EE364a Homework 6 solutions

6.9 *Minimax rational function fitting.* Show that the following problem is quasiconvex:

$$\text{minimize} \quad \max_{i=1,\ldots,k} \left| \frac{p(t_i)}{q(t_i)} - y_i \right|$$

where

$$p(t) = a_0 + a_1 t + a_2 t^2 + \cdots + a_m t^m, \qquad q(t) = 1 + b_1 t + \cdots + b_n t^n,$$

and the domain of the objective function is defined as

$$D = \{(a, b) \in \mathbf{R}^{m+1} \times \mathbf{R}^n \mid q(t) > 0,\ \alpha \le t \le \beta\}.$$

In this problem we fit a rational function $p(t)/q(t)$ to given data, while constraining the denominator polynomial to be positive on the interval $[\alpha, \beta]$. The optimization variables are the numerator and denominator coefficients a_i, b_i. The interpolation points $t_i \in [\alpha, \beta]$, and desired function values y_i, $i = 1, \ldots, k$, are given.

Solution. Let's show the objective is quasiconvex. Its domain is convex. Since $q(t_i) > 0$ for $i = 1, \ldots, k$, we have

$$\max_{i=1,\ldots,k} |p(t_i)/q(t_i) - y_i| \le \gamma$$

if and only if

$$|p(t_i) - y_i q(t_i)| \le \gamma q(t_i), \quad i = 1, \ldots, k,$$

which defines a convex set in the variables a and b, since the lefthand side is convex, and the righthand side is linear. We can further express these inequalities as a set of $2k$ linear inequalities,

$$-\gamma q(t_i) \le p(t_i) - y_i q(t_i) \le \gamma q(t_i), \quad i = 1, \ldots, k.$$

Solutions to additional exercises

1. *Minimax rational fit to the exponential.* (See exercise 6.9.) We consider the specific problem instance with data

$$t_i = -3 + 6(i-1)/(k-1), \quad y_i = e^{t_i}, \quad i = 1, \ldots, k,$$

where $k = 201$. (In other words, the data are obtained by uniformly sampling the exponential function over the interval $[-3, 3]$.) Find a function of the form

$$f(t) = \frac{a_0 + a_1 t + a_2 t^2}{1 + b_1 t + b_2 t^2}$$

that minimizes $\max_{i=1,\ldots,k} |f(t_i) - y_i|$. (We require that $1 + b_1 t_i + b_2 t_i^2 > 0$ for $i = 1, \ldots, k$.)

Find optimal values of $a_0,\ a_1,\ a_2,\ b_1,\ b_2$, and give the optimal objective value, computed to an accuracy of 0.001. Plot the data and the optimal rational function fit on the same plot. On a different plot, give the fitting error, *i.e.*, $f(t_i) - y_i$.

Hint. You can use `strcmp(cvx_status,'Solved')`, after `cvx_end`, to check if a feasibility problem is feasible.

Solution. The objective function (and therefore also the problem) is not convex, but it is quasiconvex. We have $\max_{i=1,\ldots,k} |f(t_i) - y_i| \le \gamma$ if and only if

$$\left| \frac{a_0 + a_1 t_i + a_2 t_i^2}{1 + b_1 t_i + b_2 t_i^2} - y_i \right| \le \gamma, \quad i = 1, \ldots, k.$$

This is equivalent to (since the denominator is positive)

$$|a_0 + a_1 t + a_2 t_i^2 - y_i(1 + b_1 t_i + b_2 t_i^2)| \le \gamma(1 + b_1 t_i + b_2 t_i^2), \quad i = 1, \ldots, k,$$

which is a set of $2k$ linear inequalities in the variables a and b (for fixed γ). In particular, this shows the objective is quasiconvex. (In fact, it is a generalized linear fractional function.)

To solve the problem we can use a bisection method, solving an LP feasibility problem at each step. At each step we select some value of γ and solve the feasibility problem

$$\begin{array}{ll} \text{find} & a,\ b \\ \text{subject to} & |a_0 + a_1 t_i + a_2 t_i^2 - y_i(1 + b_1 t_i + b_2 t_i^2)| \le \gamma(1 + b_1 t_i + b_2 t_i^2), \quad i = 1, \ldots, k, \end{array}$$

with variables a and b. (Note that as long as $\gamma > 0$, the condition that the denominator is positive is enforced automatically.) This can be turned into the LP feasibility problem

$$\begin{array}{ll} \text{find} & a,\ b \\ \text{subject to} & a_0 + a_1 t_i + a_2 t_i^2 - y_i(1 + b_1 t_i + b_2 t_i^2) \le \gamma(1 + b_1 t_i + b_2 t_i^2), \quad i = 1, \ldots, k \\ & a_0 + a_1 t_i + a_2 t_i^2 - y_i(1 + b_1 t_i + b_2 t_i^2) \ge -\gamma(1 + b_1 t_i + b_2 t_i^2), \quad i = 1, \ldots, k. \end{array}$$

The following Matlab code solves the problem for the particular problem instance.

```
k=201;
t=(-3:6/(k-1):3)';
y=exp(t);

Tpowers=[ones(k,1) t t.^2];

u=exp(3); l=0; % initial upper and lower bounds
bisection_tol=1e-3; % bisection tolerance

while u-l>= bisection_tol
    gamma=(l+u)/2;
    cvx_begin % solve the feasibility problem
    cvx_quiet(true);
    variable a(3);
    variable b(2);
    subject to
        abs(Tpowers*a-y.*(Tpowers*[1;b])) <= gamma*Tpowers*[1;b];
    cvx_end

    if strcmp(cvx_status,'Solved')
        u=gamma;
        a_opt=a;
        b_opt=b;
        objval_opt=gamma;
    else
        l=gamma;
    end
end

y_fit=Tpowers*a_opt./(Tpowers*[1;b_opt]);

figure(1);
plot(t,y,'b', t,y_fit,'r+');
xlabel('t');
ylabel('y');

figure(2);
plot(t, y_fit-y);
xlabel('t');
ylabel('err');
```

