Homework 9

1. *Efficient numerical method for a regularized least-squares problem.* We consider a regularized least squares problem with smoothing,

$$\text{minimize} \quad \sum_{i=1}^{k}(a_i^T x - b_i)^2 + \delta \sum_{i=1}^{n-1}(x_i - x_{i+1})^2 + \eta \sum_{i=1}^{n} x_i^2,$$

where $x \in \mathbf{R}^n$ is the variable, and $\delta, \eta > 0$ are parameters.

   (a) Express the optimality conditions for this problem as a set of linear equations involving x. (These are called the normal equations.)

   (b) Now assume that $k \ll n$. Describe an efficient method to solve the normal equations found in (1a). Give an approximate flop count for a general method that does not exploit structure, and also for your efficient method.

   (c) *A numerical instance.* In this part you will try out your efficient method. We'll choose $k = 100$ and $n = 2000$, and $\delta = \eta = 1$. First, randomly generate A and b with these dimensions. Form the normal equations as in (1a), and solve them using a generic method. Next, write (short) code implementing your efficient method, and run it on your problem instance. Verify that the solutions found by the two methods are nearly the same, and also that your efficient method is much faster than the generic one.

   Note: You'll need to know some things about Matlab to be sure you get the speedup from the efficient method. Your method should involve solving linear equations with tridiagonal coefficient matrix. In this case, both the factorization and the back substitution can be carried out very efficiently. The Matlab documentation says that banded matrices are recognized and exploited, when solving equations, but we found this wasn't always the case. To be sure Matlab knows your matrix is tridiagonal, you can declare the matrix as sparse, using `spdiags`, which can be used to create a tridiagonal matrix. You could also create the tridiagonal matrix conventionally, and then convert the resulting matrix to a sparse one using `sparse`.

   One other thing you need to know. Suppose you need to solve a group of linear equations with the same coefficient matrix, *i.e.*, you need to compute $F^{-1}a_1, ..., F^{-1}a_m$, where F is invertible and a_i are column vectors. By concatenating columns, this can be expressed as a single matrix

$$\left[F^{-1}a_1 \ \cdots \ F^{-1}a_m\right] = F^{-1}\left[a_1 \ \cdots \ a_m\right].$$

   To compute this matrix using Matlab, you should collect the righthand sides into one matrix (as above) and use Matlab's backslash operator: `F\A`. This will do the right thing: factor the matrix F once, and carry out multiple back substitutions for the righthand sides.

2. *Bounding portfolio risk with incomplete covariance information.* Consider the following instance of the problem described in §4.6, on p171–173 of *Convex Optimization*. We suppose that Σ_{ii}, which are the squares of the price volatilities of the assets, are known. For the off-diagonal entries of Σ, all we know is the sign (or, in some cases, nothing at all). For example, we might be given that $\Sigma_{12} \geq 0$, $\Sigma_{23} \leq 0$, etc. This means that we do not know the correlation between p_1 and p_2, but we do know that they are nonnegatively correlated (*i.e.*, the prices of assets 1 and 2 tend to rise or fall together).

   Compute σ_{wc}, the worst-case variance of the portfolio return, for the specific case

$$x = \begin{bmatrix} 0.1 \\ 0.2 \\ -0.05 \\ 0.1 \end{bmatrix}, \quad \Sigma = \begin{bmatrix} 0.2 & + & + & \pm \\ + & 0.1 & - & - \\ + & - & 0.3 & + \\ \pm & - & + & 0.1 \end{bmatrix},$$

   where a "+" entry means that the element is nonnegative, a "−" means the entry is nonpositive, and "±" means we don't know anything about the entry. (The negative value in x represents a *short position*: you sold stocks that you didn't have, but must produce at the end of the investment period.) In addition to σ_{wc}, give the covariance matrix Σ_{wc} associated with the maximum risk. Compare the worst-case risk with the risk obtained when Σ is diagonal.

