Assignment #2: Unconstrained Optimization — ME 575

due 2/10/2021 before midnight via Learning Suite — **50 possible points**

Overview: Write an unconstrained gradient-based optimization algorithm. We have discussed various line search methods and search direction methods. You can use any combination of method(s) you want, but you must develop your own code. You will test your algorithm on the three test functions described at the end of the problem. For each of the three test problems you will solve it two ways: 1) using an algorithm you develop, and 2) using an existing optimization algorithm.

Be sure to address the following elements in your report:

2.1 Describe your algorithm and why/how you selected that methodology.

2.2 Create a table reporting the number of function calls required for convergence on all three test problems and for both algorithms (yours and `fminunc`) on each problem. While not required, it might be helpful to try one more than one starting point. Sometimes just using one point can be misleading. Discuss your findings.

2.3 Create a convergence plot for all three test problems. For each plot you only need to show the result using your algorithm. A convergence plot should show iterations on the x-axis and a convergence metric with a log scale on the y-axis. Some suggested convergence metrics include $|f - f^*|$, or $||g||$ (since f^* is usually not known).

2.4 Discuss the main challenges you faced, lessons learned, and areas where you think your approach could most effectively be improved. (As always you should include your code also).

Test Cases:

The three tests cases are found in the appendix: C.1.1 (a quadratic function), C.1.2 (Rosenbrock), and C.1.7 (the same Brachistochrone problem you used in the previous homework). For the Brachistochrone use 60 evenly spaced across the path (i.e., 58 design variables).

Extra Credit:

Top-performing algorithms will be awarded with extra credit. If you wish to participate in the extra credit portion you must do two more things:

1. Provide a psuedoname so we can share results with the class.
2. Download the template file from the course website and *do not* change the function signature.

Performance will be measured by the total number of function calls required to converge all three test problems, plus two additional test problems that will not be disclosed until after submission. In order to facilitate automated testing, your code submission must use the expected function signature. In the testing we will use our own implementations of the five functions, and will supply exact gradients.

Tips:

- Obtaining gradients for the first two functions is straightforward. The Brachistochrone problem is bit tricker. I've provided them for you in a script in the template folder.
- Start simple. Get something simple working before moving to something more complex. Beginners often try to write everything then test at the end. The difficulty is that when something inevitably goes wrong it is harder to track down. Better to test basic functions in isolation then build up complexity. For example, test your line search function by itself. Give it various cases and make sure the results are as you would expect.

- Before testing a problem on your optimizer, I would test it on an existing optimizer. If it doesn't work with those, then it is likely that there is a problem with your formulation.
- Plotting can help. Especially with the 2D problems you can visualize what your algorithm is doing.
- Anytime you provide gradients you should always check them. First I would run the optimization without providing gradients (the optimizer will internally finite difference). Then I would compare the provided gradients against finite differencing. This is not hard to do yourself, but many optimizers will check for you if you set the appropriate option.