The optimal values are

$$a_0 = 1.0099, \quad a_1 = 0.6117, \quad a_2 = 0.1134, \quad b_1 = -0.4147, \quad b_2 = 0.0485,$$

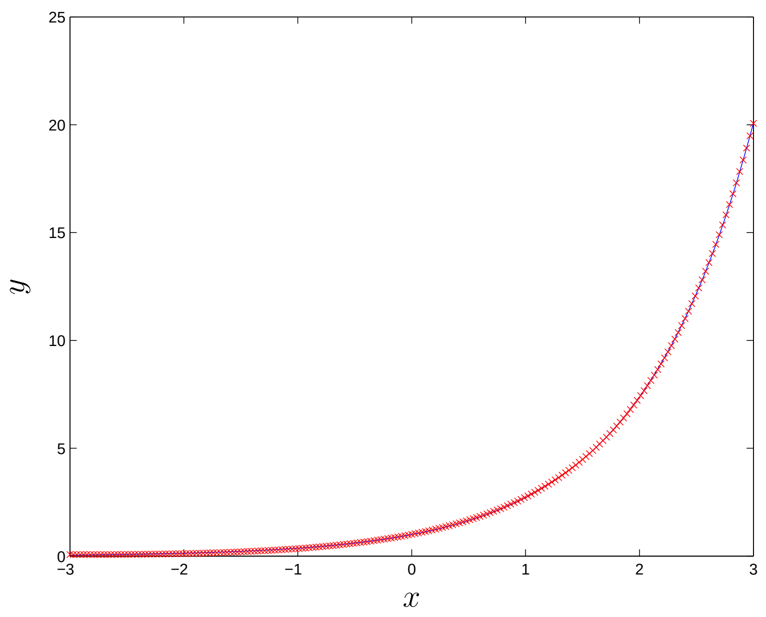


Figure 1 Chebyshev fit with rational function. The line represents the data and the crosses the fitted points.

and the optimal objective value is 0.0233. We also get the following plots.

2. *Maximum likelihood prediction of team ability.* A set of n teams compete in a tournament. We model each team's ability by a number $a_j \in [0,1]$, $j = 1,\dots,n$. When teams j and k play each other, the probability that team j wins is equal to $\mathbf{prob}(a_j - a_k + v > 0)$, where $v \sim \mathcal{N}(0,\sigma^2)$.

   You are given the outcome of m past games. These are organized as

   $$(j^{(i)}, k^{(i)}, y^{(i)}), \quad i = 1,\dots,m,$$

   meaning that game i was played between teams $j^{(i)}$ and $k^{(i)}$; $y^{(i)} = 1$ means that team $j^{(i)}$ won, while $y^{(i)} = -1$ means that team $k^{(i)}$ won. (We assume there are no ties.)

   (a) Formulate the problem of finding the maximum likelihood estimate of team abilities, $\hat{a} \in \mathbf{R}^n$, given the outcomes, as a convex optimization problem. You will find the *game incidence matrix* $A \in \mathbf{R}^{m\times n}$, defined as

   $$A_{il} = \begin{cases} y^{(i)} & l = j^{(i)} \\ -y^{(i)} & l = k^{(i)} \\ 0 & \text{otherwise}, \end{cases}$$

   useful.

   The prior constraints $\hat{a}_i \in [0,1]$ should be included in the problem formulation. Also, we note that if a constant is added to all team abilities, there is no change in

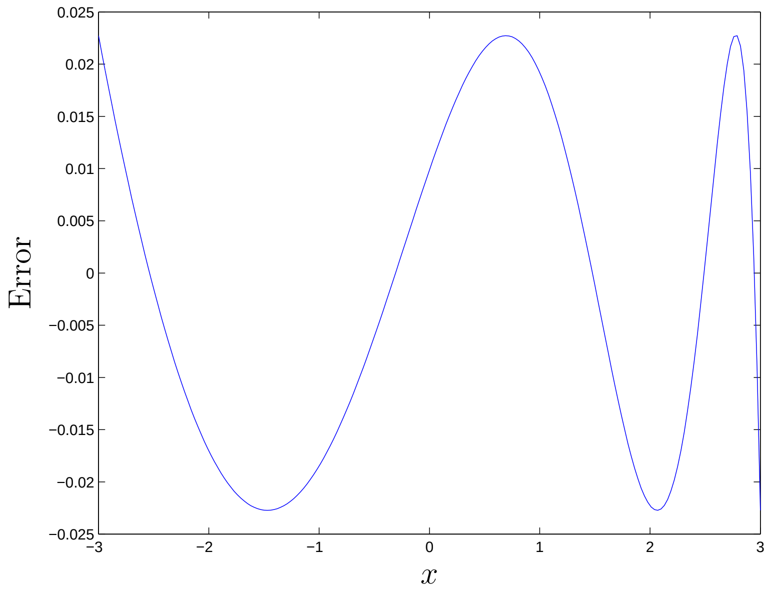


Figure 2 Fitting error for Chebyshev fit of exponential with rational function.

the probabilities of game outcomes. This means that $\hat{a}$ is determined only up to a constant, like a potential. But this doesn't affect the ML estimation problem, or any subsequent predictions made using the estimated parameters.

(b) Find $\hat{a}$ for the team data given in `team_data.m`, in the matrix `train`. (This matrix gives the outcomes for a tournament in which each team plays each other team once.)

CVX does not support the concave function $\log \Phi$, where Φ is the cumulative distribution of a unit Gaussian, but we have provided a good enough approximation, `log_normcdf`, on the course web site. This function is overloaded to handle vector inputs (elementwise).

You can form A using the commands

```
A = sparse(1:m,train(:,1),train(:,3),m,n) + ...
    sparse(1:m,train(:,2),-train(:,3),m,n);
```

(c) Use the maximum likelihood estimate $\hat{a}$ found in part (b) to predict the outcomes of next year's tournament games, given in the matrix `test`, using $\hat{y}^{(i)} = \mathbf{sign}(\hat{a}_{j^{(i)}} - \hat{a}_{k^{(i)}})$. Compare these predictions with the actual outcomes, given in the third column of `test`. Given the fraction of correctly predicted outcomes.

