

## Final exam

You may use any books, notes, or computer programs (*e.g.*, Matlab, CVX), but you may not discuss the exam with anyone until June 12, after everyone has taken the exam. The only exception is that you can ask us for clarification, via the course staff email address. We've tried pretty hard to make the exam unambiguous and clear, so we're unlikely to say much.

Please make a copy of your exam before handing it in.

**Please attach the cover page to the front of your exam.** Assemble your solutions in order (problem 1, problem 2, problem 3, ...), starting a new page for each problem. Put everything associated with each problem (*e.g.*, text, code, plots) together; do not attach code or plots at the end of the final.

**We will deduct points from long needlessly complex solutions, even if they are correct.** Our solutions are not long, so if you find that your solution to a problem goes on and on for many pages, you should try to figure out a simpler one. We expect neat, legible exams from everyone, including those enrolled Cr/N.

When a problem involves computation you must give all of the following: a clear discussion and justification of exactly what you did, the Matlab (or other) source code that produces the result, and the final numerical results or plots.

To download Matlab files containing problem data, you'll have to type the whole URL given in the problem into your browser; there are no links on the course web page pointing to these files. To get a file called `filename.m`, for example, you would retrieve

```
http://www.stanford.edu/class/ee364a/finalMfiles/filename.m
```

with your browser.

Please respect the honor code. Although we allow you to work on homework assignments in small groups, you cannot discuss the final with anyone, at least until everyone has taken it.

All problems have equal weight. Some are easier than they might appear at first glance.

Be sure you are using the most recent version of CVX, which is Version 1.2 (build 711). You can check this using the command `cvx_version`.

Be sure to check your email often during the exam, just in case we need to send out an important announcement.

1. *Fitting with censored data.* In some experiments there are two kinds of measurements or data available: The usual ones, in which you get a number (say), and *censored data*, in which you don't get the specific number, but are told something about it, such as a lower bound. A classic example is a study of lifetimes of a set of subjects (say, laboratory mice). For those who have died by the end of data collection, we get the lifetime. For those who have not died by the end of data collection, we do not have the lifetime, but we do have a lower bound, *i.e.*, the length of the study. These are the censored data values.

We wish to fit a set of data points,

$$(x^{(1)}, y^{(1)}), \dots, (x^{(K)}, y^{(K)}),$$

with  $x^{(k)} \in \mathbf{R}^n$  and  $y^{(k)} \in \mathbf{R}$ , with a linear model of the form  $y \approx c^T x$ . The vector  $c \in \mathbf{R}^n$  is the model parameter, which we want to choose. We will use a least-squares criterion, *i.e.*, choose  $c$  to minimize

$$J = \sum_{k=1}^K (y^{(k)} - c^T x^{(k)})^2.$$

Here is the tricky part: some of the values of  $y^{(k)}$  are censored; for these entries, we have only a (given) lower bound. We will re-order the data so that  $y^{(1)}, \dots, y^{(M)}$  are given (*i.e.*, uncensored), while  $y^{(M+1)}, \dots, y^{(K)}$  are all censored, *i.e.*, unknown, but larger than  $D$ , a given number. All the values of  $x^{(k)}$  are known.

- (a) Explain how to find  $c$  (the model parameter) and  $y^{(M+1)}, \dots, y^{(K)}$  (the censored data values) that minimize  $J$ .
- (b) Carry out the method of part (a) on the data values in `cens_fit_data.m`. Report  $\hat{c}$ , the value of  $c$  found using this method.