3. *Equilibrium position of a system of springs.* We consider a collection of n small masses in $\mathbf{R}^2$, with locations $(x_1, y_1), \ldots, (x_n, y_n)$, and masses $m_1, \ldots, m_n$. (In other words, the vector $x \in \mathbf{R}^n$ gives the x-coordinates, and $y \in \mathbf{R}^n$ gives the y-coordinates, of the points.) The masses m_i are, of course, positive.

   For $i = 1, \ldots, n-1$, mass i is connected to mass $i+1$ by a spring. The potential energy in the ith spring is a function of the (Euclidean) distance $d_i = \|(x_i, y_i) - (x_{i+1}, y_{i+1})\|_2$ between the ith and $(i+1)$st masses, given by

$$E_i = \begin{cases} 0 & d_i < l_i \\ (k_i/2)(d_i - l_i)^2 & d_i \geq l_i \end{cases}$$

   where $l_i \geq 0$ is the rest length, and $k_i > 0$ is the stiffness, of the ith spring. The gravitational potential energy of the ith mass is gm_iy_i, where g is a positive constant. The total potential energy of the system is therefore

$$E = \sum_{i=1}^{n-1} E_i + gm^T y.$$

   The locations of the first and last mass are fixed. The equilibrium location of the other masses is the one that minimizes E.

   (a) Show how to find the equilibrium positions of the masses $2, \ldots, n-1$ using convex optimization. Be sure to justify convexity of any functions that arise in your formulation (if it is not obvious). Justify any new variables or constraints that you introduce. The problem data are m_i, k_i, l_i, g, x_1, y_1, x_n, and y_n.

(b) Carry out your method to find the equilibrium positions for a problem with $n = 10$, $m_i = 1$, $k_i = 10$, $l_i = 1$, $x_1 = y_1 = 0$, $x_n = 10$, $y_n = 10$, with g varying from $g = 0$ (no gravity) to $g = 10$ (say). Verify that the results look reasonable. Plot the equilibrium configuration for several values of g.

4. *Estimating a vector with unknown measurement nonlinearity.* (A specific instance of exercise 7.9 in *Convex Optimization*.) We want to estimate a vector $x \in \mathbf{R}^n$, given some measurements

$$y_i = \phi(a_i^T x + v_i), \quad i = 1, \ldots, m.$$

Here $a_i \in \mathbf{R}^n$ are known, v_i are IID $\mathcal{N}(0, \sigma^2)$ random noises, and $\phi : \mathbf{R} \to \mathbf{R}$ is an unknown monotonic increasing function, known to satisfy

$$\alpha \leq \phi'(u) \leq \beta,$$

for all u. (Here α and β are known positive constants, with $\alpha < \beta$.) We want to find a maximum likelihood estimate of x and ϕ, given y_i. (We also know a_i, σ, α, and β.)

This sounds like an infinite-dimensional problem, since one of the parameters we are estimating is a function. In fact, we only need to know the m numbers $z_i = \phi^{-1}(y_i)$, $i = 1, \ldots, m$. So by estimating ϕ we really mean estimating the m numbers $z_1, \ldots, z_m$. (These numbers are not arbitrary; they must be consistent with the prior information $\alpha \leq \phi'(u) \leq \beta$ for all u.)

(a) Explain how to find a maximum likelihood estimate of x and ϕ (*i.e.*, $z_1, \ldots, z_m$) using convex optimization.

(b) Carry out your method on the data given in `nonlin_meas_data.m`, which includes a matrix $A \in \mathbf{R}^{m \times n}$, with rows $a_1^T, \ldots, a_m^T$. Give $\hat{x}_{\mathrm{ml}}$, the maximum likelihood estimate of x. Plot your estimated function $\hat{\phi}_{\mathrm{ml}}$. (You can do this by plotting $(\hat{z}_{\mathrm{ml}})_i$ versus y_i, with y_i on the vertical axis and $(\hat{z}_{\mathrm{ml}})_i$ on the horizontal axis.)

Hint. You can assume the measurements are numbered so that y_i are sorted in nondecreasing order, *i.e.*, $y_1 \leq y_2 \leq \cdots \leq y_m$. (The data given in the problem instance for part (b) is given in this order.)

6.079 / 6.975 Introduction to Convex Optimization
Fall 2009