

EE364a Homework 7 additional problems

Suggestions for exercise 9.30

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`.

Suggestions for exercise 9.31

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 \mathbf{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).

Additional exercises

1. *A structural optimization problem.* Figure 1 shows a two-bar truss with height $2h$ and width w . The two bars are cylindrical tubes with inner radius r and outer radius R . We are interested in determining the values of r , R , w , and h that minimize the weight of the truss subject to a number of constraints. The structure should be strong enough

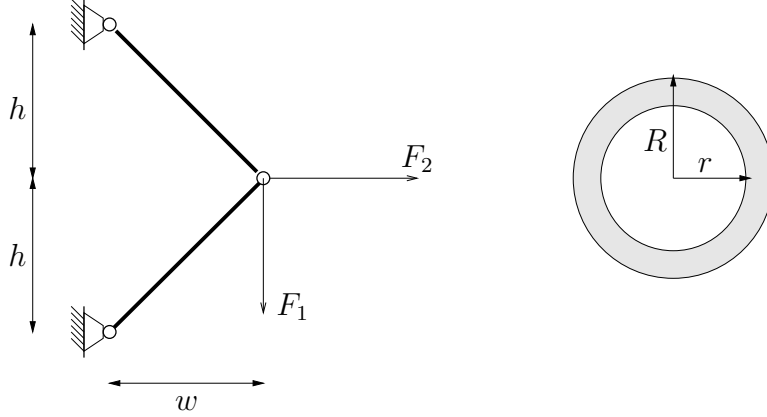


Figure 1 Two-bar truss and cross section of one of the two bars.

for two loading scenarios. In the first scenario a vertical force F_1 is applied to the node; in the second scenario the force is horizontal with magnitude F_2 .

The weight of the truss is proportional to the total volume of the bars, which is given by

$$2\pi(R^2 - r^2)\sqrt{w^2 + h^2}$$

This is the cost function in the design problem.

The first constraint is that the truss should be strong enough to carry the load F_1 , *i.e.*, the stress caused by the external force F_1 must not exceed a given maximum value. To formulate this constraint, we first determine the forces in each bar when the structure is subjected to the vertical load F_1 . From the force equilibrium and the geometry of the problem we can determine that the magnitudes of the forces in two bars are equal and given by

$$\frac{\sqrt{w^2 + h^2}}{2h}F_1.$$

The maximum force in each bar is equal to the cross-sectional area times the maximum allowable stress σ (which is a given constant). This gives us the first constraint:

$$\frac{\sqrt{w^2 + h^2}}{2h}F_1 \leq \sigma\pi(R^2 - r^2).$$

The second constraint is that the truss should be strong enough to carry the load F_2 . When F_2 is applied, the magnitudes of the forces in two bars are again equal and given by

$$\frac{\sqrt{w^2 + h^2}}{2w}F_2,$$

which gives us the second constraint:

$$\frac{\sqrt{w^2 + h^2}}{2w}F_2 \leq \sigma\pi(R^2 - r^2).$$

We also impose limits $w_{\min} \leq w \leq w_{\max}$ and $h_{\min} \leq h \leq h_{\max}$ on the width and the height of the structure, and limits $1.1r \leq R \leq R_{\max}$ on the outer radius.

In summary, we obtain the following problem:

$$\begin{aligned}
& \text{minimize} && 2\pi(R^2 - r^2)\sqrt{w^2 + h^2} \\
& \text{subject to} && \frac{\sqrt{w^2 + h^2}}{2h}F_1 \leq \sigma\pi(R^2 - r^2) \\
& && \frac{\sqrt{w^2 + h^2}}{2w}F_2 \leq \sigma\pi(R^2 - r^2) \\
& && w_{\min} \leq w \leq w_{\max} \\
& && h_{\min} \leq h \leq h_{\max} \\
& && 1.1r \leq R \leq R_{\max} \\
& && R > 0, r > 0, w > 0, h > 0.
\end{aligned}$$

The variables are R, r, w, h .

Formulate this as a geometric programming problem.

2. *Bounding object position from multiple camera views.* A small object is located at unknown position $x \in \mathbf{R}^3$, and viewed by a set of m cameras. Our goal is to find a box in \mathbf{R}^3 ,

$$\mathcal{B} = \{z \in \mathbf{R}^3 \mid l \preceq z \preceq u\},$$

for which we can guarantee $x \in \mathcal{B}$. We want the smallest possible such bounding box. (Although it doesn't matter, we can use volume to judge 'smallest' among boxes.)