The games played in `train` and `test` are the same, so another, simpler method for predicting the outcomes in `test` it to just assume the team that won last year's match will also win this year's match. Give the percentage of correctly predicted outcomes using this simple method.

Solution.

(a) The likelihood of the outcomes y given a is

$$p(y|a) = \prod_{i=1,\ldots,n} \Phi\left(\frac{1}{\sigma} y^{(i)}(a_{j^{(i)}} - a_{k^{(i)}})\right),$$

where Φ is the cumulative distribution of the standard normal. The log-likelihood function is therefore

$$l(a) = \log p(y|a) = \sum_i \log \Phi\left((1/\sigma)(Aa)_i\right).$$

This is a concave function.

The maximum likelihood estimate $\hat{a}$ is any solution of

$$\begin{array}{ll} \text{maximize} & l(a) \\ \text{subject to} & 0 \preceq a \preceq 1. \end{array}$$

This is a convex optimization problem since the objective, which is maximized, is concave, and the constraints are $2n$ linear inequalities.

(b) The following code solves the problem

```
% Form adjacency matrix
A1 = sparse(1:m,train(:,1),train(:,3),m,n);
A2 = sparse(1:m,train(:,2),-train(:,3),m,n);
A = A1+A2;

% Estimate abilities
cvx_begin
    variable a_hat(n)
    minimize(-sum(log_normcdf(A*a_hat/sigma)))
    subject to
        a_hat >= 0
        a_hat <= 1
cvx_end
```

Using this code we get that $\hat{a} = (1.0, 0.0, 0.68, 0.37, 0.79, 0.58, 0.38, 0.09, 0.67, 0.58)$.

(c) The following code is used to predict the outcomes in the test set

```
% Estimate errors in test set
res = sign(a_hat(test(:,1))-a_hat(test(:,2)));
Pml = 1-length(find(res-test(:,3)))/m_test
Ply = 1-length(find(train(:,3)-test(:,3)))/m_test
```

The maximum likelihood estimate gives a correct prediction of 86.7% of the games in `test`. On the other hand, 75.6% of the games in `test` have the same outcome as the games in `train`.

3. *Piecewise-linear fitting.* In many applications some function in the model is not given by a formula, but instead as tabulated data. The tabulated data could come from empirical measurements, historical data, numerically evaluating some complex expression or solving some problem, for a set of values of the argument. For use in a convex optimization model, we then have to fit these data with a convex function that is compatible with the solver or other system that we use. In this problem we explore a very simple problem of this general type.

   Suppose we are given the data (x_i, y_i), $i = 1, \ldots, m$, with $x_i,\ y_i \in \mathbf{R}$. We will assume that x_i are sorted, *i.e.*, $x_1 < x_2 < \cdots < x_m$. Let $a_0 < a_1 < a_2 < \cdots < a_K$ be a set of fixed knot points, with $a_0 \leq x_1$ and $a_K \geq x_m$. Explain how to find the convex piecewise linear function f, defined over $[a_0, a_K]$, with knot points a_i, that minimizes the least-squares fitting criterion

$$\sum_{i=1}^{m} (f(x_i) - y_i)^2.$$

   You must explain what the variables are and how they parametrize f, and how you ensure convexity of f.

   Hints. One method to solve this problem is based on the Lagrange basis, $f_0, \ldots, f_K$, which are the piecewise linear functions that satisfy

$$f_j(a_i) = \delta_{ij}, \quad i, j = 0, \ldots, K.$$

   Another method is based on defining $f(x) = \alpha_i x + \beta_i$, for $x \in (a_{i-1}, a_i]$. You then have to add conditions on the parameters α_i and β_i to ensure that f is continuous and convex.

   Apply your method to the data in the file `pwl_fit_data.m`, which contains data with $x_j \in [0, 1]$. Find the best affine fit (which corresponds to $a = (0, 1)$), and the best piecewise-linear convex function fit for 1, 2, and 3 internal knot points, evenly spaced in $[0, 1]$. (For example, for 3 internal knot points we have $a_0 = 0,\ a_1 = 0.25,\ a_2 = 0.50,\ a_3 = 0.75,\ a_4 = 1$.) Give the least-squares fitting cost for each one. Plot the data and the piecewise-linear fits found. Express each function in the form

$$f(x) = \max_{i=1\ldots,K} (\alpha_i x + \beta_i).$$

   (In this form the function is easily incorporated into an optimization problem.)

   Solution. Following the hint, we will use the Lagrange basis functions $f_0, \ldots, f_K$. These can be expressed as

$$f_0(x) = \left(\frac{a_1 - x}{a_1 - a_0} \right)_+,$$

$$f_i(x) = \left(\min \left(\frac{x - a_{i-1}}{a_i - a_{i-1}}, \frac{a_{i+1} - x}{a_i - a_{i+1}} \right) \right)_+, \quad i = 1, \ldots, K - 1,$$

and

$$f_K(x) = \left(\frac{x - a_{K-1}}{a_K - a_{K-1}} \right)_+ .$$

The function f can be parametrized as

$$f(x) = \sum_{i=0}^{K} z_i f_i(x),$$

where $z_i = f(a_i)$, $i = 0, \ldots, K$. We will use $z = (z_0, \ldots, z_K)$ to parametrize f. The least-squares fitting criterion is then

$$J = \sum_{i=1}^{m} (f(x_i) - y_i)^2 = \|Fz - y\|_2^2,$$

where $F \in \mathbf{R}^{m \times (K+1)}$ is the matrix

$$F_{ij} = f_j(x_i), \quad i = 1, \ldots, m, \quad j = 0, \ldots, K.$$

(We index the columns of F from 0 to K here.)