Also find  $\hat{c}_{\text{ls}}$ , the least-squares estimate of  $c$  obtained by simply ignoring the censored data samples, *i.e.*, the least-squares estimate based on the data

$$(x^{(1)}, y^{(1)}), \dots, (x^{(M)}, y^{(M)}).$$

The data file contains  $c_{\text{true}}$ , the true value of  $c$ , in the vector `c_true`. Use this to give the two relative errors

$$\frac{\|c_{\text{true}} - \hat{c}\|_2}{\|c_{\text{true}}\|_2}, \quad \frac{\|c_{\text{true}} - \hat{c}_{\text{ls}}\|_2}{\|c_{\text{true}}\|_2}.$$

2. *Deducing costs from samples of optimal decision.* A system (such as a firm or an organism) chooses a vector of values  $x$  as a solution of the LP

$$\begin{aligned} & \text{minimize} && c^T x \\ & \text{subject to} && Ax \succeq b, \end{aligned}$$

with variable  $x \in \mathbf{R}^n$ . You can think of  $x \in \mathbf{R}^n$  as a vector of activity levels,  $b \in \mathbf{R}^m$  as a vector of requirements, and  $c \in \mathbf{R}^n$  as a vector of costs or prices for the activities. With this interpretation, the LP above finds the cheapest set of activity levels that meet all requirements. (This interpretation is not needed to solve the problem.)

We suppose that  $A$  is known, along with a set of data

$$(b^{(1)}, x^{(1)}), \dots, (b^{(r)}, x^{(r)}),$$

where  $x^{(j)}$  is an optimal point for the LP, with  $b = b^{(j)}$ . (The solution of an LP need not be unique; all we say here is that  $x^{(j)}$  is *an* optimal solution.) Roughly speaking, we have samples of optimal decisions, for different values of requirements.

You *do not* know the cost vector  $c$ . Your job is to compute the tightest possible bounds on the costs  $c_i$  from the given data. More specifically, you are to find  $c_i^{\max}$  and  $c_i^{\min}$ , the maximum and minimum possible values for  $c_i$ , consistent with the given data.

Note that if  $x$  is optimal for the LP for a given  $c$ , then it is also optimal if  $c$  is scaled by any positive factor. To normalize  $c$ , then, we will assume that  $c_1 = 1$ . Thus, we can interpret  $c_i$  as the relative cost of activity  $i$ , compared to activity 1.

- (a) Explain how to find  $c_i^{\max}$  and  $c_i^{\min}$ . Your method can involve the solution of a reasonable number (not exponential in  $n$ ,  $m$  or  $r$ ) of convex or quasiconvex optimization problems.
- (b) Carry out your method using the data found in `deducing_costs_data.m`. You may need to determine whether individual inequality constraints are tight; to do so, use a tolerance threshold of  $\epsilon = 10^{-3}$ . (In other words: if  $a_k^T x - b_k \leq 10^{-3}$ , you can consider this inequality as tight.)  
Give the values of  $c_i^{\max}$  and  $c_i^{\min}$ , and make a very brief comment on the results.

3. *Power flow optimization with ‘ $N - 1$ ’ reliability constraint.* We model a network of power lines as a graph with  $n$  nodes and  $m$  edges. The power flow along line  $j$  is denoted  $p_j$ , which can be positive, which means power flows along the line in the direction of the edge, or negative, which means power flows along the line in the direction opposite the edge. (In other words, edge orientation is only used to determine the direction in which power flow is considered positive.) Each edge can support power flow in either direction, up to a given maximum capacity  $P_j^{\max}$ , *i.e.*, we have  $|p_j| \leq P_j^{\max}$ .

Generators are attached to the first  $k$  nodes. Generator  $i$  provides power  $g_i$  to the network. These must satisfy  $0 \leq g_i \leq G_i^{\max}$ , where  $G_i^{\max}$  is a given maximum power available from generator  $i$ . The power generation costs are  $c_i > 0$ , which are given; the total cost of power generation is  $c^T g$ .

