Solutions to hw5 homework on Convex Optimization https://web.stanford.edu/class/ee364a/homework.html

Andrei Keino

October 9, 2020

5.17

Robust linear programming with polyhedral uncertainty. Consider the robust LP:

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & \sup_{a \in P_i} a^T x \le b_i, \ i = 1, ..., m \end{array}$$

with variable $x \in R^n$, where $P_i = \{a : \ C_i a \preceq d_i\}$. The problem data are $a \in R^n$, $C_i \in R^{m_i \times n}$, $d_i \in R^{m_i}$, and $b \in R^m$. We asume the polyhedra P_i are nonempty. Show that this problem is equivalent to the LP:

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & d_i^T z_i \le b_i, \ i = 1, ..., m \\ & C_i z_i = x, \ i = 1, ..., m \\ & z_i \succeq 0, \ i = 1, ..., m \end{array}$$

with variables $x \in R^n$, $z_i \in R^{m_i}, \ i = 1, ..., m$. Hint: find the dual of the problem of maximizing $a_i^T x$ over $a_i \in P_i$ (with variable a_i).

Solution:

The problem of maximizing $a_i^T x$ over $a_i \in P_i$ (with variable a_i) is:

$$
\begin{array}{ll}
\text{maximize} & a_i^T x \\
\text{subject to} & a_i \in P_i, \text{where } P_i = \{a : \ C_i a \preceq d_i\}
\end{array}
$$

or

$$
\begin{array}{ll}
\text{minimize} & -\,a_i^T x \\
\text{subject to} & C_i a_i \preceq d_i
\end{array}
$$

The Lagrange dual of this problem is:

$$
\begin{array}{ll}
\text{minimize} & \displaystyle\sum_{i=1}^{m} \lambda_i d_i \\
\text{subject to} & C_i \lambda_i = x \\
 & \lambda_i \succeq 0
\end{array}
$$

The optimal value of this problem is less or equal to b_i, so we have the equivalent problem to our LP:

$$
\begin{array}{ll}
\text{minimize} & c^T x \\
\text{subject to} & d_i^T \lambda_i \leq b_i, \ i = 1, ..., m \\
 & C_i \lambda_i = x, \ i = 1, ..., m \\
 & \lambda_i \succeq 0, \ i = 1, ..., m
\end{array}
$$

5.40

E - optimal experiment design. A variation on two optimal experiment design problems of exercise 5.10 is the E - optimal design problem:

$$
\begin{array}{ll}
\text{minimize} & \lambda_{max} (\displaystyle\sum_{i=1}^{p} x_i v_i v_i^T)^{-1} \\
\text{subject to} & x \succeq 0, \ \mathbf{1}^T x = 1
\end{array}
$$

(See also §7.5.) Derive a dual for this problem first by reformulating it as:

$$\begin{array}{ll} \text{minimize} & 1/t \\ \text{subject to} & \sum_{i=1}^{p} x_i v_i v_i^T \succeq t\boldsymbol{I} \\ & x \succeq 0, \ \mathbf{1}^T x = 1 \end{array}$$

with variables $t \in R$, $x \in R^p$ and domain $R_{++} \times R^p$, and applying Lagrange duality. Simplify the dual problem as much as you can.

Solution: Let us introduce a variable $t \in R_{++}$. Then for a matrix A, inequality $\lambda_{max}(A^{-1}) \leq 1/t$ means $A^{-1} \preceq \frac{1}{t}I$, or $A \preceq tI$. Setting A to $\sum_{i=1}^{p} x_i v_i v_i^T$ we get a problem:

$$\begin{array}{ll} \text{minimize} & 1/t \\ \text{subject to} & \sum_{i=1}^{p} x_i v_i v_i^T \succeq t\boldsymbol{I} \\ & x \succeq 0, \ \mathbf{1}^T x = 1 \end{array}$$

The Lagrangian is:

$$L(x,t,Z,z) = 1/t - tr(Z(\sum_{i=1}^{p} x_i v_i v_i^T - t\boldsymbol{I})) - z^T x + \nu(\mathbf{1}^T x - 1) = 1/t + t\, tr(Z) + \sum_{i=1}^{p} x_i(-v_i^T Z v_i - z_i + \nu) - \nu$$

the infimum of x is bounded below only if $-v_i^T Z v_i - z_i + \nu = 0$. The

$$inf_{\nu} 1/t + t\, tr(Z) = \begin{cases} 2\sqrt{tr(Z)}, & Z \succeq 0 \\ -\infty, & \text{otherwise} \end{cases}$$

the dual function is

$$L(Z,z,\nu) = \begin{cases} 2\sqrt{tr(Z)} - \nu, & Z \succeq 0, \, -v_i^T Z v_i - z_i + \nu = 0, \\ -\infty, & \text{otherwise} \end{cases}$$