We must add the constraint that f is convex. This is the same as the condition that the slopes of the segments are nondecreasing, *i.e.*,

$$\frac{z_{i+1} - z_i}{a_{i+1} - a_i} \geq \frac{z_i - z_{i-1}}{a_i - a_{i-1}}, \quad i = 1, \ldots, K-1.$$

This is a set of linear inequalities in z. Thus, the best PWL convex fit can be found by solving the QP

$$\begin{array}{ll} \text{minimize} & \|Fz - y\|_2^2 \\ \text{subject to} & \frac{z_{i+1} - z_i}{a_{i+1} - a_i} \geq \frac{z_i - z_{i-1}}{a_i - a_{i-1}}, \quad i = 1, \ldots, K-1. \end{array}$$

The following code solves this problem for the data in `pwl_fit_data`.

```
cvx_quiet('true')
figure
plot(x,y,'k:','linewidth',2)
hold on

% Single line
p = [x ones(100,1)]\y;
alpha = p(1)
beta = p(2)
plot(x,alpha*x+beta,'b','linewidth',2)
mse = norm(alpha*x+beta-y)^2
```

```
for K = 2:4
    % Generate Lagrange basis
    a = (0:(1/K):1)';
    F = max((a(2)-x)/(a(2)-a(1)),0);
    for k = 2:K
        a_1 = a(k-1);
        a_2 = a(k);
        a_3 = a(k+1);
        f = max(0,min((x-a_1)/(a_2-a_1),(a_3-x)/(a_3-a_2)));
        F = [F f];
    end
    f = max(0,(x-a(K))/(a(K+1)-a(K)));
    F = [F f];

    % Solve problem
    cvx_begin
        variable z(K+1)
        minimize(norm(F*z-y))
        subject to
            (z(3:end)-z(2:end-1))./(a(3:end)-a(2:end-1)) >=...
                (z(2:end-1)-z(1:end-2))./(a(2:end-1)-a(1:end-2))
    cvx_end

    % Calculate alpha and beta
    alpha = (z(2:end)-z(1:end-1))./(a(2:end)-a(1:end-1))
    beta = z(2:end)-alpha(1:end).*a(2:end)

    % Plot solution
    y2 = F*z;
    mse = norm(y2-y)^2
    if K==2
        plot(x,y2,'r','linewidth',2)
    elseif K==3
        plot(x,y2,'g','linewidth',2)
    else
        plot(x,y2,'m','linewidth',2)
    end

end
xlabel('x')
ylabel('y')
```

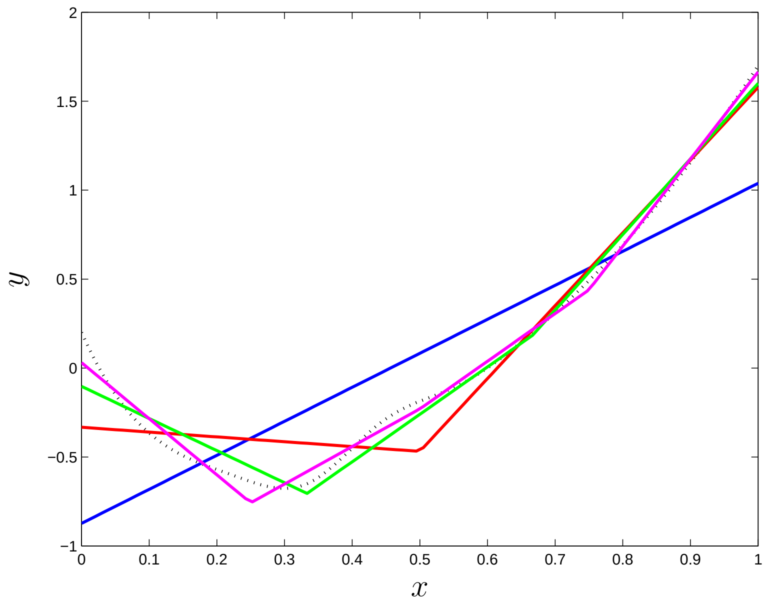


Figure 3 Piecewise-linear approximations for $K = 1, 2, 3, 4$

This generates figure 3. We can see that the approximation improves as K increases. The following table shows the result of this approximation.

K	$\alpha_1, \dots, \alpha_K$	$\beta_1, \dots, \beta_K$	J
1	1.91	-0.87	12.73
2	$-0.27, 4.09$	$-0.33, -2.51$	2.62
3	$-1.80, 2.67, 4.25$	$-0.10, -1.59, -2.65$	0.60
4	$-3.15, 2.11, 2.68, 4.90$	$0.03, -1.29, -1.57, -3.23$	0.22

There is another way to solve this problem. We are looking for a piecewise linear function. If we have at least one internal knot ($K \geq 2$), the function should satisfy the two following constraints:

- convexity: $\alpha_1 \leq \alpha_2 \leq \cdots \leq \alpha_K$
- continuity: $\alpha_i a_i + \beta_i = \alpha_{i+1} a_i + \beta_{i+1},\ i = 1, \dots, K-1.$

Therefore, the opimization problem is

$$\begin{array}{ll} \text{minimize} & (\sum_{i=1}^m f(x_i) - y_i)^2 \\ \text{subject to} & \alpha_i \leq \alpha_{i+1}, \quad i = 1, \dots, K-1 \\ & \alpha_i a_i + \beta_i = \alpha_{i+1} a_i + \beta_{i+1}, \quad i = 1, \dots, K-1 \end{array}$$