Now we describe the cameras. The object at location $x \in \mathbf{R}^3$ creates an image on image plane of camera i at location

$$v_i = \frac{1}{c_i^T x + d_i}(A_i x + b_i) \in \mathbf{R}^2.$$

The matrices $A_i \in \mathbf{R}^{2 \times 3}$, vectors $b_i \in \mathbf{R}^2$ and $c_i \in \mathbf{R}^3$, and real numbers $d_i \in \mathbf{R}$ are known, and depend on the camera positions and orientations. We assume that $c_i^T x + d_i > 0$. The 3×4 matrix

$$P_i = \begin{bmatrix} A_i & b_i \\ c_i^T & d_i \end{bmatrix}$$

is called the *camera matrix* (for camera i). It is often (but not always) the case that the first 3 columns of P_i (*i.e.*, A_i stacked above c_i^T) form an orthogonal matrix, in which case the camera is called *orthographic*.

We do not have direct access to the image point v_i ; we only know the (square) pixel that it lies in. In other words, the camera gives us a measurement \hat{v}_i (the center of the pixel that the image point lies in); we are guaranteed that

$$\|v_i - \hat{v}_i\|_\infty \leq \rho_i/2,$$

where ρ_i is the pixel width (and height) of camera i . (We know nothing else about v_i ; it could be any point in this pixel.)

Given the data $A_i, b_i, c_i, d_i, \hat{v}_i, \rho_i$, we are to find the smallest box \mathcal{B} (*i.e.*, find the vectors l and u) that is guaranteed to contain x . In other words, find the smallest box in \mathbf{R}^3 that contains all points consistent with the observations from the camera.

- (a) Explain how to solve this using convex or quasiconvex optimization. You must explain any transformations you use, any new variables you introduce, etc. If the convexity or quasiconvexity of any function in your formulation isn't obvious, be sure justify it.
 - (b) Solve the specific problem instance given in the file `camera_data.m`. Be sure that your final numerical answer (*i.e.*, l and u) stands out.
3. *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.

Hint. To handle positive semidefiniteness of a matrix $X \in \mathbf{S}^n$ in CVX, use `X == semidefinite(n)`; see the user guide.

4. *Three-way linear classification.* We are given data

$$x^{(1)}, \dots, x^{(N)}, \quad y^{(1)}, \dots, y^{(M)}, \quad z^{(1)}, \dots, z^{(P)},$$

three nonempty sets of vectors in \mathbf{R}^n . We wish to find three affine functions on \mathbf{R}^n ,

$$f_i(z) = a_i^T z - b_i, \quad i = 1, 2, 3,$$

that satisfy the following properties:

$$\begin{aligned} f_1(x^{(j)}) &> \max\{f_2(x^{(j)}), f_3(x^{(j)})\}, & j = 1, \dots, N, \\ f_2(y^{(j)}) &> \max\{f_1(y^{(j)}), f_3(y^{(j)})\}, & j = 1, \dots, M, \\ f_3(z^{(j)}) &> \max\{f_1(z^{(j)}), f_2(z^{(j)})\}, & j = 1, \dots, P. \end{aligned}$$

In words: f_1 is the largest of the three functions on the x data points, f_2 is the largest of the three functions on the y data points, f_3 is the largest of the three functions on the z data points. We can give a simple geometric interpretation: The functions f_1 , f_2 , and f_3 partition \mathbf{R}^n into three regions,

$$\begin{aligned} R_1 &= \{z \mid f_1(z) > \max\{f_2(z), f_3(z)\}\}, \\ R_2 &= \{z \mid f_2(z) > \max\{f_1(z), f_3(z)\}\}, \\ R_3 &= \{z \mid f_3(z) > \max\{f_1(z), f_2(z)\}\}, \end{aligned}$$

defined by where each function is the largest of the three. Our goal is to find functions with $x^{(j)} \in R_1$, $y^{(j)} \in R_2$, and $z^{(j)} \in R_3$.

Pose this as a convex optimization problem. You may not use strict inequalities in your formulation.

Solve the specific instance of the 3-way separation problem given in `sep3way_data.m`, with the columns of the matrices \mathbf{X} , \mathbf{Y} and \mathbf{Z} giving the $x^{(j)}$, $j = 1, \dots, N$, $y^{(j)}$, $j = 1, \dots, M$ and $z^{(j)}$, $j = 1, \dots, P$. To save you the trouble of plotting data points and separation boundaries, we have included the plotting code in `sep3way_data.m`. (Note that `a1`, `a2`, `a3`, `b1` and `b2` contain arbitrary numbers; you should compute the correct values using `cvx`.)