The dual problem:

$$\begin{array}{ll} \text{maximize} & 2\sqrt{tr(Z)} - \nu \\ \text{subject to} & v_i^T Z v_i + z_i \leq 0, \ i = 1, ..., p \\ & Z \succeq 0, \, \nu \geq 0 \end{array}$$

We can define $W = (1/\nu)Z$

$$\begin{array}{ll} \text{maximize} & 2\sqrt{\nu}\sqrt{tr(W)} - \nu \\ \text{subject to} & v_i^T W v_i \le 1,\ i = 1, ..., p \\ & W \succeq 0,\ \nu \ge 0 \end{array}$$

Maximizing over ν we get $\nu = tr(W)$, so the problem is:

$$\begin{array}{ll} \text{maximize} & tr(W) \\ \text{subject to} & v_i^T W v_i \le 1,\ i = 1, ..., p \\ & W \succeq 0 \end{array}$$

6.3

Formulate the following approximation problems as LPs, QPs, SOCPs, or SDPs. The problem data are $A \in R^{n \times m}$ and $b \in R^m$. The rows of A are denoted a_i^T.

(a) Deadzone-linear penalty approximation: minimize $\sum_{i=1}^m \phi(a_i^T x - b_i)$, where

$$\phi(u) = \begin{cases} 0, & |u| \le a \\ |u| - a, & |u| > a \end{cases}$$

where $a > 0$.

(b) Log-barrier penalty approximation: minimize $\sum_{i=1}^m \phi(a_i^T x - b_i)$, where

$$\phi(u) = \begin{cases} -a^2 log(1 - (u/a)^2), & |u| < a \\ \infty, & |u| \ge a \end{cases}$$

with $a > 0$.

(c) Huber penalty approximation: minimize $\sum_{i=1}^m \phi(a_i^T x - b_i)$, where

$$\phi(u) = \begin{cases} u^2, & |u| \le M \\ M(2|u| - M), & |u| > M \end{cases}$$

with $M > 0$.

(d) Log-Chebyshev approximation: minimize $max_{i=1,...,m}|log(a_i^T x) - log(b_i)|$. We assume $b \succ 0$. An equivalent convex from is

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & 1/t \le a_i^T x / b_i \le t,\ i = 1, ..., m \end{array}$$

with variables $x \in R^n$ and $t \in R$ and domain $R^n \times R_{++}$.

(e) Minimizing the sum of the largest k residuals:

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{k} |r|_{[i]} \\ \text{subject to} & r = Ax - b \end{array}$$

where $|r|_{[1]} \geq |r|_{[2]} \geq, ..., \geq |r|_{[m]}$ are the numbers $|r_1|, |r_2|, ..., |r_m|$ sorted in decreasing order. (For $k = 1$ this reduces to l_∞ - norm approximation; for $k = m$ this reduces to l_1 norm approximation.) Hint: See exercise 5.19.

Solution:

(a) Introduce the variable $y \in R^m$ so that $|a_i^T x - b_i| \leq |a + y_i|$, $y_i \geq 0$ then we have equivalent problem:

$$\begin{array}{ll} \text{minimize} & \mathbf{1}^T y \\ \text{subject to} & -y - a\mathbf{1} \preceq Ax - b \preceq y + a\mathbf{1} \\ & y \succeq 0 \end{array}$$

(b)

Introduce the variable $y = Ax - b$, $t_i \leq (1 - y_i/a)(1 + y_i + a)$, and add inequality $-a \leq y_i \leq a$ this problem will transform to:

$$\begin{array}{ll} \text{maximize} & \prod_{i=1}^{m} t_i^2 \\ \text{subject to} & y = Ax - b \\ & t_i \leq (1 - y_i/a)(1 + y_i/a),\ i = 1, ..., m \\ & -1 \leq y_i/a \leq 1,\ i = 1, ..., m \end{array}$$

with variables $t, y \in R^m$, $x \in R^n$. Next assume that $m = 2^k$. It can be done without loss of generality, as in other case we always can add missing components of y_i as unity values ($a_i = 0$ and $b_i = -1$). Lets transform this problem for $m = 4$.

$$\begin{array}{ll} \text{maximize} & (t_1 t_2 t_3 t_4)^2 \\ \text{subject to} & y = Ax - b \\ & t_i \leq (1 - y_i/a)(1 + y_i/a),\ i = 1, ..., m \\ & -1 \leq y_i/a \leq 1,\ i = 1, ..., m \end{array}$$

this problem is equivalent to:

$$\begin{array}{ll}\text{maximize} & z_1 z_2 \\ \text{subject to} & \\ & z_1^2 \leq t_1 t_2 \\ & z_2^2 \leq t_3 t_4 \\ & y = Ax - b \\ & t_i \leq (1 - y_i/a)(1 + y_i/a),\ i = 1, ..., m \\ & -1 \leq y_i/a \leq 1,\ i = 1, ..., m\end{array}$$

and also as:

$$\begin{array}{ll}\text{maximize} & z \\ \text{subject to} & \\ & z^2 \leq z_1 z_2 \\ & z_1^2 \leq t_1 t_2 \\ & z_2^2 \leq t_3 t_4 \\ & y = Ax - b \\ & t_i \leq (1 - y_i/a)(1 + y_i/a),\ i = 1, ..., m \\ & -1 \leq y_i/a \leq 1,\ i = 1, ..., m\end{array}$$

as it is easy to show that $x^T x \leq yz$ where $x \in R^n$, $y, z \in R_+$ is equivalent to:

$$\left\| \begin{bmatrix} x \\ y - z \end{bmatrix} \right\|_2 \leq y + z$$

overriding the first three inequalities with their norm analogue we have:

$$
\begin{aligned}
&\text{minimize} && -z \\
&\text{subject to} && \\
& && \left\| \begin{bmatrix} z \\ z_1 - z_2 \end{bmatrix} \right\|_2 \leq z_1 + z_2 \\
& && \left\| \begin{bmatrix} z_1 \\ t_1 - t_2 \end{bmatrix} \right\|_2 \leq t_1 + t_2 \\
& && \left\| \begin{bmatrix} z_2 \\ t_3 - t_4 \end{bmatrix} \right\|_2 \leq t_3 + t_4 \\
& && y = Ax - b \\
& && t_i \leq (1 - y_i/a)(1 + y_i/a),\ i = 1, ..., m \\
& && -1 \leq y_i/a \leq 1,\ i = 1, ..., m
\end{aligned}
$$

which is Second Order Cone Program (SOCP).

(c)

Lets show that this problem is equivalent to QP:

$$
\begin{aligned}
&\text{minimize} && \sum_{i=1}^{m} (u_i^2 + 2Mv_i) \\
&\text{subject to} && -u - v \preceq Ax - b \preceq u + v \\
& && 0 \preceq u \preceq M\mathbf{1} \\
& && v \succeq 0
\end{aligned}
$$

Proof: Lets fix x in our QP. For the optimum point we must have $u_i + v_i = |a_i^T x - b_i|$. In other case, if $u_i + v_i > |a_i^T x b_i|$ and $0 \leq u_i \leq M$ and $v_i \geq 0$, then as u_i and v_i are not both zero, we can decrease u_i and/or v_i without violating the constraints and the objective will be decreased also. So, at the optimum we have:

$$v_i = |a_i^T x - b_i| - u_i$$

Eliminating v_i yields equivalent problem:

$$
\begin{aligned}
&\text{minimize} && \sum_{i=1}^{m} (u_i^2 - 2Mu_i + 2M|a_i^T x - b_i|) \\
&\text{subject to} && 0 \preceq u_i \preceq min(M,\ |a_i^T x - b_i|)
\end{aligned}
$$

It $M > |a_i^T x - b_i|$ the optimal choice for u_i is $|a_i^T x - b_i|$. In this case the objective function reduces to $|a_i^T x - b_i|^2$. Otherwise the optimal choice for u_i is M, and the objective function reduces to $2M|a_i^T x - b_i| - M^2$. So, we conclude that with

$\phi(a_i^T x - b_i)$ these problems are equivalent.
(c) The constraint $ta_i^T x \geq b_i$, $t \geq 0$, $a_i^T x \geq 0$ can be formulated as an LMI

$$\begin{bmatrix} t & \sqrt{b_i} \\ \sqrt{b_i} & a_i^T x \end{bmatrix} \succeq 0$$

or as follows:

$$\left\| \begin{bmatrix} 2\sqrt{b_i} \\ t - a_i^T x \end{bmatrix} \right\|_2 \leq t + a_i^T x$$

(e) As in exercise 5.19, we have a problem:

$$\begin{array}{ll} \text{minimize} & kt + \mathbf{1}z \\ \text{subject to} & -t\mathbf{1} - z \preceq Ax - b \preceq t\mathbf{1} + z \\ & z \succeq 0 \end{array}$$

where $x \in R^n$, $t \in R$, $z \in R^m$

6.8 a - b

Formulate the following robust approximation problems as LPs, QPs, SOCPs, or SDPs. For each subproblem, consider the l_1-, l_2-, and the $l_\infty-$ norms.

(a) Stochastic robust approximation with a finite set of parameter values, i.e., the sum of - norms problem

$$\text{minimize} \quad \sum_{i=1}^{k} p_i \left\| A_i x - b \right\|$$

where $p \succeq 0$ and $\mathbf{1}^T p = 1$. (See §6.4.1.)

(b) Worst-case robust approximation with coefficient bounds:

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

where $\mathcal{A} = \{A \in R^{m \times n} | l_{ij} \leq a_{ij} \leq u_{ij}, i = 1, ..., m, j = 1, ..., n\}$. Here the uncertainty set is described by giving upper and lower bounds for the components of A. We assume $l_{ij} < u_{ij}$.