Reformulating the problem by representing $f(x_i)$ in matrix form, we get

$$\begin{array}{ll} \text{minimize} & \| \mathbf{diag}(x) F\alpha + F\beta - y\|^2 \\ \text{subject to} & \alpha_i \leq \alpha_{i+1}, \quad i = 1, \dots, K-1 \\ & \alpha_i a_i + \beta_i = \alpha_{i+1} a_i + \beta_{i+1}, \quad i = 1, \dots, K-1 \end{array}$$

where the variables are $\alpha \in \mathbf{R}^K$ and $\beta \in \mathbf{R}^K$, and problem data are $x \in \mathbf{R}^m$, $y \in \mathbf{R}^m$ and

$$F_{ij} = \begin{cases} 1 & \text{if } a_{j-1} = x_i,\ j = 1 \\ 1 & \text{if } a_{j-1} < x_i \leq a_j \\ 0 & \text{otherwise}~. \end{cases}$$

```
% another approach for PWL fitting problem
clear all;
pwl_fit_data;
m  = length(x);
xp = 0:0.001:1;  % for fine-grained pwl function plot
mp = length(xp);
yp = [];

for K = 1:4 % internal knot 1,2,3

    a = [0:1/K:1]'; % a_0,...,a_K
    % matrix for sum f(x_i)
    F = sparse(1:m,max(1,ceil(x*K)),1,m,K);

    % solve problem
    cvx_begin
    variables alpha(K) beta(K)
    minimize( norm(diag(x)*F*alpha+F*beta-y) )
    subject to
    if (K>=2)
        alpha(1:K-1).*a(2:K)+beta(1:K-1) == alpha(2:K).*a(2:K)+beta(2:K)
        a(1:K-1) <= a(2:K)
    end
    cvx_end

    fp = sparse(1:mp,max(1,ceil(xp*K)),1,mp,K);
    yp = [yp diag(xp)*fp*alpha+fp*beta];
end
plot(x,y,'b.',xp,yp);
```

4. *Robust least-squares with interval coefficient matrix.* An *interval matrix* in $\mathbf{R}^{m\times n}$ is a matrix whose entries are intervals:

$$\mathcal{A} = \{A \in \mathbf{R}^{m\times n} \mid |A_{ij} - \bar{A}_{ij}| \leq R_{ij},\ i = 1,\dots,m,\ j = 1,\dots,n\}.$$

The matrix $\bar{A} \in \mathbf{R}^{m\times n}$ is called the *nominal value* or *center value*, and $R \in \mathbf{R}^{m\times n}$, which is elementwise nonnegative, is called the *radius*.

The robust least-squares problem, with interval matrix, is

$$\text{minimize} \quad \sup_{A\in\mathcal{A}} \|Ax-b\|_2,$$

with optimization variable $x \in \mathbf{R}^n$. The problem data are $\mathcal{A}$ (*i.e.*, $\bar{A}$ and R) and $b \in \mathbf{R}^m$. The objective, as a function of x, is called the *worst-case residual norm.* The robust least-squares problem is evidently a convex optimization problem.

(a) Formulate the interval matrix robust least-squares problem as a standard optimization problem, *e.g.*, a QP, SOCP, or SDP. You can introduce new variables if needed. Your reformulation should have a number of variables and constraints that grows linearly with m and n, and not exponentially.

(b) Consider the specific problem instance with $m = 4$, $n = 3$,

$$\mathcal{A} = \begin{bmatrix} 60 \pm 0.05 & 45 \pm 0.05 & -8 \pm 0.05 \\ 90 \pm 0.05 & 30 \pm 0.05 & -30 \pm 0.05 \\ 0 \pm 0.05 & -8 \pm 0.05 & -4 \pm 0.05 \\ 30 \pm 0.05 & 10 \pm 0.05 & -10 \pm 0.05 \end{bmatrix}, \qquad b = \begin{bmatrix} -6 \\ -3 \\ 18 \\ -9 \end{bmatrix}.$$

(The first part of each entry in $\mathcal{A}$ gives $\bar{A}_{ij}$; the second gives R_{ij}, which are all 0.05 here.) Find the solution x_{ls} of the nominal problem (*i.e.*, minimize $\|\bar{A}x-b\|_2$), and robust least-squares solution x_{rls}. For each of these, find the nominal residual norm, and also the worst-case residual norm. Make sure the results make sense.

Solution:

(a) The problem is equivalent to

$$\text{minimize} \quad \sup_{A\in\mathcal{A}} \|Ax-b\|_2^2,$$

which can be reformulated as

$$\begin{array}{ll} \text{minimize} & y^T y \\ \text{subject to} & -y \preceq Ax - b \preceq y, \quad \text{for all } A \in \mathcal{A}. \end{array}$$

We have

$$\begin{aligned} \sup_{A\in\mathcal{A}} (Ax-b)_i &= \sum_{j=1}^n (\bar{A}_{ij}x_j + R_{ij}|x_j|) - b_i \\ \inf_{A\in\mathcal{A}} (Ax-b)_i &= \sum_{j=1}^n (\bar{A}_{ij}x_j - R_{ij}|x_j|) - b_i. \end{aligned}$$

We can therefore write the problem as

$$\begin{array}{ll} \text{minimize} & y^T y \\ \text{subject to} & \bar{A}x + R|x| - b \preceq y \\ & \bar{A}x - R|x| - b \succeq -y \end{array}$$

where $|x| \in \mathbf{R}^n$ is the vector with elements $|x|_i = |x_i|$, or equivalently as the QP

$$\begin{array}{ll} \text{minimize} & y^T y \\ \text{subject to} & \bar{A}x + Rz - b \preceq y \\ & \bar{A}x - Rz - b \succeq -y \\ & -z \preceq x \preceq z. \end{array}$$

The variables are $x \in \mathbf{R}^n$, $y \in \mathbf{R}^m$, $z \in \mathbf{R}^n$.

The problem also has an alternative formulation: the trick is to find an alternative expression for the worst-case residual norm as a function of x. Let $f(x) = \sup_{A \in \mathcal{A}} \|Ax - b\|_2^2$.

$$\begin{aligned} f(x) &= \sup\{\|Ax - b\|_2^2 \mid A = \bar{A} + \Delta,\ |\Delta_{ij}| \le R_{ij},\ i = 1, \ldots, m,\ j = 1, \ldots, n\} \\ &= \sup\{\|\bar{A}x + \Delta x - b\|_2^2 \mid |\Delta_{ij}| \le R_{ij},\ i = 1, \ldots, m,\ j = 1, \ldots, n\} \\ &= \sup\{\|r + \Delta x\|_2^2 \mid |\Delta_{ij}| \le R_{ij},\ i = 1, \ldots, m,\ j = 1, \ldots, n\} \end{aligned}$$

where $r = \bar{A}x - b$.