Electrical loads are connected to the nodes  $k + 1, \dots, n$ . We let  $d_i \geq 0$  denote the demand at node  $k + i$ , for  $i = 1, \dots, n - k$ . We will consider these loads as given. In this simple model we will neglect all power losses on lines or at nodes. Therefore, power must balance at each node: the total power flowing into the node must equal the sum of the power flowing out of the node. This power balance constraint can be expressed as

$$Ap = \begin{bmatrix} -g \\ d \end{bmatrix},$$

where  $A \in \mathbf{R}^{n \times m}$  is the node-incidence matrix of the graph, defined by

$$A_{ij} = \begin{cases} +1 & \text{edge } j \text{ enters node } i, \\ -1 & \text{edge } j \text{ leaves node } i, \\ 0 & \text{otherwise.} \end{cases}$$

In the basic power flow optimization problem, we choose the generator powers  $g$  and the line flow powers  $p$  to minimize the total power generation cost, subject to the constraints listed above. The (given) problem data are the incidence matrix  $A$ , line capacities  $P^{\max}$ , demands  $d$ , maximum generator powers  $G^{\max}$ , and generator costs  $c$ .

In this problem we will add a basic (and widely used) reliability constraint, commonly called an ‘ $N - 1$  constraint’. ( $N$  is not a parameter in the problem; ‘ $N - 1$ ’ just means ‘all-but-one’.) This states that the system can still operate even if any one power line goes out, by re-routing the line powers. The case when line  $j$  goes out is called ‘failure contingency  $j$ ’; this corresponds to replacing  $P_j^{\max}$  with 0. The requirement is that there must exist a contingency power flow vector  $p^{(j)}$  that satisfies all the constraints above, with  $p_j^{(j)} = 0$ , using the same given generator powers. (This corresponds to the idea that power flows can be re-routed quickly, but generator power can only be changed more slowly.) The ‘ $N - 1$  reliability constraint’ requires that for each line, there is a contingency power flow vector. The ‘ $N - 1$  reliability constraint’ is (implicitly) a constraint on the generator powers.

The questions below concern the specific instance of this problem with data given in `rel_pwr_flow_data.m`. (Executing this file will also generate a figure showing the

network you are optimizing.) Especially for part (b) below, you must explain exactly how you set up the problem as a convex optimization problem.

- (a) *Nominal optimization.* Find the optimal generator and line power flows for this problem instance (without the  $N - 1$  reliability constraint). Report the optimal cost and generator powers. (You do not have to give the power line flows.)
- (b) *Nominal optimization with  $N - 1$  reliability constraint.* Minimize the nominal cost, but you must choose generator powers that meet the  $N - 1$  reliability requirement as well. Report the optimal cost and generator powers. (You do not have to give the nominal power line flows, or any of the contingency flows.)

4. *Spectrum analysis with quantized measurements.* A sample is made up of  $n$  compounds, in quantities  $q_i \geq 0$ , for  $i = 1, \dots, n$ . Each compound has a (nonnegative) spectrum, which we represent as a vector  $s^{(i)} \in \mathbf{R}_+^m$ , for  $i = 1, \dots, n$ . (Precisely what  $s^{(i)}$  means won't matter to us.) The spectrum of the sample is given by  $s = \sum_{i=1}^n q_i s^{(i)}$ . We can write this more compactly as  $s = Sq$ , where  $S \in \mathbf{R}^{m \times n}$  is a matrix whose columns are  $s^{(1)}, \dots, s^{(n)}$ .

Measurement of the spectrum of the sample gives us an interval for each spectrum value, *i.e.*,  $l, u \in \mathbf{R}_+^m$  for which

$$l_i \leq s_i \leq u_i, \quad i = 1, \dots, m.$$

(We don't directly get  $s$ .) This occurs, for example, if our measurements are quantized. Given  $l$  and  $u$  (and  $S$ ), we cannot in general deduce  $q$  exactly. Instead, we ask you to do the following. For each compound  $i$ , find the range of possible values for  $q_i$  consistent with the spectrum measurements. We will denote these ranges as  $q_i \in [q_i^{\min}, q_i^{\max}]$ . Your job is to find  $q_i^{\min}$  and  $q_i^{\max}$ .

Note that if  $q_i^{\min}$  is large, we can confidently conclude that there is a significant amount of compound  $i$  in the sample. If  $q_i^{\max}$  is small, we can confidently conclude that there is not much of compound  $i$  in the sample.

