Assignment 5.

Given March 24, due March 31.

Objective: Work with optimization methods and robust software.

This assignment asks you to write good, robust numerical software for finding the minimum of a function of several variables. Pay particular attention to independent testing, modularity, and graceful failure. Make sure the test cases for individual components sometimes lead to failure so you can see the failure mechanisms working. In general, it is important to design testers to exercise all features of the code.

1. Newton's method for solving a system of nonlinear equations is

$$x_{n+1} = x_n - \left(\frac{\partial f}{\partial x}(x_n) \right)^{-1} f(x_n) \; . \tag{1}$$

Here $\left(\frac{\partial f}{\partial x}(x_n) \right)$ is the $n \times n$ Jacobian matrix of partial derivatives. We could instead use different variables linearly related to x, which may be written $y = Ax$. For example, changing units would give a diagonal A with conversion factors on the diagonal. We denote the function in the y variables by g, which gives $g(y) = f(A^{-1}y)$. Show that applying Newton's method to g in the y variables is equivalent to using (1) in the x variables in the sense that if the first iterates are equivalent: $y_0 = Ax_0$, then all the subsequent iterates will also be equivalent.

2. Write a program that performs unsafeguarded Newton's method to find the minimum of a function of several variables. Use your Choleski factorization routine from assignment 4. Make sure to test whether the Choleski factorization succeeded and fail gracefully if it did not. Try your program on the function

$$V(x,y) = \phi\left(x^2 + a(y - b\sin(cx))^2 \right) \tag{2}$$

with $a = 2$, $b = 1$, and $c = 1$. Here, $\phi(t) = e^t - 1/(1+t^2)$. Try initial guesses close and far from the solution ($x = y = 0$). With a good initial guess you should see quadratic convergence to the minimizer. With a poor initial guess you should see wild behavior or termination with failure. Notes:

- You have to code the gradient and Hessian of V. Do this so that it would be exact in exact arithmetic. It may be easier not to do this in a single formula such as $\partial_{xy}V = \cdots\cdots$, but to express the results in terms of helpful intermediate results, such as $\phi' = \cdots$, $\phi'' = \cdots$, $w(x,y) = x^2 + a(y - \cdots)^2$, $w' = \cdots$, etc.

- Write a tester that checks whether the first and second partials of a function $V(x, y)$ have been coded correctly by using finite differences to estimate the derivatives and comparing the estimates to the supposedly exact (in exact arithmetic) values. Write the tester so that it calls the procedure or procedures that compute V and its derivatives with the same calling arguments (signature) used by the optimizer. Hand in output from the tester but do not incorporate the tester into the "production" code that actually does the optimization.
- Make a contout plot of the function V that shows the iterates as dots or little circles or something similar. Be take the time to make fine tune (not terribly fine) the plot so that it is informative. This is particularly important for the more extreme parameter values in part 3 below.
- Print compute and print the residuals after each iteration, noting on the output where quadratic convergence is or is not evident.
- Your optimization program must not have any potential infinite loops. This means, for example, a bound on the number of iterations before it gives up. As always, the routine must not fail silently if it fails for any reason to produce an acceptable answer.

3. Add two safeguards to the program from step 2. Do a binary bisection line search to make sure the step is not crazy, and make sure that the Newton step is a descent direction. You may use the "identity" Hessian (i.e. the negative gradient) or the modified Choleski facorization. the latter requires you to change just one line of code in your Choleski factorization routing. Use this to minimize the objective function (2) with $a = 30$, $b = 1$, and $c = 6$, starting from initial guess $(x_1, x_2) = (3, 1)$. Follow all the notes from part 2. In addition, write a tester for the line search program and test it on some simple model one dimensional problems. The line search part should not know it is optimizing a multivariate function at all.

4. Just to check that you did not hard code the dimension, apply your code to minimizing the function

$$V(x) = \frac{1}{2} x^* A x + e^{x_1} \; , \tag{3}$$

where A is the $n \times n$ martix from part 2 of assignment 4 with $s = 0$. You may use initial guess $x = 0$ and $n = 10$. If you did parts 2 and 3 well, this should take just a few minutes.