Since $\|r+\Delta x\|_2^2 = \sum_{i=1}^m (r_i + \sum_{j=1}^n \Delta_{ij} x_j)^2 = \sum_{i=1}^m |r_i + \sum_{j=1}^n \Delta_{ij} x_j|^2$, it's easy to see that this expression is separable in the rows of Δ. So in order to find an expression for $f(x)$, we just need to find an expression for $\sup\{|r_i + \sum_{j=1}^n \Delta_{ij} x_j| \mid |\Delta_{ij}| \le R_{ij},\ j = 1, \ldots, n\}$. The supremum is achieved by taking $\Delta_{ij} = R_{ij}\,\mathbf{sign}(x_j)$ if $r_i \ge 0$ and $\Delta_{ij} = -R_{ij}\,\mathbf{sign}(x_j)$ if $r_i < 0$. The supremum is equal to $|r_i| + \sum_{j=1}^n R_{ij}|x_j|$. Therefore

$$\begin{aligned} f(x) &= \sum_{i=1}^m (|r_i| + \sum_{j=1}^n R_{ij}|x_j|)^2 \\ &= \||\bar{A}x - b| + R|x|\|_2^2 \end{aligned}$$

The robust least-square problem then be reformulated as

$$\text{minimize} \quad \||\bar{A}x - b| + R|x|\|_2$$

Note that the objective function is convex since the Euclidian norm is convex and increasing on $\mathbf{R}_+^m$ and $|\bar{A}x - b| + R|x|$ is convex and nonnegative.

(b) The following script computes the least-squares and the robust solutions and also computes, for each one, the nominal and the worst-case residual norms.

```
% input data
A_bar = [ 60    45    -8; ...
          90    30   -30; ...
           0    -8    -4; ...
          30    10   -10];
d = .05;
R = d*ones(4,3);
```

```
b = [ -6; -3; 18; -9];

% least-squares solution
x_ls = A_bar\b;

% robust least-squares solution
cvx_begin
    variables x(3) y(4) z(3)
    minimize ( norm( y ) )
    A_bar*x + R*z - b <= y
    A_bar*x - R*z - b >= -y
    x <= z
    x + z >= 0
cvx_end

% computing nominal residual norms
nom_res_ls = norm(A_bar*x_ls - b);
nom_res_rls = norm(A_bar*x - b);

% computing worst-case nominal norms
r = A_bar*x_ls - b;
Delta = zeros(4,3);
for i=1:length(r)
    if r(i) < 0
        Delta(i,:) = -d*sign(x_ls');
    else
        Delta(i,:) = d*sign(x_ls');
    end
end
wc_res_ls = norm(r + Delta*x_ls);
wc_res_rls = cvx_optval;

% display
disp('Residual norms for the nominal problem when using LS solution: ');
disp(nom_res_ls);
disp('Residual norms for the nominal problem when using robust solution: ');
disp(nom_res_rls);
disp('Residual norms for the worst-case problem when using LS solution: ');
disp(wc_res_ls);
disp('Residual norms for the worst-case problem when using robust solution: ');
disp(wc_res_rls);
```

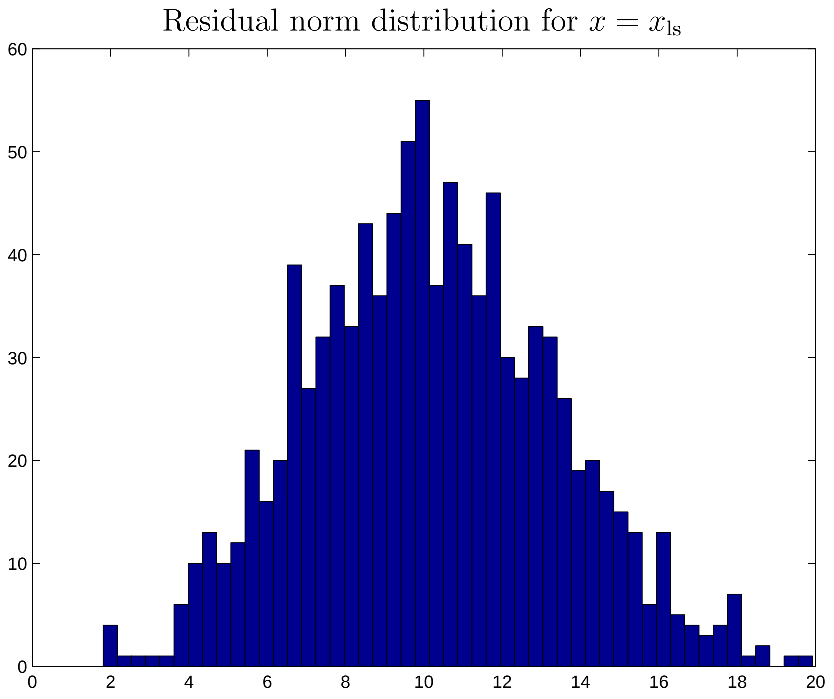
The robust least-square solution can also be found using the following script:

```
cvx_begin
    variables x(3) t(4)
    minimize ( norm ( t ) )
    abs(A_bar*x - b) + R*abs(x) <= t
cvx_end
```