Solution:
(a) l_1 norm: Introduce a slack variable $y : |y_i| \succeq |A_i x - b|$. We have LP:

$$\begin{array}{ll} \text{minimize} & p^T y \\ \text{subject to} & -y_i \preceq A_i x - b \preceq y_i \end{array}$$

l_2 norm: Introduce a slack variable $y : y_i \succeq ||A_i x - b||_2$. We have SOCP:

$$\begin{array}{ll} \text{minimize} & p^T y \\ \text{subject to} & ||A_i x - b|| \preceq y_i \end{array}$$

l_∞ norm: We have LP:

$$\begin{array}{ll} \text{minimize} & p^T y \\ \text{subject to} & -y_i \mathbf{1} \preceq A_i x - b \preceq y_i \mathbf{1} \end{array}$$

(b)

$$\sup_{l_{ij} \le a_{ij} \le u_{ij}} |a_i^T x - b_i| = \sup_{l_{ij} \le a_{ij} \le u_{ij}} max(-a_i^T x + b_i, a_i^T x - b_i) = max(\sup_{l_{ij} \le a_{ij} \le u_{ij}} -a_i^T x + b_i, \sup_{l_{ij} \le a_{ij} \le u_{ij}} a_i^T x - b_i)$$

$$\sup_{l_{ij} \le a_{ij} \le u_{ij}} -a_i^T x + b_i = \overline{-a}_i^T x + b_i + v_i^T |x|$$

where $\overline{a}_{ij} = (u_{ij} + l_{ij})/2$, $v_{ij} = (u_{ij} - l_{ij})/2$ and

$$\sup_{l_{ij} \le a_{ij} \le u_{ij}} a_i^T x - b_i = \overline{a}_i^T x - b_i + v_i^T |x|$$

Therefore

$$\sup_{l_{ij} \le a_{ij} \le u_{ij}} |a_i^T x - b_i| = \bar{|} a_i^T x - b_i| + v_i^T |x|$$

(a) l_1 norm.

$$\text{minimize} \qquad \sum_{i=1}^{m} (|\overline{a}_i x - b_i| + v_i^T |x|)$$

Introducing slack variables $y : |y_i| \ge |\overline{a}_i x - b_i|$ and $w : |w_i| \ge |x_i|$ we have LP:

$$\begin{array}{ll} \text{minimize} & \mathbf{1}^T(y+Vw) \\ \text{subject to} & -y \preceq \overline{A}x - b \preceq y \\ & -w \preceq x \preceq w \end{array}$$

l_2 norm.

$$\text{minimize} \quad \sum_{i=1}^{m}(|\overline{a}_i x - b_i| + v_i^T|x|)^2$$

introduce the same slack variables and variable y, w and the new variable t : $t \leq ||y + Vw||_2$ we have SOCP:

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & -y \preceq \overline{A}x - b \preceq y \\ & -w \preceq x \preceq w \\ & t \geq ||y+Vw||_2 \end{array}$$

l_∞ norm:

$$\text{minimize} \quad \max_{i=1,\dots,m}(|\overline{a}_i x - b_i| + v_i^T|x|)^2$$

this can be expressed as LP:

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & -y \preceq \overline{A}x - b \preceq y \\ & -w \preceq x \preceq w \\ & -t\mathbf{1} \preceq y + Vw \preceq t\mathbf{1} \end{array}$$

A5.4

Penalty function approximation. We consider the approximation problem

$$\text{minimize} \quad \phi(Ax+b)$$

where $a \in R^m \times n$, $b\ inR^m$, the variable is $x \in R^n$, and $\phi : R^m \to R$ is a convex penalty function that measures the quality of the approximation $Ax \approx b$. We will consider the following choices of penalty function:

(a) Euclidean norm.

$$\phi(y) = ||y||_2 = (\sum_{k=1}^{m} y_k^2)^{1/2}$$

(b) l_1 - norm:

$$\phi(y) = ||y||_1 = \sum_{k=1}^{m} |y_k|$$

(c) Sum of the largest $m/2$ absolute values:

$$\phi(y) = \sum_{k=1}^{m/2} |y_{[k]}|$$

where $y_{[1]}, y_{[2]}, ...$ denote the the absolute values of the components of y sorted in the decreasing order.

