph, wrt their indicies
    hold on
end

% labels for graph
xlabel('iterations')
ylabel('gradient norms')
title("Performance of the steepest descent method on problems")
make_legend = ["f_1(x_1,x_2) = x_1^2 + 2x_2^2";
               "f_2(x_1,x_2) = x_1^2 + 4x_2^2";
               "f_3(x_1,x_2) = x_1^2 + 8x_2^2";
               "f_4(x_1,x_2) = x_1^2 + 16x_2^2"];
legend(make_legend)

% minimizers
for i = 1:4
    disp("Minimizer for f" + string(i) + ": "+ string(xmins(i)))
end
end



function [x,norms]=steepest_descent(F,a,x1)
% [x,norms]=steepest_descent(F,a,x1)
% finds the norms and the final minimizer of a function
% Inputs: F, the anonymous gradient function
%         a, the step size
%         x1, initial value for x
% Outputs: x, the minimizer found by the steepest descent method
%          norms, the vector of norms iterated over by the method


% initial values
e = 10^(-6);
x = x1;
y = 1;
grad = F(x,y);
norms = [];

% iterating until norm < e
while norm(grad, 4) >= e
    norms = [norms; norm(grad, 2)];
    x = x-a*grad(1);
    y = y-a*grad(2);
```

```
    grad = F(x,y);
end
end



function simulated_annealing_method_driver
% simulated_annealing_method_driver
% runs simulated annealing and steepest descent for the cost function
 f
% Inputs: None
% Outputs: None


% generates anonymous versions of the function and the gradient
x= sym('x');
fsym = -exp(1)^(-x^2) -0.3*exp(1)^(-(x-3)^2) +0.01*x^2;
f = matlabFunction(fsym);
F = matlabFunction(gradient(fsym));

% steepest descent
e = 10^(-6);
a = 1/8;
x = 2;
df = F(x);
while abs(df) >= e
    x = x-a*df;
    df = F(x);
end

% fmincon
fminconmin=fmincon(f,0,0,1);

% runs simulated annealing 1000 times, collects the minimizers
minimizers = [];
for i = 1:1000
    minimizers = [minimizers
        simulated_annealing_method(e,a,2,50,1,F,f)];
end

% finds the amount of times simulated annealing found the global
 minimizer
globalmin = (find(abs(minimizers)<0.001));
goodtrials = length(globalmin);

% prints a summary of results
fprintf('\n')
disp('summary of results for the cost function f')
fprintf('\n')
disp("steepest descent minimizer: " + string(x))
disp("fmincon minimizer: " + string(fminconmin))
disp("number of successful trials for the simulated annealing method:
 " + string(goodtrials))
end
```

```
function [x]=simulated_annealing_method(e,a,x1,delta,t1,F,f)
% [x]=simulated_annealing_method(e,a,x1,delta,t1,F,f)
% runs simulated annealing method for given parameters
% Inputs: e, the tolerance for the norm of the gradient
%         a, the step size
%         x1, inital value to start simulated annealing
%         delta, a randomness parameter
%         t1, initial temprature
%         F, anonymous gradient function
%         f, anonymous cost function
% Outputs: x, the minimizer found using simulated annealing


% initialize variables
x = x1;
t=t1;
df = F(x);

% run simulated annealing
while abs(df) >= e
    R = randn;
    xplus = x-a*df+t*delta*R;
    p = min(exp((f(x)-f(xplus))/t),1);
    a = rand;
    if a < p % if the condition is met, update x
        x = xplus;
        df = F(x);
    end
    t = 0.9*t;
end
end
```

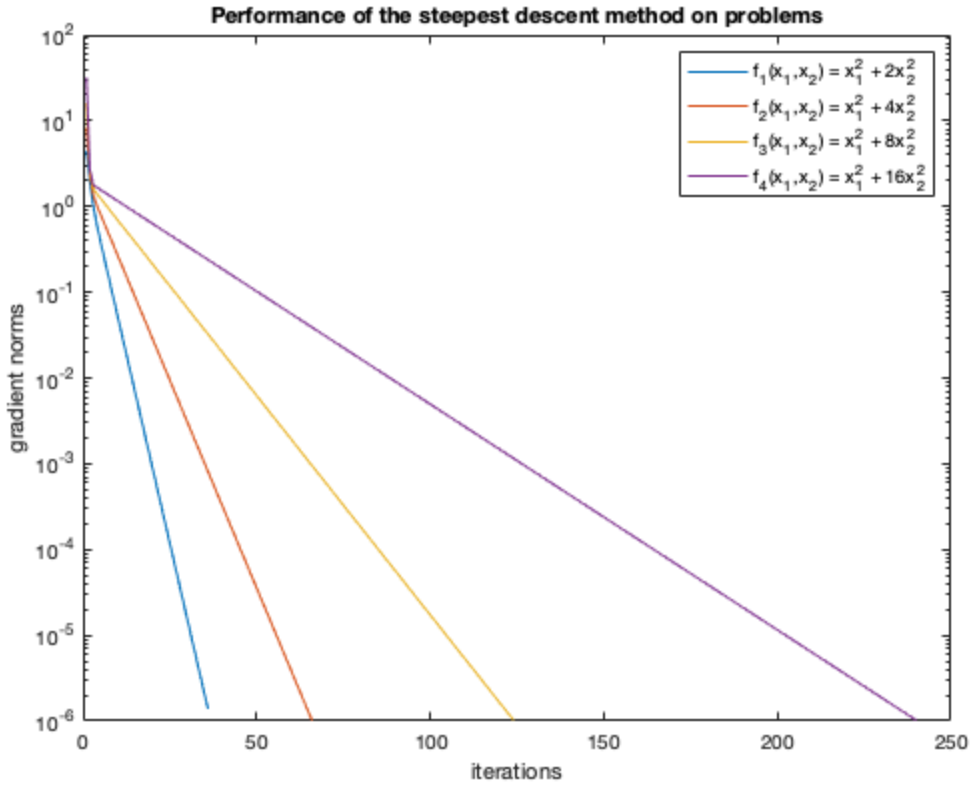
```
Minimizer for f1: 4.5784e-07
Minimizer for f2: 4.0173e-07
Minimizer for f3: 4.5403e-07
Minimizer for f4: 4.7979e-07

Local minimum found that satisfies the constraints.

Optimization completed because the objective function is non-
decreasing in
feasible directions, to within the value of the optimality tolerance,
and constraints are satisfied to within the value of the constraint
 tolerance.


summary of results for the cost function f

steepest descent minimizer: 2.9002
fmincon minimizer: 0.00011003
number of successful trials for the simulated annealing method: 987
```



Published with MATLAB® R2019a