

Final exam solutions

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}_{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_{true} , the true value of c , in the vector `c_true`. Use this to give the two relative errors

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

Solution.

- (a) The trick is to introduce dummy variables to serve as placeholders for the measurements which are censored, *i.e.*, $y^{(k)}$, $k = M + 1, \dots, K$. By introducing the dummy variables $(z^{(1)}, \dots, z^{(K-M)})$, we get the QP

$$\begin{aligned} & \text{minimize} && \sum_{k=1}^M \left(y^{(k)} - c^T x^{(k)} \right)^2 + \sum_{k=M+1}^K \left(z^{(k-M)} - c^T x^{(k)} \right)^2 \\ & \text{subject to} && z^{(k)} \geq D, \quad k = 1, \dots, K - M, \end{aligned}$$

where the variables are c and $z^{(k)}$, $k = 1, \dots, K - M$.

- (b) The following code solves the problem

```
cens_fit_data;

% Using censored data method
cvx_begin
    variables c(n) z(K-M)
    minimize(sum_square(y-X(:,1:M) '*c)+sum_square(z-X(:,M+1:K) '*c))
    subject to
        z >= D
cvx_end
c_cens = c;

% Comparison to least squares method, ignoring all censored data
cvx_begin
    variable c(n)
    minimize(sum_square(y-X(:,1:M) '*c))
cvx_end
c_ls = c;

[c_true c_cens c_ls]
cens_relerr = norm(c_cens-c_true)/norm(c_true)
ls_relerr = norm(c_ls-c_true)/norm(c_true)
```

We get the following estimates of our parameter vector:

```
[c_true c_cens c_ls] =
-0.4326   -0.2946   -0.3476
-1.6656   -1.7541   -1.7955
 0.1253    0.2589    0.2000
 0.2877    0.2241    0.1672
-1.1465   -0.9917   -0.8357
 1.1909    1.3018    1.3005
 1.1892    1.4262    1.8276
```

-0.0376	-0.1554	-0.5612
0.3273	0.3785	0.3686
0.1746	0.2261	-0.0454
-0.1867	-0.0826	-0.1096
0.7258	1.0427	1.5265
-0.5883	-0.4648	-0.4980
2.1832	2.1942	2.4164
-0.1364	-0.3586	-0.5563
0.1139	-0.1973	-0.3701
1.0668	1.0194	0.9900
0.0593	-0.1186	-0.2539
-0.0956	-0.1211	-0.1762
-0.8323	-0.7523	-0.4349

This gives a relative error of 0.1784 for \hat{c} , and a relative error of 0.3907 for \hat{c}_{1s} .

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.

Solution. A feasible point x is optimal for the LP if and only if there exists a $\lambda \succeq 0$ with $c = A^T \lambda$ and $\lambda_k (a_k^T x - b_k) = 0$, $k = 1, \dots, m$, where a_k^T is the k th row of A . So all we know about $x^{(j)}$ is that there exists a $\lambda^{(j)} \succeq 0$ with $c = A^T \lambda^{(j)}$ and $\lambda_k^{(j)} (a_k^T x^{(j)} - b_k^{(j)}) = 0$, $k = 1, \dots, m$. But we don't know the vectors $\lambda^{(j)}$.

We can express this as follows: c is a possible cost vector if and only if there exists $\lambda^{(j)}$, $j = 1, \dots, r$, such that

$$\begin{aligned} \lambda^{(j)} &\succeq 0, & j &= 1, \dots, r \\ c &= A^T \lambda^{(j)}, & j &= 1, \dots, r \\ \lambda_k^{(j)} (a_k^T x^{(j)} - b_k^{(j)}) &= 0, & j &= 1, \dots, r, \quad k = 1, \dots, m. \end{aligned}$$

