HW2
CS 169/268 Optimization
Fall 2018
Due: Thursday October 18 11:59pm on EEE

As in HW1:
Allowed: Reading (but not copying) pseudocode and code in the three recommended/optional books listed first in the class Syllabus. Also, reading external pseudocode after you have tried to write your own code.
Not allowed: Other outside/external code reading or code use (eg. copying or execution).

For each problem turn in: 1-page written description of your approach and results, together with source code and I/O files proving your results.
File types allowed for submission: .doc, .pdf, .py, .txt, and .ipynb files.
Programming language, external software: As in HW1 (non-extra-credit portion).

In like manner to HW1:

1. (Undergrad + Grad students) Implement and test the conjugate-gradient method of optimization for multi-variable unconstrained optimization problems. Test on two multivariable unconstrained optimization problems, of dimensions at least 2 and 10. You may choose the problems but give them nontrivial dependencies among the variables. (For example, Rosenbrock's function for dimension 2.) Parameters to vary include: number of CG iterates before restart; stopping criterion; others optional (eg control parameters or stopping criteria for the line minimization search). For each value of the parameters report averages of: (a) average error of the output argument vector and (b) function value produced by the algorithm, and also (c) total number of CG iterations (not sweeps) as a measure of computational cost, along with statistical error bars on those quantities. Probability distribution to use in producing a single average error measurement includes a distribution (eg a broad Gaussian) over starting points; just say what you chose.

2. (Grad students only; UG for extra credit) Compare the performance of your CG numerically (as above) to some other gradient-based method that you program, such as steepest descent, or (more sophisticated!) Newton's method or a quasi-Newton method; but *not* coordinate descent (that was in HW1). Based on your computer experiments, what are the pros and cons of CG vs. the other method you chose to implement?

Everyone:

Included with this assignment specification is a template to use for your optimizer. For problem 1, fill in the method conjugate_gradients. For Problem 2, fill in alternative_method, adding other functions to the template if necessary. You may not modify the function header or return type of either alternative_method or conjugate_gradients, except to add additional arguments with default values. The template has examples of how to include optional arguments in this way (x_tol and f_tol in conjugate_gradients, and gamma and step_evaluator in alternative_method) - feel free to remove them once you get started. Your functions must be callable with just the positional arguments, i.e. this file must be runnable with the last two lines unchanged.
(… See template code next page!)

```
def conjugate_gradients(func, x0, fprime, restart_frequency, x_tol = 0.0005, f_tol = 0.01):
    #----------------------
    #student code goes here:
    x_final = x0
    f_final = func(x0)
    CG_iterations = 1
    #----------------------
    return x_final, f_final, CG_iterations
def alternate_method(func, x0, fprime, gamma = 6.28, step_evaluator=lambda x: x**2.0):
    #----------------------
    #student code goes here:
    x_final = x0
    f_final = func(x0)
    #----------------------
    return x_final, f_final
if __name__=='__main__':
    test_func = lambda x : x[0]**2.0 + 15.0*x[0]*x[1] + 3.0*(x[1]**2.0)
    test_fprime = lambda x: (
        2.0*x[0] + 15.0*x[1],
        6.0*x[1] + 15.0*x[0]
    )
    init_pt = (0.3, 0.1)
    print(conjugate_gradients(test_func, init_pt, test_fprime, 1))
    print(alternate_method(test_func, init_pt, test_fprime))
```