(d) A piecewise-linear penalty.

$$\phi(y) = \sum_{k=1}^{m} h(y_k), \quad h(u) = \begin{cases} 0, & |u| \leq 0.2 \\ |u| - 0.2, & 0.2 \leq |u| \leq 0.3 \\ 2|u| - 0.5, & |u| \geq 0.3 \end{cases}$$

(e) Huber penalty.

$$\phi(y) = \sum_{k=1}^{m} h(y_k), \quad h(u) = \begin{cases} u^2, & |u| \leq M \\ M(2|u| - M), & |u| \geq M \end{cases}$$

with $M = 0.2$

(f) Log-barrier penalty.

$$\phi(y) = \sum_{k=1}^{m} h(y_k), \quad h(u) = -log(1 - u^2), \quad \boldsymbol{dom}\ h = \{u \mid |u| < 1.\}$$

with $M = 0.2$

Here is the problem. Generate data A and b as follows:

```
m = 200;
n = 100;
A = randn(m,n);
b = randn(m,1);
b = b/(1.01*max(abs(b)));
```

(The normalization of b ensures that the domain of $\phi(Ax-b)$ is nonempty if we use the log-barrier penalty.) To compare the results, plot a histogram of the vector of residuals $y = Ax - b$ for each of the solutions x, using the Matlab command

```
hist(A*x-b,m/2);
```

Some additional hints and remarks for the individual problems:
(a) This problem can be solved using least-squares `(x=A\b)`.
(b) Use the CVX function `norm(y,1)`.
(c) Use the CVX function `norm_largest()`.
(d) Use CVX, with the overloaded `max()`, `abs()`, and `sum()` functions.
(e) Use the CVX function `huber()`.
(f) The current version of CVX handles the logarithm using an iterative procedure, which is slow and not entirely reliable. However, you can reformulate this problem as

$$\text{maximize} \qquad \prod_{k=1}^{m}((1-(Ax-b)_k)(1+(Ax-b)_k))^{1/2m},$$

and use the CVX function `geo_mean()`.

Solution:
(a)
Matlab code:

```
m = 200;
n = 100;
A = randn(m,n);
b = randn(m,1);
b = b/(1.01*max(abs(b)));

x = A \ b;

hist(A*x-b,m/2);
saveas(gcf,'C:/! Convex_Optimization/homework_solutions/part_1/hw5/A_5_4_a.png')
```

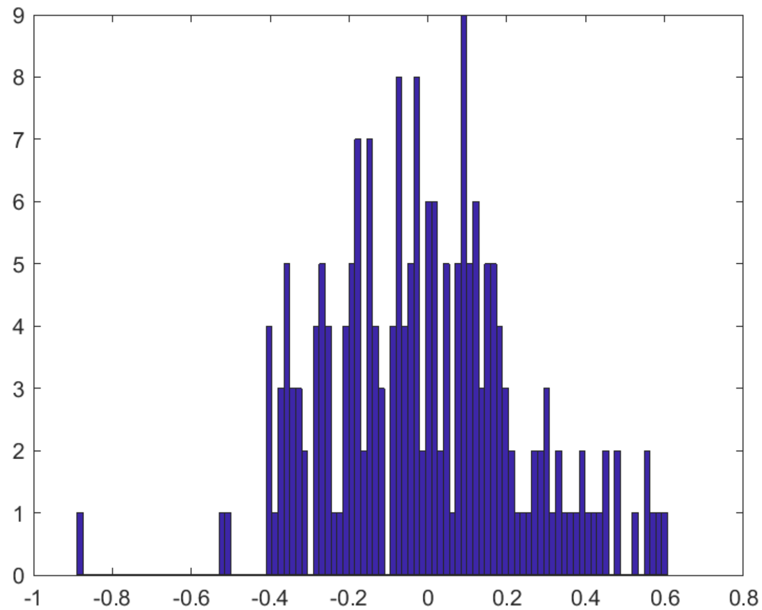

Figure 1: Euclidean norm

(b)
Matlab code:

```
m = 200;
n = 100;
A = randn(m,n);
b = randn(m,1);
b = b/(1.01*max(abs(b)));

cvx_begin
variable x(n);
variable y(m);
minimize norm(A * x - b, 1)
cvx_end

hist(A*x-b,m/2);
saveas(gcf,'C:/! Convex_Optimization/homework_solutions/part_1/hw5/A_5_4_b.png')
```

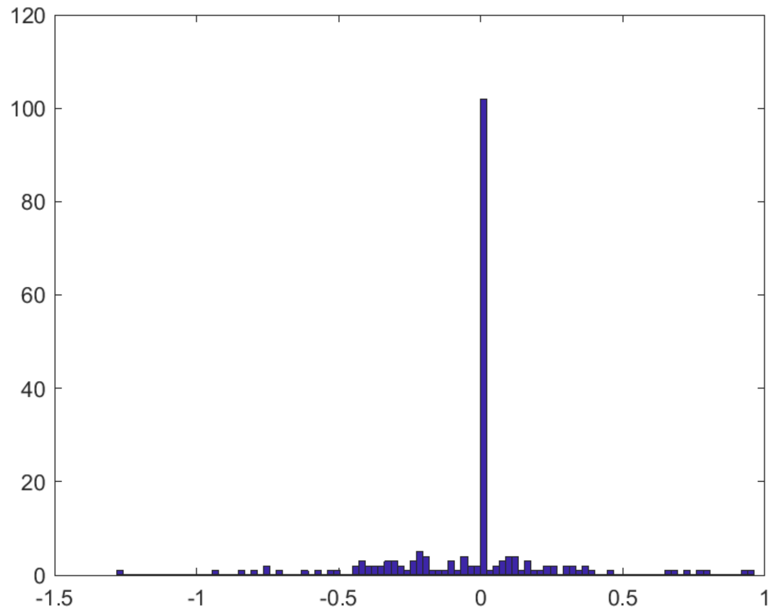

Figure 2: l_1−norm.