- (a) Explain how to find  $q_i^{\min}$  and  $q_i^{\max}$ , given  $S$ ,  $l$ , and  $u$ .
- (b) Carry out the method of part (a) for the problem instance given in `spectrum_data.m`. (Executing this file defines the problem data, and plots the compound spectra and measurement bounds.) Plot the minimum and maximum values versus  $i$ , using the commented out code in the data file. Report your values for  $q_4^{\min}$  and  $q_4^{\max}$ .

5. *Optimal detector design.* We adopt here the notation of §7.3 of the book. Explain how to design a (possibly randomized) detector that minimizes the worst-case probability of our estimate being off by more than one,

$$P_{\text{wc}} = \max_{\theta} \mathbf{Prob}(|\hat{\theta} - \theta| \geq 2).$$

(The probability above is under the distribution associated with  $\theta$ .)

Carry out your method for the problem instance with data in `off_by_one_det_data.m`. Give the optimal detection probability matrix  $D$ . Compare the optimal worst-case probability  $P_{\text{wc}}^*$  with the worst-case probability  $P_{\text{wc}}^{\text{ml}}$  obtained using a maximum-likelihood detector.

6. *Optimal vehicle speed scheduling.* A vehicle (say, an airplane) travels along a fixed path of  $n$  segments, between  $n + 1$  waypoints labeled  $0, \dots, n$ . Segment  $i$  starts at waypoint  $i - 1$  and terminates at waypoint  $i$ . The vehicle starts at time  $t = 0$  at waypoint 0. It travels over each segment at a constant (nonnegative) speed;  $s_i$  is the speed on segment  $i$ . We have lower and upper limits on the speeds:  $s^{\min} \preceq s \preceq s^{\max}$ . The vehicle does not stop at the waypoints; it simply proceeds to the next segment. The travel distance of segment  $i$  is  $d_i$  (which is positive), so the travel time over segment  $i$  is  $d_i/s_i$ . We let  $\tau_i$ ,  $i = 1, \dots, n$ , denote the time at which the vehicle arrives at waypoint  $i$ . The vehicle is required to arrive at waypoint  $i$ , for  $i = 1, \dots, n$ , between times  $\tau_i^{\min}$  and  $\tau_i^{\max}$ , which are given. The vehicle consumes fuel over segment  $i$  at a rate that depends on its speed,  $\Phi(s_i)$ , where  $\Phi$  is positive, increasing, and convex, and has units of kg/s. You are given the data  $d$  (segment travel distances),  $s^{\min}$  and  $s^{\max}$  (speed bounds),  $\tau^{\min}$  and  $\tau^{\max}$  (waypoint arrival time bounds), and the fuel use function  $\Phi : \mathbf{R} \rightarrow \mathbf{R}$ . You are to choose the speeds  $s_1, \dots, s_n$  so as to minimize the total fuel consumed in kg.

- (a) Show how to pose this as a convex optimization problem. If you introduce new variables, or change variables, you must explain how to recover the optimal speeds from the solution of your problem. If convexity of the objective or any constraint function in your formulation is not obvious, explain why it is convex.
- (b) Carry out the method of part (a) on the problem instance with data in `veh_speed_sched_data.m`. Use the fuel use function  $\Phi(s_i) = as_i^2 + bs_i + c$  (the parameters  $a$ ,  $b$ , and  $c$  are defined in the data file). What is the optimal fuel consumption? Plot the optimal speed versus segment, using the matlab command `stairs` to better show constant speed over the segments.

7. *Optimal jamming power allocation.* A set of  $n$  jammers transmit with (nonnegative) powers  $p_1, \dots, p_n$ , which are to be chosen subject to the constraints

$$p \succeq 0, \quad Fp \preceq g.$$

The jammers produce interference power at  $m$  receivers, given by

$$d_i = \sum_{j=1}^n G_{ij} p_j, \quad i = 1, \dots, m,$$

where  $G_{ij}$  is the (nonnegative) channel gain from jammer  $j$  to receiver  $i$ .