Here we know $a_k, x^{(j)}, b^{(j)}$; we don't know c or $\lambda^{(j)}$. But careful examination of these equations and inequalities shows that they are in fact a set of linear equalities and inequalities on the unknowns, c and $\lambda^{(j)}$. So we can minimize (or maximize) over c_i by solving the optimization problem:

$$\begin{aligned} & \text{minimize} && c_i \\ & \text{subject to} && \lambda^{(j)} \succeq 0, \quad j = 1, \dots, r \\ & && c = A^T \lambda^{(j)}, \quad j = 1, \dots, r \\ & && \lambda_k^{(j)} (a_k^T x^{(j)} - b_k^{(j)}) = 0, \quad j = 1, \dots, r, \quad k = 1, \dots, m \\ & && c_1 = 1, \end{aligned}$$

with variables c and $\lambda^{(1)}, \dots, \lambda^{(r)}$. The solution gives us c_i^{\min} . If you look at this carefully, you'll see that it is an LP. $\lambda_k^{(j)} (a_k^T x^{(j)} - b_k^{(j)}) = 0$ reduces to $\lambda_k^{(j)} \geq 0$ when $a_k^T x^{(j)} = b_k^{(j)}$, and $\lambda_k^{(j)} = 0$ when $a_k^T x^{(j)} > b_k^{(j)}$. If we maximize instead of minimize, we get c_i^{\max} . Therefore, we solve a set of $2(n-1)$ LPs: one to maximize each c_i and one to minimize each c_i , for $i = 2, \dots, n$.

In the problem instance, $m = 10, n = 5$, and $r = 10$. Our method requires the solution of 8 LPs. The following Matlab code solves the problem.

```
%this script solves the deducing costs problem
deducing_costs_data;

cmin = ones(n,1);
cmax = ones(n,1);
for i = 2:n
% calculate lower bound cmin(i)
cvx_begin
variables c(n) lambdas(m,r)
    minimize(c(i))
    c(1) == 1;
    lambdas >= 0;
    lambdas(abs(A*X-B)>=1e-3) == 0;
    c*ones(1,r) == A'*lambdas;
cvx_end
cmin(i) = cvx_optval;

% calculate upper bound cmax(i)
cvx_begin
variables c(n) lambdas(m,r)
    maximize(c(i))
    c(1) == 1;
    lambdas >= 0;
```

```

    lambdas(abs(A*X-B)>=1e-3) == 0;
    c*ones(1,r) == A'*lambdas;
cvx_end
cmax(i) = cvx_optval;
end

disp('cmin c_true cmax:')
[cmin c_true cmax]

```

The bounds, displayed as [cmin c_true cmax], are

```

ans =

    1.0000    1.0000    1.0000
    0.6477    1.3183    1.7335
    0.0609    0.3077    0.4044
    0.0161    1.0190    1.5127
    0.7378    0.8985    1.0723

```

The tightest bounds are on c_3 and c_5 and the weakest bound is on c_4 .

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.)

Solution.

- (a) To find the optimal generators and line power flows we solve the LP

$$\begin{aligned} & \text{minimize} && c^T g \\ & \text{subject to} && Ap = \begin{bmatrix} -g \\ d \end{bmatrix} \\ & && -P^{\max} \preceq p \preceq P^{\max} \\ & && 0 \preceq g \preceq G^{\max}, \end{aligned}$$

with variables g and p .

- (b) To handle the additional $N - 1$ reliability constraint, we must introduce a set of power flow vectors for each contingency. We then solve the LP

$$\begin{aligned} & \text{minimize} && c^T g \\ & \text{subject to} && Ap^{(j)} = \begin{bmatrix} -g \\ d \end{bmatrix}, \quad j = 1, \dots, m \\ & && p_j^{(j)} = 0, \quad j = 1, \dots, m \\ & && -P^{\max} \preceq p^{(j)} \preceq P^{\max}, \quad j = 1, \dots, m \\ & && 0 \preceq g \preceq G^{\max}, \end{aligned}$$

with variables $g \in \mathbf{R}^k$ and $p^{(1)}, \dots, p^{(m)} \in \mathbf{R}^m$.