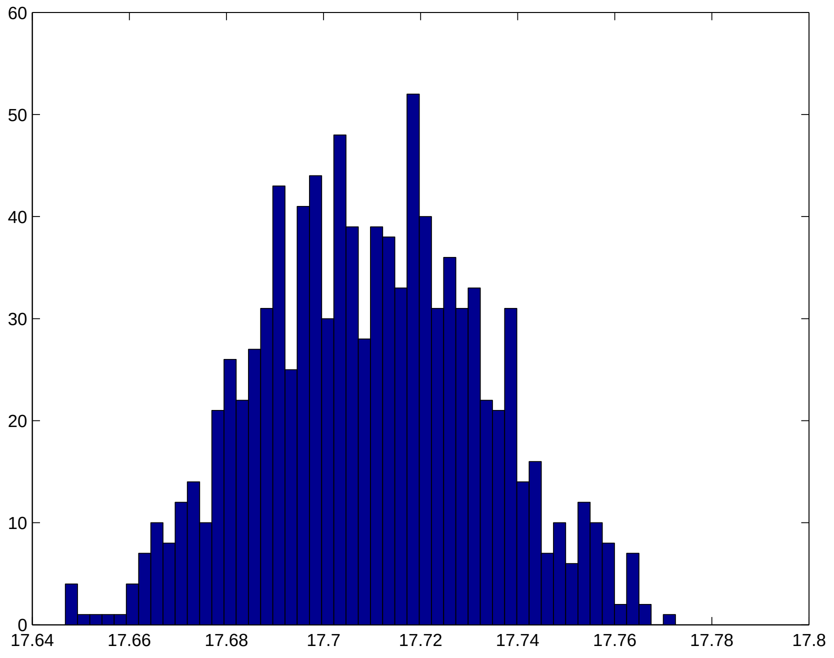
This script returns the following results:

```
Residual norms for the nominal problem when using LS solution:
    7.5895

Residual norms for the nominal problem when using robust solution:
   17.7106

Residual norms for the worst-case problem when using LS solution:
   26.7012

Residual norms for the worst-case problem when using robust solution:
   17.7940
```

We also generated, for fun, the following histograms showing the distribution of the residual norms for the case where $x = x_{\rm ks}$ and $x = x_{\rm rls}$. Those were obtained by creating 1000 instances of A by sampling A_{ij} uniformly between $\bar{A}_{ij} - R_{ij}$ and $\bar{A}_{ij} + R_{ij}$, and then evaluating the residual norm for each A and each of the 2 solutions.



Residual norm distribution for $x = x_{\rm rls}$



The following script generates these histograms:

```
% Monte-Carlo simulation
N = 1000;
res_ls = zeros(N,1);
res_rls = zeros(N,1);
for k=1:N
    Delta = d*(2*rand(4,3)-1);
    A = A_bar + Delta;
    res_ls(k) = norm(A*x_ls - b);
    res_rls(k) = norm(A*x - b);
end
figure;
hist(res_ls,50);
figure;
hist(res_rls,50);
```

5. *Total variation image interpolation.* A grayscale image is represented as an $m \times n$ matrix of intensities $U^{\rm orig}$. You are given the values $U^{\rm orig}_{ij}$, for $(i,j) \in \mathcal{K}$, where $\mathcal{K} \subset \{1,\ldots,m\} \times \{1,\ldots,n\}$. Your job is to *interpolate* the image, by guessing the missing values. The reconstructed image will be represented by $U \in \mathbf{R}^{m\times n}$, where U satisfies the interpolation conditions $U_{ij} = U^{\rm orig}_{ij}$ for $(i,j) \in \mathcal{K}$.

   The reconstruction is found by minimizing a roughness measure subject to the interpolation conditions. One common roughness measure is the ℓ_2 variation (squared),

$$\sum_{i=2}^{m}\sum_{j=2}^{n}\left((U_{ij}-U_{i-1,j})^2+(U_{ij}-U_{i,j-1})^2\right).$$

Another method minimizes instead the *total variation*,

$$\sum_{i=2}^{m}\sum_{j=2}^{n}\left(|U_{ij}-U_{i-1,j}|+|U_{ij}-U_{i,j-1}|\right).$$

Evidently both methods lead to convex optimization problems.

Carry out ℓ_2 and total variation interpolation on the problem instance with data given in `tv_img_interp.m`. This will define `m`, `n`, and matrices `Uorig` and `Known`. The matrix `Known` is $m \times n$, with (i,j) entry one if $(i,j) \in \mathcal{K}$, and zero otherwise. The mfile also has skeleton plotting code. (We give you the entire original image so you can compare your reconstruction to the original; obviously your solution cannot access U_{ij}^{orig} for $(i,j) \notin \mathcal{K}$.)

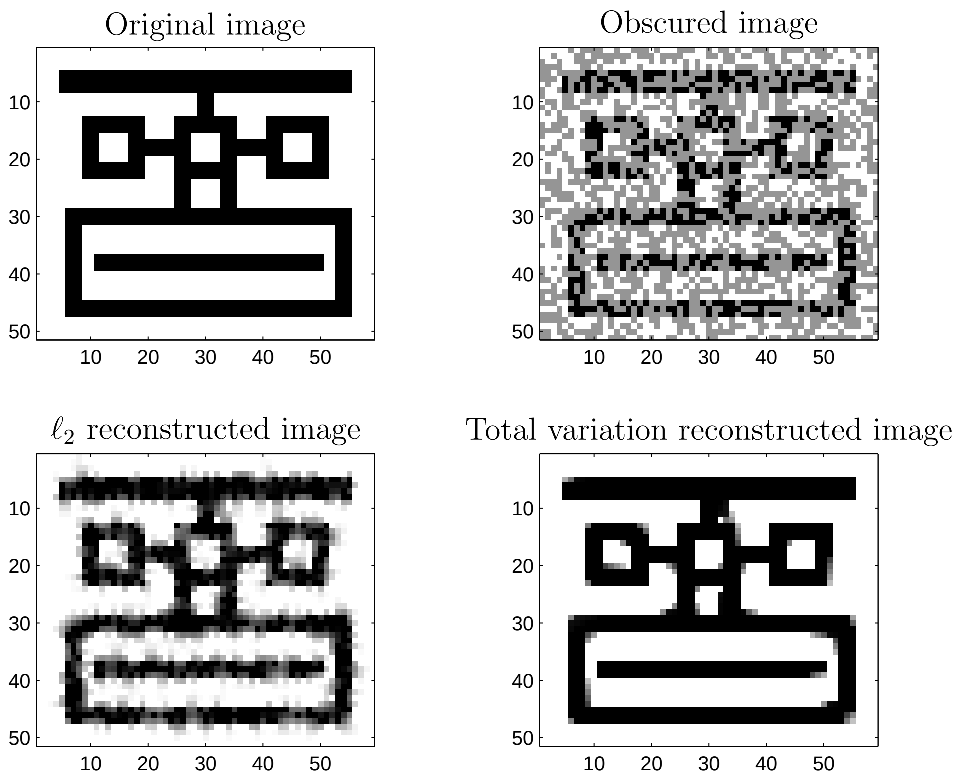
Solution. The code for the interpolation is very simple. For ℓ_2 interpolation, the code is the following.

```
cvx_begin
    variable Ul2(m, n);
    Ul2(Known) == Uorig(Known); % Fix known pixel values.
    Ux = Ul2(2:end,2:end) - Ul2(2:end,1:end-1); % x (horiz) differences
    Uy = Ul2(2:end,2:end) - Ul2(1:end-1,2:end); % y (vert) differences
    minimize(norm([Ux(:); Uy(:)], 2)); % l2 roughness measure
cvx_end
```