(c)
Matlab code:

```
m = 200;
n = 100;
A = randn(m,n);
b = randn(m,1);
b = b/(1.01*max(abs(b)));

cvx_begin
variable x(n);
variable y(m);
minimize norm_largest(A * x - b, floor(m / 2))
cvx_end

hist(A*x-b,m/2);
saveas(gcf,'C:/! Convex_Optimization/homework_solutions/part_1/hw5/A_5_4_c.png')
```

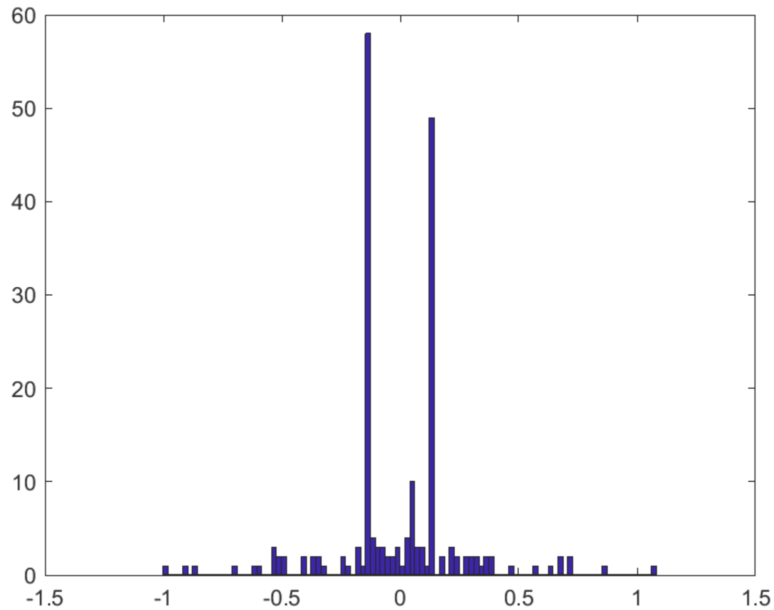

Figure 3: Sum of the largest $m/2$ absolute values.

(d)
Matlab code:

```
m = 200;
n = 100;
A = randn(m,n);
b = randn(m,1);
b = b/(1.01*max(abs(b)));

cvx_begin
variable x(n);
minimize sum(max([zeros(m), abs(A * x - b) - 0.2, 2 * abs(A * x - b) - 0.5]))
cvx_end

hist(A*x-b,m/2);
saveas(gcf,'C:/! Convex_Optimization/homework_solutions/part_1/hw5/A_5_4_d.png')
```

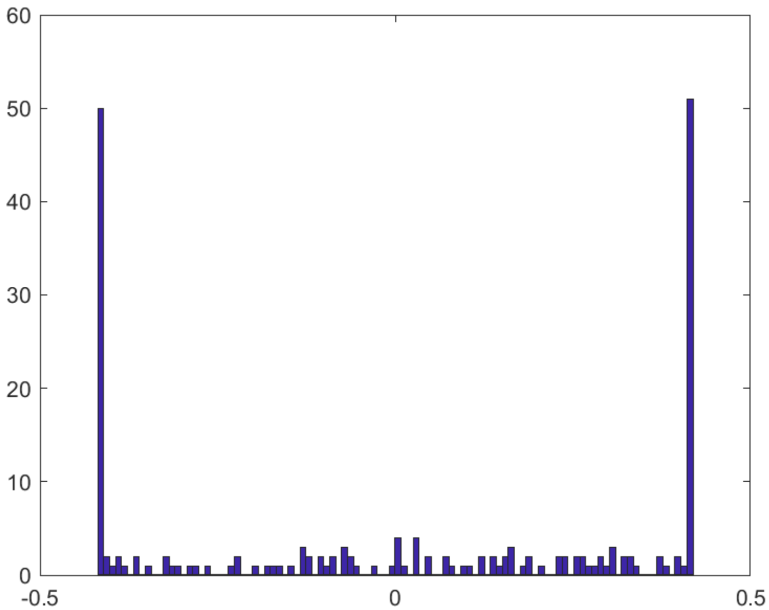

Figure 4: A piecewise-linear penalty.

