EE5321, Spring 2018
Homework 2: Static Optimization – Due Feb 8

Problem 1) Unconstrained optimization of a single variable - 30 points
Given the following polynomial
$y = (x^4 - 4x^3 + 8x + 1)$

a) Compute the first derivative to determine the critical points.
b) Using the second derivative, determine the nature of the critical points (i.e. min, max, etc.). Compute the value of y for each critical point. What is the global min?
c) Using MATLAB, plot the function in the range $-2 \leq x \leq 4$
d) Develop a MATLAB script that uses fminunc to numerically find the minimum near the starting point $x = -1.5$, stating the minimum value of the function y and the optimal value of x.
e) Use that same script but start at $x = 1.1$, stating the minimum value of the function y and the optimal value of x.
f) What happens if you start at x = 1.0?

Example of a cost function:

```
function cost = hw2p1_cost(x)
   cost = 1 / 2 * (x^2 – 2*x + 1);
end
```

Problem 2) Unconstrained optimization of two variables - 30 points
Replace the polynomial function of Problem 1 with the following

$$z = \sin(x) + \cos(y)$$

With the limits $\frac{pi}{2} \leq x \leq 2 * pi$ and $\frac{pi}{2} \leq y \leq 2 * pi$

a) Plot the function for the given ranges of x and y.
b) Starting at $[x, y] = [5.5, 5.5]$, find the nearest minimum using fminunc, stating the optimal values of x and y along with the optimal cost.

Problem 3) Constrained optimization of two variables - 40 points
Now switch from *fminunc* to *fmincon* to see the impact of constraining the optimization procedure for the polynomial of Problem 2, above. Note that *fmincon* requires more inputs, but offers more flexibility and control over the fitting process. Look at the help on *fmincon* to see a description of the use of the function. In addition to the cost function you defined in Problem 2, you'll need to develop a constraint function which is outlined below. Have the constraint function implement the constraint inequality $y > 4$.
To test if you have *fmincon* implemented correctly, you can have both cineq and ceq return [] initially, which means that the constraints are satisfied.

a) Starting at $[x, y] = [5.5, 5.5]$, find the nearest minimum using *fmincon* that satisfies the constraint y > 4. Repeat the optimization starting at [2, 2] and note the differences, if any.

b) Show a contour plot of the function and show the location of the minimum without the inequality constraint and the location of the minimum with the constraint. Sketch the constraint line on the graph as well.

Example of a constraint function:

```
function [cineq, ceq] = hw_constraint(p)
% cineq – scalar or vector of inequality constraints, make cineq = [] if there are no
inequality constraints
% ceq – scalar or vector of equality constraints, make ceq = [] if there are no equality
constraints
% p – parameter vector being optimized, you may call them anything you want
% perform ceq and cineq calculations, of which there can be one or many of each

   ceq = [];
   if p(1) + p(2) > 0
      cineq = 0;
   else
      cineq = <insert your condition here>;
   end

end
```