Homework 10 additional problems

1. *Suggestions for exercises 9.30 in* Convex Optimization. We recommend the following to generate a problem instance:

   ```
   n = 100;
   m = 200;
   randn('state',1);
   A=randn(m,n);
   ```

   Of course, you should try out your code with different dimensions, and different data as well.

   In all cases, be sure that your line search *first* finds a step length for which the tentative point is in **dom** f; if you attempt to evaluate f outside its domain, you'll get complex numbers, and you'll never recover.

   To find expressions for $\nabla f(x)$ and $\nabla^2 f(x)$, use the chain rule (see Appendix A.4); if you attempt to compute $\partial^2 f(x)/\partial x_i \partial x_j$, you will be sorry.

   To compute the Newton step, you can use `vnt=-H\g`.

2. *Suggestions for exercise 9.31 in* Convex Optimization. For 9.31a, you should try out $N = 1$, $N = 15$, and $N = 30$. You might as well compute and store the Cholesky factorization of the Hessian, and then back solve to get the search directions, even though you won't really see any speedup in Matlab for such a small problem. After you evaluate the Hessian, you can find the Cholesky factorization as `L=chol(H,'lower')`. You can then compute a search step as `-L'\(L\g)`, where `g` is the gradient at the current point. Matlab will do the right thing, *i.e.*, it will first solve `L\g` using forward substitution, and then it will solve `-L'\(L\g)` using backward substitution. Each substitution is order n^2.

   To fairly compare the convergence of the three methods (*i.e.*, $N = 1$, $N = 15$, $N = 30$), the horizontal axis should show the approximate total number of flops required, and not the number of iterations. You can compute the approximate number of flops using $n^3/3$ for each factorization, and $2n^2$ for each solve (where each 'solve' involves a forward substitution step and a backward substitution step).

6.079 / 6.975 Introduction to Convex Optimization
Fall 2009