The optimal costs are 44.60 and 56.20 for parts (a) and (b) respectively. The optimal generator powers are

$$g_{\text{nom}} = \begin{bmatrix} 3.0 \\ 0.0 \\ 2.3 \\ 7.0 \end{bmatrix}, \quad g_{\text{rel}} = \begin{bmatrix} 1.9 \\ 1.9 \\ 4.0 \\ 4.5 \end{bmatrix}.$$

We can say a little about these results. In the nominal case, it turns out that the line capacities are not tight; that is, we have no congestion on the transmission lines. So

we must select a set of generator powers that deliver the required power, which is the sum of the demands (12.32 power units). To do this most efficiently, we start with generator 4, which has the lowest cost, and we set it to its maximum, which gives us a total of 7 power units. We then go the second cheapest generator, generator 1, and set it to its maximum (3 power units), which gives us a total of 10 power units. Finally, we go to the third cheapest generator, generator 3, and use it to satisfy the remaining demand.

When we impose the additional $N - 1$ reliability constraint, we are forced to shift some generation to more expensive generators.

The following matlab code solves parts (a) and (b).

```
rel_pwr_flow_data;

% nominal case
cvx_begin
    variables p(m) g_nom(k)
    minimize (c'*g_nom)
    subject to
        A*p == [-g_nom;d];
        abs(p) <= Pmax;
        g_nom <= Gmax;
        g_nom >= 0;
cvx_end
nom_cost = cvx_optval;

% N-1 case
cvx_begin
    variables P(m,m) g_rel(k)
    minimize (c'*g_rel)
    subject to
        A*P == [-g_rel;d]*ones(1,m);
        diag(P) == 0;
        abs(P) <= Pmax*ones(1,m);
        g_rel <= Gmax;
        g_rel >= 0;
cvx_end
rel_cost = cvx_optval;

% show nominal optimal and reliable optimal cost
[nom_cost rel_cost]
% show nominal optimal and reliable optimal generator powers
[g_nom g_rel]
```

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} .

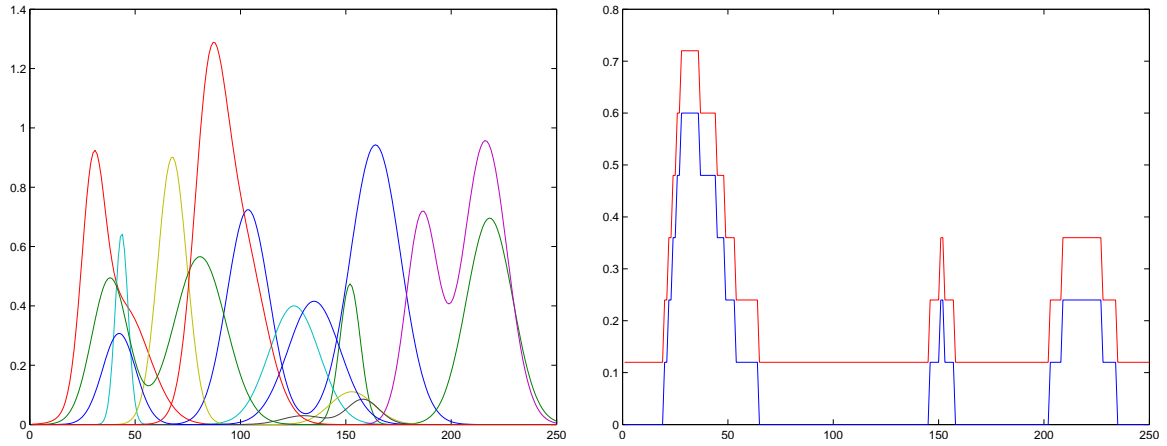
Solution. To find q_i^{\min} , we solve the convex optimization problem

$$\begin{aligned} & \text{minimize} && q_i \\ & \text{subject to} && l \preceq Sq \preceq u, \quad q \succeq 0, \end{aligned}$$