Receiver  $i$  has capacity (in bits/s) given by

$$C_i = \alpha \log(1 + \beta_i / (\sigma_i^2 + d_i)), \quad i = 1, \dots, m,$$

where  $\alpha$ ,  $\beta_i$ , and  $\sigma_i$  are positive constants. (Here  $\beta_i$  is proportional to the signal power at receiver  $i$  and  $\sigma_i^2$  is the receiver  $i$  self-noise, but you won't need to know this to solve the problem.)

Explain how to choose  $p$  to *minimize* the sum channel capacity,  $C = C_1 + \dots + C_m$ , using convex optimization. (This corresponds to the most effective jamming, given the power constraints.) The problem data are  $F$ ,  $g$ ,  $G$ ,  $\alpha$ ,  $\beta_i$ ,  $\sigma_i$ .

If you change variables, or transform your problem in any way that is not obvious (for example, you form a relaxation), you must explain fully how your method works, and why it gives the solution. If your method relies on any convex functions that we have not encountered before, you must show that the functions are convex.

*Please note that the EE364a staff does not endorse jamming, optimal or otherwise.*

8. *Conformal mapping via convex optimization.* Suppose that  $\Omega$  is a closed bounded region in  $\mathbf{C}$  with no holes (*i.e.*, it is simply connected). The Riemann mapping theorem states that there exists a conformal mapping  $\varphi$  from  $\Omega$  onto  $D = \{z \in \mathbf{C} \mid |z| \leq 1\}$ , the unit disk in the complex plane. (This means that  $\varphi$  is an analytic function, and maps  $\Omega$  one-to-one onto  $D$ .)

One proof of the Riemann mapping theorem is based on an infinite dimensional optimization problem. We choose a point  $a \in \mathbf{int} \Omega$  (the interior of  $\Omega$ ). Among all analytic functions that map  $\partial\Omega$  (the boundary of  $\Omega$ ) into  $D$ , we choose one that maximizes the magnitude of the derivative at  $a$ . Amazingly, it can be shown that this function is a conformal mapping of  $\Omega$  onto  $D$ .

We can use this theorem to construct an approximate conformal mapping, by sampling the boundary of  $\Omega$ , and by restricting the optimization to a finite-dimensional subspace of analytic functions. Let  $b_1, \dots, b_N$  be a set of points in  $\partial\Omega$  (meant to be a sampling of the boundary). We will search only over polynomials of degree up to  $n$ ,

$$\hat{\varphi}(z) = \alpha_1 z^n + \alpha_2 z^{n-1} + \dots + \alpha_n z + \alpha_{n+1},$$

where  $\alpha_1, \dots, \alpha_{n+1} \in \mathbf{C}$ . With these approximations, we obtain the problem

$$\begin{array}{ll} \text{maximize} & |\hat{\varphi}'(a)| \\ \text{subject to} & |\hat{\varphi}(b_i)| \leq 1, \quad i = 1, \dots, N, \end{array}$$

with variables  $\alpha_1, \dots, \alpha_{n+1} \in \mathbf{C}$ . The problem data are  $b_1, \dots, b_N \in \partial\Omega$  and  $a \in \mathbf{int} \Omega$ .

- (a) Explain how to solve the problem above via convex or quasiconvex optimization.
- (b) Carry out your method on the problem instance given in `conf_map_data.m`. This file defines the boundary points  $b_i$  and plots them. It also contains code that will plot  $\hat{\varphi}(b_i)$ , the boundary of the mapped region, once you provide the values of  $\alpha_j$ ; these points should be very close to the boundary of the unit disk. (Please turn in this plot, and give us the values of  $\alpha_j$  that you find.) The function `polyval` may be helpful.

*Remarks.*

- We've been a little informal in our mathematics here, but it won't matter.
- You do not need to know any complex analysis to solve this problem; we've told you everything you need to know.
- A basic result from complex analysis tells us that  $\hat{\varphi}$  is one-to-one if and only if the image of the boundary does not 'loop over' itself. (We mention this just for fun; we're not asking you to verify that the  $\hat{\varphi}$  you find is one-to-one.)