(e)
Matlab code:

```
m = 200;
n = 100;
A = randn(m,n);
b = randn(m,1);
b = b/(1.01*max(abs(b)));

cvx_begin
variable x(n);
minimize sum(huber(A * x - b, 0.2))
cvx_end

hist(A*x-b,m/2);
saveas(gcf,'C:/! Convex_Optimization/homework_solutions/part_1/hw5/A_5_4_e.png');
```

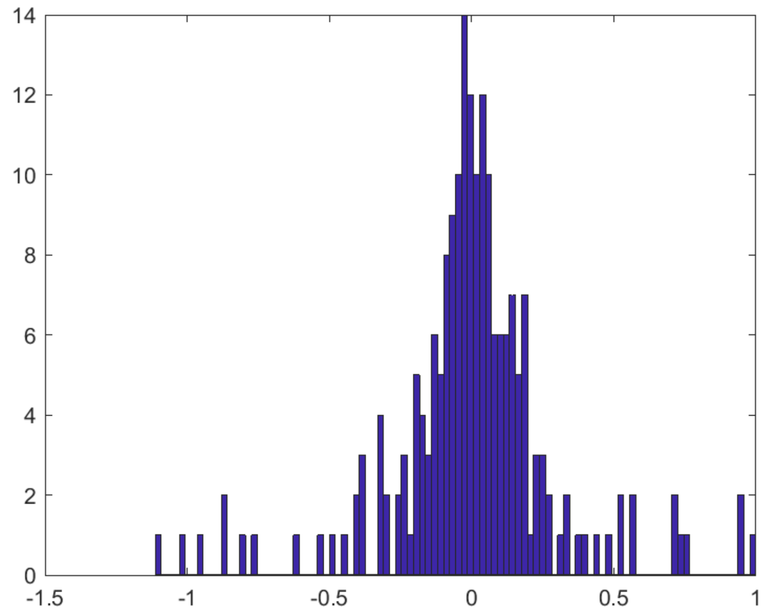

Figure 5: Huber penalty.

(f)
Matlab code:

```
m = 200;
n = 100;
A = randn(m,n);
b = randn(m,1);
b = b/(1.01*max(abs(b)));

cvx_begin
variable x(n);
minimize (- geo_mean([1 - A * x + b; 1 + A * x - b]))
cvx_end

hist(A*x-b,m/2);

saveas(gcf,'C:/! Convex_Optimization/homework_solutions/part_1/hw5/A_5_4_f.png')
```

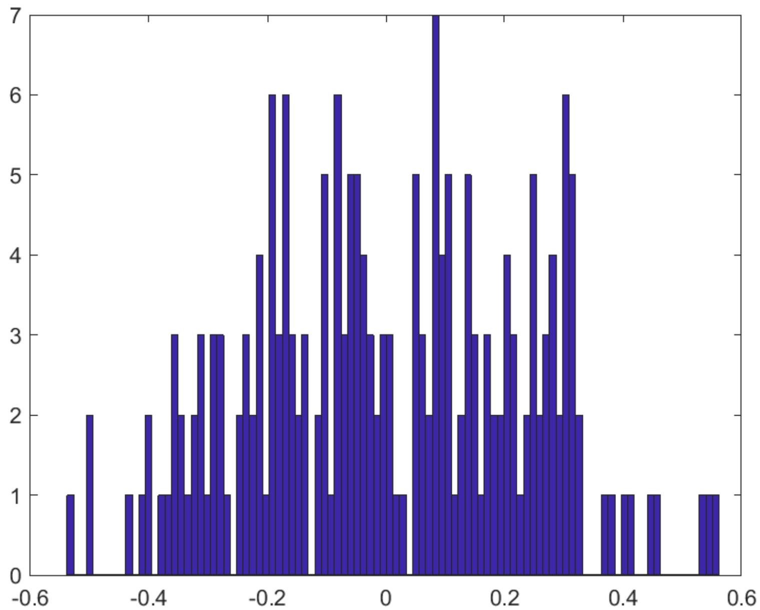

Figure 6: Log-barrier penalty.

A12.6

Antenna array weight design. We consider an array of n omnidirectional antennas in a plane, at positions $(x_k, y_k,)$ $k = 1, ..., n$. A unit plane wave with frequency ω is incident from an angle θ. This incident wave induces in the kth antenna element a (complex) signal $exp(i(x_k cos(\theta) + y_k sin(\theta) - \omega t))$ (For simplicity we assume that the spatial units are normalized so that the wave number is one, i.e., the wavelength is $\lambda = 2\pi$.) The baseband signals of the n antennas are combined linearly to form the output of the antenna array

$$G(\theta) = \sum_{k=1}^{n} \omega_k e^{x_k cos(\theta) + y_k sin(\theta)}$$
$$= \sum_{k=1}^{n} (\omega_{re,k} cos(\gamma_k(\theta)) - \omega_{im,k} sin(\gamma_k(\theta)) + (\omega_{re,k} sin(\gamma_k(\theta)) + \omega_{im,k} cos(\gamma_k(\theta))$$