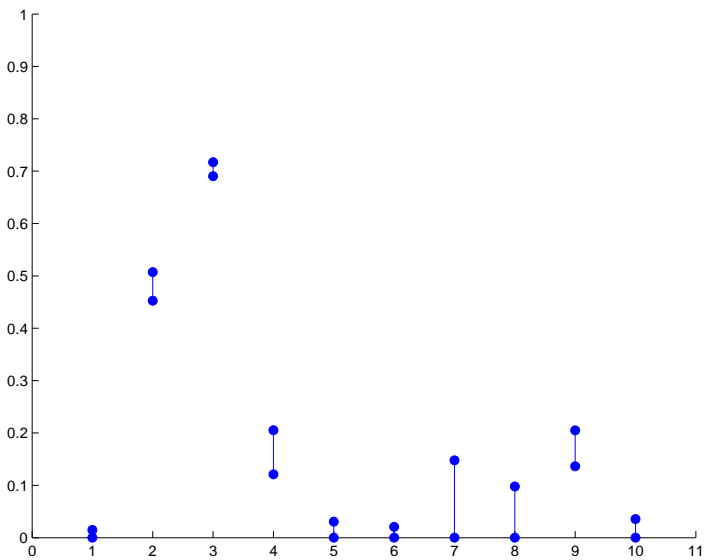
with variable $q \in \mathbf{R}^n$. Then we set $q_i^{\min} = q_i^*$. Similarly to find q_i^{\max} , we solve the convex optimization problem

$$\begin{aligned} & \text{maximize} && q_i \\ & \text{subject to} && l \preceq Sq \preceq u, \quad q \succeq 0, \end{aligned}$$

and we set $q_i^{\max} = q_i^*$. Here is a plot of the spectra of the compounds $s^{(1)}, \dots, s^{(n)}$, alongside the lower and upper bounds u and l .



Here is a plot of q_i^{\min} and q_i^{\max} , for $i = 1, \dots, n$.



For this particular instance, $q_4^{\min} = 0.121$, $q_4^{\max} = 0.205$.

The matlab code to solve this problem is given below.

```
spectrum_data;

qmin = zeros(n,1); qmax = zeros(n,1);
for i = 1:n
    cvx_begin
        variable q(n)
        l <= S*q; u >= S*q;
        q >= 0;
        minimize(q(i))
    end
    qmin(i) = qmin; qmax(i) = qmax;
end
```

```

    cvx_end
    qmin(i) = q(i);
    cvx_begin
        variable q(n)
        l <= S*q; u >= S*q;
        q >= 0;
        maximize(q(i))
    cvx_end
    qmax(i) = q(i);
end

% print quantity bounds
[qmin qmax]

figure; hold on;
for i = 1:n
    plot([i,i],[qmin(i),qmax(i)],'o-','MarkerFaceColor','b');
end
axis([0,11,0,1]);
%print('-depsc','spect_minmax.eps');

```

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 θ .)

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_{wc}^* with the worst-case probability $P_{\text{wc}}^{\text{ml}}$ obtained using a maximum-likelihood detector.

Solution. For $\theta = j$ we have

$$\mathbf{Prob}(|\hat{\theta} - \theta| \geq 2) = \sum_{|i-j| \geq 2} D_{ij} = 1 - D_{j+1,j} - D_{jj} - D_{j-1,j},$$

where we interpret D_{ij} as zero for $i = 0$ and $i = m + 1$. Therefore we find our detector by solving the problem

$$\begin{aligned} & \text{minimize} && \max_j \left(\sum_{|i-j| \geq 2} D_{ij} \right) \\ & \text{subject to} && D = [t_1 \cdots t_n] P \\ & && t_k \succeq 0, \quad \mathbf{1}^T t_k = 1, \quad k = 1, \dots, n, \end{aligned}$$

with variables t_1, \dots, t_n . This problem is evidently convex, since the constraints are linear equalities and inequalities, and the objective is a piecewise linear convex function.