For total variation interpolation, we use the following code.

```
cvx_begin
    variable Utv(m, n);
    Utv(Known) == Uorig(Known); % Fix known pixel values.
    Ux = Utv(2:end,2:end) - Utv(2:end,1:end-1); % x (horiz) differences
    Uy = Utv(2:end,2:end) - Utv(1:end-1,2:end); % y (vert) differences
    minimize(norm([Ux(:); Uy(:)], 1)); % tv roughness measure
cvx_end
```

We get the following images



6. *Relaxed and discrete A-optimal experiment design.* This problem concerns the A-optimal experiment design problem, described on page 387, with data generated as follows.

```
n = 5; % dimension of parameters to be estimated
p = 20; % number of available types of measurements
m = 30; % total number of measurements to be carried out
randn('state', 0);
V=randn(n,p); % columns are vi, the possible measurement vectors
```

Solve the relaxed A-optimal experiment design problem,

$$\begin{array}{ll} \text{minimize} & (1/m)\,\mathbf{tr}\left(\sum_{i=1}^p \lambda_i v_i v_i^T\right)^{-1} \\ \text{subject to} & \mathbf{1}^T\lambda = 1, \quad \lambda \succeq 0, \end{array}$$

with variable $\lambda \in \mathbf{R}^p$. Find the optimal point $\lambda^\star$ and the associated optimal value of the relaxed problem. This optimal value is a lower bound on the optimal value of the discrete A-optimal experiment design problem,

$$\begin{array}{ll} \text{minimize} & \mathbf{tr}\left(\sum_{i=1}^p m_i v_i v_i^T\right)^{-1} \\ \text{subject to} & m_1 + \cdots + m_p = m, \quad m_i \in \{0,\ldots,m\}, \quad i = 1,\ldots,p, \end{array}$$

with variables $m_1, \ldots, m_p$. To get a suboptimal point for this discrete problem, round the entries in $m\lambda^\star$ to obtain integers $\hat{m}_i$. If needed, adjust these by hand or some other method to ensure that they sum to m, and compute the objective value obtained. This is, of course, an upper bound on the optimal value of the discrete problem. Give the gap between this upper bound and the lower bound obtained from the relaxed problem. Note that the two objective values can be interpreted as mean-square estimation error $\mathbf{E}\,\|\hat{x} - x\|_2^2$.

Solution. The objective of the relaxed problem is convex, so it is a convex problem. Expressing it in `cvx` requires a little work. We'd like to write the objective as

```
minimize ((1/m)*trace(inv(V*diag(lambda)*V')))
```

but this won't work, because `cvx` doesn't know about matrix convex functions. Instead, we can express the objective as a sum of matrix fractional functions,

$$\begin{array}{ll} \text{minimize} & (1/m)\sum_{k=1}^n e_k^T \left(\sum_{i=1}^p \lambda_i v_i v_i^T\right)^{-1} e_k \\ \text{subject to} & \mathbf{1}^T\lambda = 1, \quad \lambda \succeq 0. \end{array}$$

where $e_k \in \mathbf{R}^n$ is the kth unit vector. (Note that e is defined in exercise 6.9 as the estimation error vector, so e_k could also mean the kth entry of the error vector. But here, clearly, e_k is kth unit vector.)

We can express this in `cvx` using the function `matrix_frac`. The following code solves the problem.

```
n = 5; % dimension
p = 20; % number of available types of measurements
m = 30; % total number of measurements to be carried out
randn('state', 0);
V=randn(n,p); % columns are vi, the possible measurement vectors

cvx_begin
    variable lambda(p)
    obj = 0;
    for k=1:n
        ek = zeros(n,1);
        ek(k)=1;
        obj = obj + (1/m)*matrix_frac(ek,V*diag(lambda)*V');
    end
    minimize( obj )
    subject to
        sum(lambda) == 1
        lambda >= 0
```

```
cvx_end

lower_bound = cvx_optval

t = -0.00; % small offset chosen by hand to make rounding work out.
% for this problem data, none is needed!
m_rnd = pos(round(m*lambda+t));
sum(m_rnd) % should be == m

% now find objective value of rounded experiment design
upper_bound = trace(inv(V*diag(m_rnd/m)*V'))/m
gap = upper_bound-lower_bound
rel_gap = gap/lower_bound
```

For this problem instance, simple rounding yielded $\hat{m}_i$ that summed to $m = 30$, so no adjustment of the rounded values is needed. The lower bound is 0.2481; the upper bound is 0.2483. The gap is 0.00023, which is around 0.1%.

What this means is this: We have found a choice of 30 measurements, each one from the set of 20 possible measurements, that yields a mean-square estimation error $\mathbf{E}\,\|\hat{x}-x\|_2^2 = 0.2483$. We do not know whether this is the optimal choice of 30 measurements. But we do know that this choice is no more than 0.1% suboptimal; the optimal choice can achieve a mean-square error that is no smaller than 0.2481. Our experiment design is, if not optimal, very nearly optimal. (In fact, it is very likely to be optimal.)