if we define $\gamma_k(\theta) = x_k cos(\theta) + y_k sin(\theta)$ The complex weights in the linear combination, $\omega_k = \omega_{re,k} + i\omega_{im,k}$ are called the antenna array coefficients or shading coefficients, and will be the design variables in the problem. For a given set of weights, the combined output $G(\theta)$. is a function of the angle of arrival θ. of the plane wave. The design problem is to select weights ω_i that achieve a desired directional pattern $G(\theta)$. We now describe a basic weight design problem. We require unit gain in a target direction θ^{tar}, i.e., $G(\theta^{tar}) = 1$. We want $G(\theta)$ small for $|\theta - \theta^{tar}| \geq \Delta$. This number is called the sidelobe level for the array; our goal is to minimize the sidelobe level. If we achieve a small sidelobe level, then the array is relatively insensitive to signals arriving from directions more than Δ away from the target direction. This results in the optimization problem:

$$\begin{array}{ll} \text{minimize} & \max_{|\theta - \theta^{tar}| \geq \Delta} |G(\theta)| \\ \text{subject to} & G(\theta^{tar}) = 1 \end{array}$$

with $\omega \in C^n$ as variables.

The objective function can be approximated by discretizing the angle of arrival with (say) N values (say, uniformly spaced) $\theta_1, ..., \theta_n$ over an interval $[-\pi, \pi]$, and replacing the objective with

$$max\{G(\theta_k) \,|\, |\theta_k - \theta^{tar}| \geq \Delta\}$$

(a) Formulate the antenna array weight design problem as an SOCP.

(b) Solve an instance using CVX, with $n = 40$, $\theta^{tar} = 15°$, $\Delta = 15°$, $N = 400$, and antenna positions generated using:

```
rand('state',0);
n = 40;
x = 30 * rand(n,1);
y = 30 * rand(n,1);
```

Compute the optimal weights and make a plot of $|G(\theta)|$ (on a logarithmic scale) versus θ. Hint. CVX can directly handle complex variables, and recognizes the modulus `abs(x)` of a complex number as a convex function of its real and imaginary parts, so you do not need to explicitly form the SOCP from part (a). Even more compactly, you can use `norm(x,Inf)` with complex argument.

Solution:

The problem can be expressed as the SOCP:

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & ||A_k x||_2 \leq t, \; k \in \mathcal{I} \\ & Bx = d \end{array}$$

with variables x, t, $\mathcal{I} = \{k| \; |\theta - \theta^{tar}| \geq \Delta\}$ and

$$x = \begin{bmatrix} \omega_{re} \\ \omega_{im} \end{bmatrix} \in R^{2n}$$

$$A_k = \begin{bmatrix} cos(\gamma_1(\theta_k)) & \dots & cos(\gamma_n(\theta_k)) & -sin(\gamma_1(\theta_k)) & \dots & -sin(\gamma_n(\theta_k)) \\ cos(\gamma_1(\theta^{tar})) & \dots & cos(\gamma_n(\theta^{tar})) & -sin(\gamma_1(\theta^{tar})) & \dots & -sin(\gamma_n(\theta^{tar})) \end{bmatrix}$$

$$B_k = \begin{bmatrix} cos(\gamma_1(\theta^{tar})) & \dots & cos(\gamma_n(\theta^{tar})) & -sin(\gamma_1(\theta^{tar})) & \dots & -sin(\gamma_n(\theta^{tar})) \\ \\ sin(\gamma_1(\theta^{tar})) & \dots & sin(\gamma_n(\theta^{tar})) & cos(\gamma_1(\theta^{tar})) & \dots & cos(\gamma_n(\theta^{tar})) \end{bmatrix}$$

$$d = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

the matlab code:

```
clear all
rand('state',0);
%N= 10; %400;
%n = 2;%40;
N = 400;
n = 40;
x = 30 * rand(n,1);
y = 30 * rand(n,1);
X = [x'; y'];

beamwidth = 15*pi/180;
theta_tar=15*pi/180;
theta = linspace(theta_tar+beamwidth, 2*pi+theta_tar-beamwidth, N)';
A = exp(i * [cos(theta), sin(theta)] * X);
Atar = exp(i * [cos(theta_tar), sin(theta_tar)] * X);

cvx_begin
variable w(n) complex
minimize(max(abs(A*w)))
subject to
Atar*w == 1;
cvx_end

gamma = zeros(n, N);
for i = 1:n
for j = 1:N
gamma(i, j) = x(i) * cos(theta(j)) + y(i) * sin(theta(j));
end
end
```

```
size(gamma)

G = zeros(N, 1);

for i = 1:N
for j = 1:n
omega = w(j);
rw = real(omega);
iw = imag(omega);
g = gamma(j, i);
G(i) = G(i) + complex(rw * cos(g) - iw * sin(g), ...
(rw * sin(g) + iw * cos(g)));
end
end

abs_G = abs(G)


plot(theta, abs(G))
```

Something wrong with this task here