The optimal detection probability matrix for the given problem instance is

$$D = \begin{bmatrix} 0.2466 & 0.2816 & 0.1616 & 0.0911 & 0.1102 \\ 0.4816 & 0.3855 & 0.4761 & 0.1807 & 0.1601 \\ 0.0046 & 0.0611 & 0.1691 & 0.0662 & 0.0014 \\ 0.1477 & 0.1687 & 0.1218 & 0.3664 & 0.3940 \\ 0.1195 & 0.1031 & 0.0715 & 0.2956 & 0.3343 \end{bmatrix}.$$

The worst-case probability that we are off by more than one is $P_{\text{wc}}^* = 0.27$. The worst-case probability of being off by more than one using a maximum-likelihood detector is $P_{\text{wc}}^{\text{ml}} = 0.74$, nearly a factor of three worse than the optimal detector.

The following matlab code finds the optimal probability detection matrix D .

```
off_by_one_det_data;

% off by one randomized detector
cvx_begin
    variables D(m,m) T(m,n)
```

```

    % d(i) is prob we're off by zero or one, with hypothesis i
    d = cvx(zeros(m,1));
    d(1) = D(1,1)+D(2,1);
    for i = 2:m-1
        d(i) = D(i-1,i)+D(i,i)+D(i+1,i);
    end
    d(m) = D(m-1,m)+D(m,m);
    T >= 0; sum(T) == 1;
    D == T*P; % detection probability matrix
    minimize (max(1-d))
cvx_end
Popt = cvx_optval; % max prob we are off by more than one

% maximum likelihood (ML) detector
% first we form the detector matrix, T_ml
Tml = zeros(m,n);
for i=1:n
    j = find(P(i,)==max(P(i,:)));
    Tml(j,i) = 1;
end

% and now the ML probability detector matrix, D_ml
Dml = Tml*P;

% worst case probability using maximum-likelihood detector
% dml(i) is prob we're off by zero or one, with hypothesis i
dml(1) = Dml(1,1)+Dml(2,1);
for i=2:m-1
    dml(i) = Dml(i-1,i)+Dml(i,i)+Dml(i+1,i);
end
dml(m) = Dml(m-1,m)+Dml(m,m);
Pml = max(1-dml); % ML max prob we are off by more than one

disp('the optimal detection prob matrix');
disp(D)
disp('max prob using optimal detector');
disp(Popt)
disp('max prob using ML detector');
disp(Pml)

```

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 τ_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 τ_i^{\min} and τ_i^{\max} , which are given. The vehicle consumes fuel over segment i at a rate that depends on its speed, $\Phi(s_i)$, where Φ 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), τ^{\min} and τ^{\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.

Solution.

- (a) The fuel consumed over the i th segment is $(d_i/s_i)\Phi(s_i)$, so the total fuel used is $\sum_{i=1}^n (d_i/s_i)\Phi(s_i)$. The vehicle arrives at waypoint i at time $\tau_i = \sum_{j=1}^i (d_j/s_j)$. Thus our problem is

$$\begin{aligned} & \text{minimize} && \sum_{i=1}^n (d_i/s_i)\Phi(s_i) \\ & \text{subject to} && s_i^{\min} \leq s_i \leq s_i^{\max} \quad i = 1, \dots, n \\ & && \tau_i^{\min} \leq \sum_{j=1}^i (d_j/s_j) \leq \tau_i^{\max} \quad i = 1, \dots, n, \end{aligned}$$

with variables s_1, \dots, s_n .

In this form, this is *not* a convex problem: the objective function need not be convex in s_i , and the inequalities $\tau_i^{\min} \leq \sum_{j=1}^i (d_j/s_j)$ are not convex.

However, we can formulate this as a convex problem by making a change of variables. We formulate the problem using the transit times of the segments, t_i , as the optimization variable, where $t_i = d_i/s_i$. (We then have $s_i = d_i/t_i$.) Our

problem can be written as

$$\begin{aligned} & \text{minimize} && \sum_{i=1}^n t_i \Phi(d_i/t_i) \\ & \text{subject to} && d_i/s_i^{\max} \leq t_i \leq d_i/s_i^{\min} \quad i = 1, \dots, n \\ & && \tau_i^{\min} \leq \sum_{j=1}^i t_j \leq \tau_i^{\max} \quad i = 1, \dots, n, \end{aligned}$$

with variables t_1, \dots, t_n .

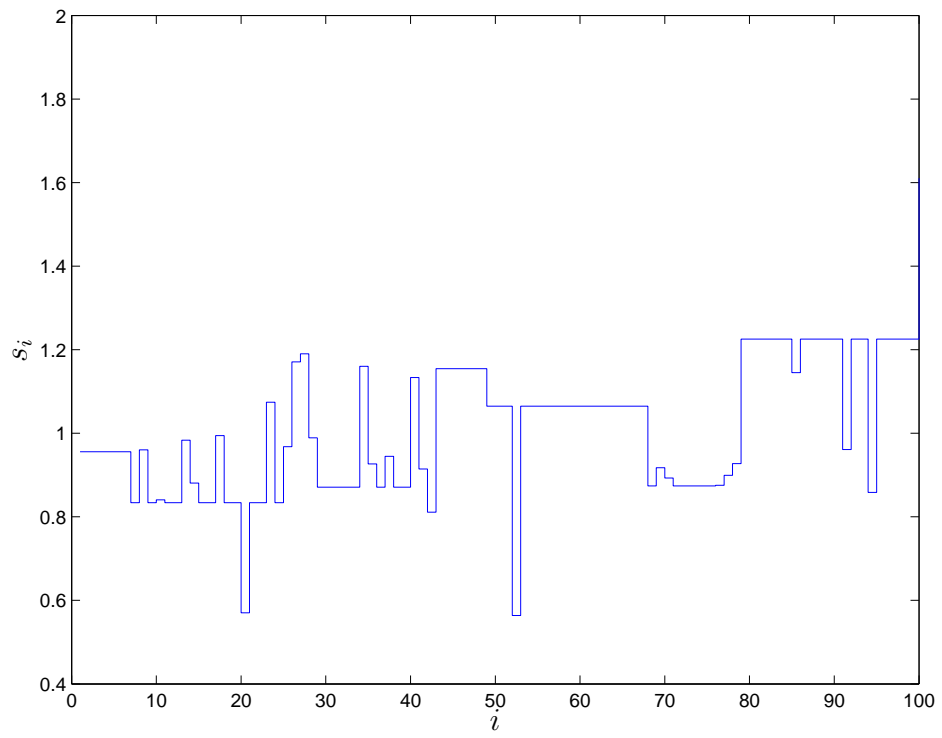
This is a convex problem. The function $t_i \Phi(d_i/t_i)$, the perspective of Φ , is convex jointly in d_i and t_i ; in particular, it is convex in t_i . Therefore the objective function is convex, since it is a positive weighted sum of convex functions. The constraints are all linear in t . Once we have solved the problem for t_i^* we recover the optimal speeds using $s_i^* = d_i/t_i^*$.

- (b) The optimal fuel consumption is 2617.83 kg. The code below solves the problem:

```
% solution to vehicle speed scheduling problem
veh_speed_sched_data

cvx_begin
    variable t(n)
    minimize(sum(a*d.^2.*inv_pos(t)+b*d+c*t))
    t<=d./smin;
    t>=d./smax;
    tau_min<=cumsum(t);
    tau_max>=cumsum(t);
cvx_end
s=d./t;

stairs(s)
title('speed over segment');xlabel('i');ylabel('si')
%print('-depsc','veh_speed.eps')
```



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 α , β_i , and σ_i are positive constants. (Here β_i is proportional to the signal power at receiver i and σ_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 , α , β_i , σ_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.

Solution. This is almost a trick question, because it is so easy: there is no need to change variables, or introduce any relaxation. We'll show that the following problem is convex:

$$\begin{aligned} & \text{minimize} && C \\ & \text{subject to} && p \succeq 0, \quad Fp \preceq g. \end{aligned}$$

We observe that the function $f(x) = \log(1 + 1/x)$ is convex for $x > 0$: we have

$$f'(x) = -\frac{1}{x^2 + x},$$

which evidently is increasing in x , so $f''(x) > 0$.

Here's another proof that $f(x) = \log(1 + 1/x)$ is convex: it is the composition of $h(u_1, u_2) = \log(e^{u_1} + e^{u_2})$, which is increasing and convex, with $u_1 = 0$ and $u_2 = -\log x$, which are convex. Thus,

$$h(g(x)) = \log(e^0 + e^{-\log x}) = \log(1 + 1/x)$$

is convex.

And, here's yet another, just for fun. We write

$$\log(1 + 1/x) = -\log(x/(x + 1)) = -\log(1 - 1/(x + 1)),$$

the composition of $h(u) = -\log(1 - u)$, which is convex and increasing, with $u = 1/(x + 1)$, which is convex.

For rows g_i^T of G , we write

$$C_i = \alpha f((g_i^T p + \sigma_i^2)/\beta_i),$$

which is convex in p , since the argument to f is affine in p . It follows that C is a convex function of p .

8. *Conformal mapping via convex optimization.* Suppose that Ω 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 φ from Ω onto $D = \{z \in \mathbf{C} \mid |z| \leq 1\}$, the unit disk in the complex plane. (This means that φ is an analytic function, and maps Ω 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 Ω). Among all analytic functions that map $\partial\Omega$ (the boundary of Ω) 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 Ω onto D .

We can use this theorem to construct an approximate conformal mapping, by sampling the boundary of Ω , 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

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

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 α_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 α_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.)

Solution. The constraint functions can be written as

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

which are evidently convex in α , since $\hat{\varphi}(b_i)$ is a (complex) affine function of α . The objective as stated is

$$|\hat{\varphi}'(a)| = \left| \alpha_1 n a^{n-1} + \alpha_2 (n-1) a^{n-2} + \cdots + \alpha_n \right|,$$

which is not concave in α . But we observe that if α satisfies the constraints, then so does $\gamma\alpha$, where $|\gamma| = 1$. Therefore we can just as well maximize $\Re\hat{\varphi}'(a)$, or even insist that $\hat{\varphi}'(a)$ be real (either of these works). This yields

$$\begin{aligned} & \text{maximize} && \Re(\alpha_1 n a^{n-1} + \alpha_2 (n-1) a^{n-2} + \cdots + \alpha_n) \\ & \text{subject to} && \left| \alpha_1 b_i^n + \alpha_2 b_i^{n-1} + \cdots + \alpha_n b_i + \alpha_{n+1} \right| \leq 1, \quad i = 1, \dots, N, \end{aligned}$$

which is convex (in fact, an SOCP).

The code which solves this problem is given below:

```
% approximate conformal mapping via convex optimization
conf_map_data;

cvx_begin
    variable alpha(n+1) complex; % polynomial coefficients
    alpha_prime = alpha(1:n).*(n:-1:1)'; % coefficients of derivative
    maximize (real(polyval(alpha_prime,a)));
    norm(polyval(alpha,b),Inf) <= 1; % must map boundary points into unit disk
cvx_end
w = polyval(alpha,b); % (boundary of) mapped region

subplot(1,2,1)
plot(real(b),imag(b));
title('Boundary of region');
axis equal;
axis([-1.5 1.5 -1.5 1.5]);

subplot(1,2,2)
plot(real(w),imag(w),'b',cos(theta),sin(theta),'g');
title('Boundary of mapped region');
axis equal;
axis([-1.5 1.5 -1.5 1.5]);

%print('-depsc','conf_map.eps');
```

The coefficients α of the conformal mapping polynomial are

alpha =

-0.0015 + 0.0051i
-0.0074 - 0.0167i
0.0107 + 0.0091i
0.0206 - 0.0281i
0.0017 + 0.0740i
0.0031 - 0.0241i
-0.1263 + 0.0543i
0.0434 - 0.1671i
-0.1122 - 0.0056i
1.0430 - 0.0000i
0.0016 - 0.0026i

This gives us the following conformal mapping:

