Convex and Nonsmooth Opt. – Problem Set 2 (Due Feb. 11)
solutions by Frederick Law

1. Prove that a function is convex if and only if its epigraph is a convex set. (You can use either the conventional definition of a convex function, or the extended definition with $f(x) = \infty$ if x is not in $\operatorname{dom} f$.)

   Solution: Recall that epi f is defined as $\operatorname{epi} f = \{(x,t) : x \in \operatorname{dom} f, t \geq f(x)\}$. Firstly suppose that f is convex. Let $(x,t),(y,s) \in \operatorname{epi} f$ and $\theta \in [0,1]$. Since f convex, then $\operatorname{dom} f$ convex so $x, y \in \operatorname{dom} f$ means $\theta x + (1-\theta)y \in \operatorname{dom} f$. Moreover by convexity of f and $(x,t),(y,s) \in \operatorname{epi} f$,

   $$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y) \leq \theta t + (1-\theta)s \quad \Longrightarrow \quad \theta \begin{pmatrix} x \\ t \end{pmatrix} + (1-\theta) \begin{pmatrix} y \\ s \end{pmatrix} \in \operatorname{epi} f$$

   so epi f is a convex set.

   Conversely, suppose that epi f is a convex set. Note that for any $x \in \operatorname{dom} f$ then $(x, f(x)) \in \operatorname{epi} f$. So let $x, y \in \operatorname{dom} f$ with $\theta \in [0,1]$. Then

   $$\operatorname{epi} f \text{ convex} \quad \Longrightarrow \quad \theta \begin{pmatrix} x \\ f(x) \end{pmatrix} + (1-\theta) \begin{pmatrix} y \\ f(y) \end{pmatrix} = \begin{pmatrix} \theta x + (1-\theta)y \\ \theta f(x) + (1-\theta)f(y) \end{pmatrix} \in \operatorname{epi} f$$

   which tells us that $\theta x + (1-\theta)y \in \operatorname{dom} f$, so $\operatorname{dom} f$ convex and it also tells us that $f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y)$ so f is convex. □

2. (BV Ex 2.31a,b) *Properties of dual cones.* Let K^* be the dual cone of a convex cone K, as defined in (2.19). Prove the following.

   (a) K^* is indeed a convex cone.

   Solution: Suppose $y \in K^*$ so that $x^T y \geq 0$ for all $x \in K$. Then for $\theta \geq 0$, $x^T(\theta y) = \theta\left(x^T y\right) \geq 0$ for all $x \in K$ so $\theta y \in K^*$. Hence K^* is a cone. To see that K^* is convex, let $y_1, y_2 \in K^*$ with $\theta \in [0,1]$. Since $\theta, 1-\theta \geq 0$ and for any $x \in K$ we have $x^T y_1, x^T y_2 \geq 0$, then $x^T(\theta y_1 + (1-\theta)y_2) = \theta x^T y_1 + (1-\theta)x^T y_2 \geq 0$. So $\theta y_1 + (1-\theta)y_2 \in K^*$, hence K^* is a convex cone. □

   (b) $K_1 \subseteq K_2$ implies $K_2^* \subseteq K_1^*$.

   Solution: Suppose $K_1 \subseteq K_2$. Let $y \in K_2^*$. Then $x^T y \geq 0$ for all $x \in K_2$. Since $K_1 \subseteq K_2$ then $x^T y \geq 0$ for all $x \in K_1$, hence $y \in K_1^*$. So $K_2^* \subseteq K_1^*$. □

3. Show that if a convex cone K is closed then $(K^*)^*$, the dual cone of the dual cone of K, is equal to K.

   Solution: Suppose K is closed. First we show that $K \subseteq (K^*)^*$. Consider $x_0 \in K$, and $y \in K^*$. By definition, $x^T y \geq 0$ for all $x \in K$, so $x_0^T y \geq 0$. Hence for any $y \in K^*$, $y^T x_0 = x_0^T y \geq 0$, so $x_0 \in (K^*)^*$.

   Next we show that $(K^*)^* \subseteq K$. Suppose not, so that there exists $x_0 \in (K^*)^*$ but $x_0 \notin K$. Since K is closed, by the separating hyperplane theorem this means that we can *strictly* separate K and x_0. That means there exists a such that $a^T x < a^T x_0$ for all $x \in K$.

   We claim that this implies $a^T x \leq 0$ for all $x \in K$. To see why this is true, if there existed $\tilde{x} \in K$ such that $a^T \tilde{x} > 0$, then since K is a cone, choose $\theta > \left|a^T x_0\right| / a^T \tilde{x} \geq 0$ and $\theta \tilde{x} \in K$. However that would imply $a^T(\theta \tilde{x}) > a^T x_0$ which is not possible. Hence it must be that $a^T x \leq 0$ for all $x \in K$.

   Moreover K is a cone and thus $0 \in K$, so choosing $x = 0$ we get $a^T x_0 > 0$. But since $(-a)^T x \geq 0$ for all $x \in K$, then $-a \in K^*$. And since $x_0 \in (K^*)^*$ then $(-a)^T x_0 \geq 0$ which implies $a^T x_0 < 0$ which is a contradiction. Hence we deduce that $(K^*)^* \subseteq K$ and therefore $K = (K^*)^*$. □

4. (BV Ex 2.32) Find the dual cone of $\{Ax : x \succeq 0\}$ where $A \in \mathbb{R}^{m\times n}$.

   Solution: Let $K = \{Ax : x \succeq\}$. Then K is the image in $\mathbb{R}^m$ of the non-negative orthant $\mathbb{R}^n_+$ under A. We claim that K^* is the preimage of $\mathbb{R}^n_+$ under A^T which map $\mathbb{R}^m \to \mathbb{R}^n$. To see this, suppose $y \in K^*$. Then $y^TAx \geq 0$ for all $x \in \mathbb{R}^n_+$. So $x^TA^Ty \geq 0$ for all $x \in \mathbb{R}^n_+$. We claim this implies $A^Ty \in \mathbb{R}^n_+$. Suppose that $A^Ty \notin \mathbb{R}^n_+$, that is $\left(A^Ty\right)_i < 0$ for some index i. Then we can choose $e_i \in \mathbb{R}^n_+$ which would give us $e_i^TA^Ty = \left(A^Ty\right)_i < 0$ which would be a contradiction to $y \in K^*$. Hence $y \in K^*$ implies $A^Ty \in \mathbb{R}^n_+$.

   Conversely, suppose $y \in \mathbb{R}^m$ such that $A^Ty \in \mathbb{R}^n_+$. Then each coordinate of A^Ty is non-negative, so for any $x \in \mathbb{R}^n_+$, $x^TA^Ty \geq 0$ as it is just the sum and product of non-negative numbers. So this implies $y^TAx \geq 0$ for all $x \in \mathbb{R}^n_+$, and hence $y \in K^*$. Therefore we deduce that $y \in K^* \iff A^Ty \in \mathbb{R}^n_+$, so K^* is the preimage of $\mathbb{R}^n_+$ under A^T. □

5. Show that the second-order cone defined on p.31 of BV is self-dual, that is, it satisfies $K^* = K$.

   Solution: Let $C = \{(x,t) : ||x||_2 \leq t\}$ be the second-order cone. First we show $C \subseteq C^*$. Fix $(x,t) \in C$ and consider any $(y,s) \in C$. By Cauchy-Schwarz, $|x^Ty| \leq ||x||_2||y||_2$, and since $(x,t),(y,s) \in C$ then $||x||_2 \leq t$ and $||y||_2 \leq s$. Hence $|x^Ty| \leq ts$ so $-x^Ty \leq ts$ and thus $x^Ty + ts \geq 0$. And so

$$\begin{pmatrix} x \\ t \end{pmatrix}^T \begin{pmatrix} y \\ s \end{pmatrix} = x^Ty + ts \geq 0 \quad \forall \begin{pmatrix} y \\ s \end{pmatrix} \in C \quad \Longrightarrow \quad \begin{pmatrix} x \\ t \end{pmatrix} \in C^*$$

   so $C \subseteq C^*$.

   To show $C^* \subseteq C$, we equivalently show $C^c \subseteq (C^*)^c$, where A^c is the complement of A. Suppose $(x,t) \notin C$, so $||x||_2 > t$. Note that $(-x, ||x||_2) \in C$. Taking the inner product,

$$\begin{pmatrix} x \\ t \end{pmatrix}^T \begin{pmatrix} -x \\ ||x||_2 \end{pmatrix} = -x^Tx + t||x||_2 = -||x||_2^2 + t||x||_2 = (t - ||x||_2)\,||x||_2 < 0$$

   since by assumption $||x||_2 > t$. Hence $(x,t) \notin C^*$. So $C^c \subseteq (C^*)^c$ thus $C^* \subseteq C$ and we deduce that C is self-dual. □

6. (MATLAB Question, CVX)

   (a) Write a MATLAB function with input arguments A and b (type "help function" at the prompt for more information), where A is a matrix with 2 rows and the same number of columns as the number of rows in the column vector b. This function should generalize the code on the web page by allowing an arbitrary number of inequalities. Note: $A(:,k)$ is the kth column of the matrix A. Include a modified version of the plotting commands inside the function and return x_c and r as output arguments. Setting A and b appropriately, rerun the example on the web page, checking that you get the right answer, and generate one additional example with more inequalities but for which there is still a point inside the polyhedron.

   Solution: Modifying the example code, we created a function `cheb_cent(A,b,p)`, a function of three variables A, b, p (can pass in only A, b with the default being $p = 2$, see part (c)). It creates CVX constraints by running a loop on the columns of A to define the LP that CVX then solves. After this we plot the solution and the inequalities. Running the example using

$$A = \begin{pmatrix} 2 & 2 & -1 & -1 \\ 1 & -1 & 2 & -2 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

we get the exact same solution as the example code, which serves as a verification of our code. This is to be expected as in our generalization we are executing essentially the same script as the example code when the same variables are passed in. Next we ran the example of

$$\widetilde{A} = \begin{pmatrix} -1 & 1 \\ -1 & 1 \end{pmatrix}, \quad \widetilde{b} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

which is the strip defined by $\{(x_1, x_2) : -1 \leq x_1 + x_2 \leq 1\}$. Analytically the width of this strip is the distance from $(-0.5, -0.5)$ to $(0.5, 0.5)$ which is $\sqrt{2}/2$. Running our program we get the correct result, with $x_c = (0, 0)$. Note that in this problem there is no unique x_c as we can slide the 2-ball along the line $y = -x$ since the strip is unbounded. We plot the results of solving with both A, b and $\widetilde{A}, \widetilde{b}$ in Fig. 1.

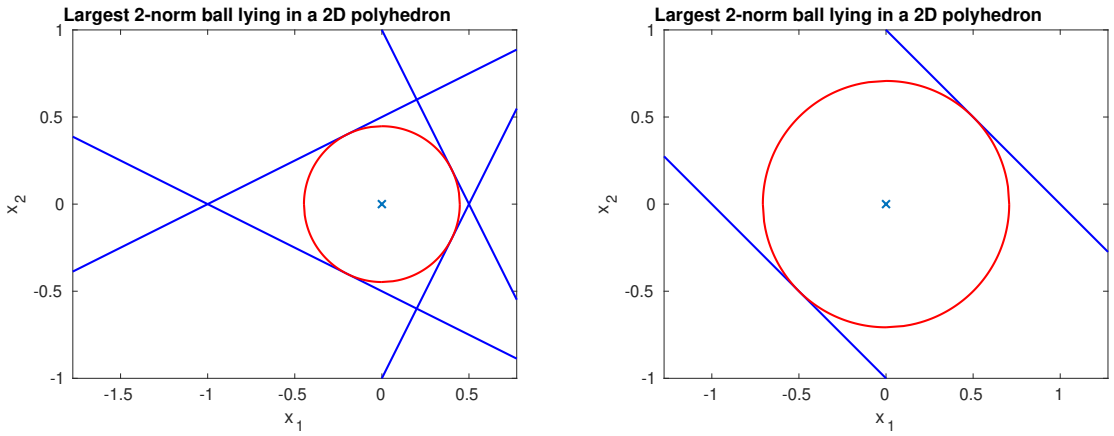

Figure 1: Samples of solutions for $p = 2$ using our code `cheb_cent`. **Left:** Verification of the code against the example CVX code. Numerically we get the exact same solution (full precision of 16 digits). This is the example with A, b. **Right:** Testing the code on a 2D polyhedron with nonempty interior. In this case the strip $\{(x_1, x_2) : -1 \leq x_1 + x_2 \leq 1\}$, corresponds to $\widetilde{A}, \widetilde{b}$.

Code printouts of both `cheb_cent` and the script for code verification and the example with $\widetilde{A}, \widetilde{b}$ (named `cheby_test.m`) are included at the end of the problem set.

(b) What happens if you choose A and b so there is no point inside the polyhedron?

Solution: Now we choose a problem such that there is no point in the feasible region. We take the above example of $\widetilde{A}, \widetilde{b}$ and flip the signs, so we solve in the exterior of the strip. This is the region $\{(x_1, x_2) : x_1 + x_2 \geq 1, x_1 + x_2 \leq -1\}$. This is done by passing in $\widehat{A}, \widehat{b}$ where

$$\widehat{A} = \begin{pmatrix} -1 & 1 \\ -1 & 1 \end{pmatrix}, \quad \widehat{b} = \begin{pmatrix} -1 \\ -1 \end{pmatrix}$$

In solving the problem, CVX returns the same ball as with the strip using $\widetilde{A}, \widetilde{b}$ from part (a), but now with $-\sqrt{2}/2$ as the radius. So when the feasible region is empty, it seems that CVX flips the inequalities until the feasible region is not empty solves there to find the radius, and then flips the sign. In plotting it still returns the same plot as in Fig. 1.

The code printout for this is part of the script `cheby_test2.m` which also contains the code for parts (c),(d). It is attached at the end of the problem set.

(c) Let's change the problem so that, given p defining a p-norm (see p.635-637), we want to find the largest "scaled unit ball" in the p-norm that lies inside the specified polyhedron. How do

equations (4.30)-(4.31) change? Is the optimization problem still an LP? Add a third input argument, p, to your MATLAB function, and change the code appropriately, including the code that plots the largest scaled "unit ball" appropriately. Note that norm(x,p) computes the p-norm of x. Submit the function listing as well as printed plots for the following choices of p: $p = 1, p = 1.5$ and $p = \infty$. (In MATLAB, inf is a valid floating point number which is, for example, equal to $1/0$.)

Solution: We altered our script `cheby_cent` to allow for p as the third optional input. With no value of p chosen the default is 2. The equations for the LP change in the following way. Consider $1 \leq p \leq \infty$ and let $B_p(x, r)$ denote the ball of radius r in the p-norm, centered at x. Then

$$B_p(x_c, r) = \{x_c + u \,:\, ||u||_p \leq r\}$$

Suppose we are at a feasible region in the variables (x_c, r). Then $a_i^T(x_c + u) \leq b_i$ becomes $a_i^T x_c + a_i^T u \leq b_i$. In the case of $p = 2$ we use that $\sup_{||u||_2 \leq r} a_i^T u = ||a||_2 r$, where the sup is attained. Now using Hölder's inequality we have

$$a_i^T u \leq ||a_i||_{p^*} ||u||_p \quad \Longrightarrow \quad \sup_{||u||_p \leq r} a_i^T u = ||a_i||_{p^*} r$$

where again we are guaranteed that the sup is attained.

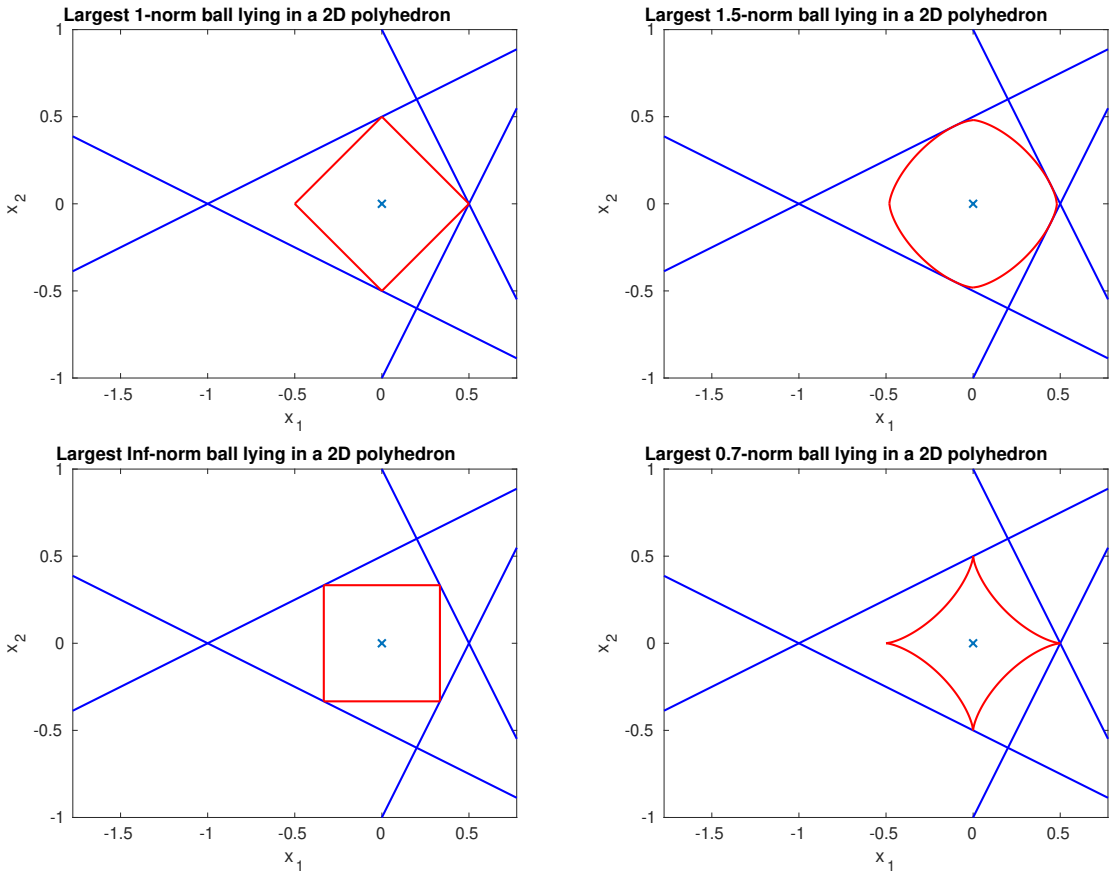

Figure 2: Solutions using the original region A, b from part (a) with different norms. **Top left:** $p = 1$-norm. **Top right:** $p = 1.5$-norm. **Bottom left:** $p = \infty$-norm. **Bottom right:** $p = 7/10$-norm.

Here p^* is the Hölder conjugate: $\frac{1}{p^*}+\frac{1}{p}=1$. So with $p=1, p^*=\infty$ and with $p=\infty, p^*=1$. Note that if $p=2$ then $p^*=2$. So generically we solve the LP

$$\text{maximize } r \qquad \text{subject to } a_i^T x_c + ||a_i||_{p^*} r \leq b_i$$

Note that this is still an LP as the objective function and constraints are linear in the variables (x_c, r). We tested various $p = 1, 1.5, \infty$ using the region given by A, b from the original problem in part (a). The results are shown in Fig. 2.

The code printout for this is part of the script `cheby_test2.m` which also contains the code for parts (b),(d). It is attached at the end of the problem set.

(d) Does this extend to working for $p < 1$, for which MATLAB's norm(x,p) is still well defined? If not exactly what goes wrong?

Solution: Note that the problem *does not* directly extend to $0 < p < 1$. This is because the Hölder conjugate is only defined for $1 \leq p \leq \infty$. However, we can still technically solve the problem by solving using the 1-norm. We claim that for $0 < p < 1$, then $B_p(x_c, r) \subset B_1(x_c, r)$. This would imply that the largest p-norm ball must the same as the largest 1-ball.

Since we have linear constraints, the largest 1-ball must touch the boundary of the polyhedron at at least one of the 4 corners of the 1-ball "square" (corresponding to $(\pm r, 0)$ and $(0, \pm r)$). The p-norm ball of the same radius also has these corners, and $B_p(x_c, r) \subset B_1(x_c, r)$ would imply that the max radius r_p^* for $p < 1$ must be at least that of the max radius for $p = 1$, r_1^*. Note however, it cannot be strictly larger. If $r_p^* > r_1^*$, then this would mean that $B_1(x_c, r_1^*)$ does not have its corners touching the boundary of the polyhedron (would be at least a distance of $r_p^* - r_1^*$ away), which would imply that it could be enlargened, which contradicts that r_1^* should be maximal. So $r_1^* = r_p^*$.

We now show $B_p(x_c, r) \subset B_1(x_c, r)$ for $p < 1$. To see this, without loss of generality take x_c to be the origin. Taking $r = 1$,

$$\begin{aligned} u \in B_p(0,1) \implies |u_1|^p + |u_2|^p \leq 1 \text{ and } |u_1|, |u_2| \leq 1 \implies |u_i| \leq |u_i|^p, i = 1, 2 \\ \implies |u_1| + |u_2| \leq |u_1|^p + |u_2|^p \leq 1 \implies u \in B_1(0,1) \end{aligned}$$

Since $p < 1$ so for any $a \in [0, 1]$ then $a < a^p$. Hence we see that $B_p(0, 1) \subset B_1(0, 1)$. For general r, if $u \in B_p(0, r)$ then $u/r \in B_p(0, 1)$ so $u/r \in B_1(0, 1)$ and thus $u \in B_1(0, r)$. Thus we have shown $B_p(x_c, r) \subset B_1(x_c, r)$ for $p < 1$ and any (x_c, r).

Since the maximal radius for $p < 1$ is the same as $p = 1$, we can hard code this case into `cheb_cent` to guarantee that when $p < 1$ is passed in, we instead solve with the 1 norm to compute (x_c, r) and then plot $B_p(x_c, r)$. Note that this is still technically an LP as we are just solving the case for $p = 1$. A plot of the solution using the original A, b from part (a) with $p = 7/10$ is shown in Fig. 2.

The code printout for this is part of the script `cheby_test2.m` which also contains the code for parts (b),(c). It is attached at the end of the problem set.

References

[1] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press 2004.

```
% code by Frederick Law
function [x_c,r] = cheby_cent(A,b,p)
% Modification on the CVX code example of finding the largest p-norm
% ball contained in in a polyhedron described by A'x <= b,
% Allows an arbitrary number of inequality constraints
%
% INPUT: A -- matrix of size 2 x m, LHS of <= constraints,
%             each column a_i yields a_i'x <= b_i
%        b -- vector of size m x 1, RHS of <= constraints
% OUTPUT: x_c -- vector of size 2 x 1, center  p-norm ball
%         r -- radius of largest p-norm ball
if nargin < 3
    p = 2; %default is 2-norm
end
% Create and solve the model
cvx_begin
    variable r(1)
    variable x_c(2)
    maximize ( r )
    if p == Inf %p=Inf, conjugate is 1, cannot use p / (p-1) formula
        for i=1:size(A,2)
        a = A(:,i);
        a'*x_c + r*norm(a,1) <= b(i);
        end
    else
        q = max(p,1); %if p < 1, solve with p=1
        for i=1:size(A,2)
        a = A(:,i);
        a'*x_c + r*norm(a,q/(q-1)) <= b(i);
        end
    end
cvx_end

% Generate the figure
x = linspace(-2,2);
for i=1:size(A,2) %plot all the lines
    a = A(:,i);
    plot(x, -x*a(1)./a(2) + b(i)./a(2),'b-','LineWidth',1.5); hold on;
end
if p == Inf %plot the Inf-norm ball (square), all 4 sides
    xline = linspace(-r,r); yline = xline;
    plot(x_c(1)+xline,
 x_c(2)+r*ones(size(xline)),'r','LineWidth',1.5); hold on;
    plot(x_c(1)+xline, x_c(2)-
r*ones(size(xline)),'r','LineWidth',1.5); hold on;
    plot(x_c(1)+r*ones(size(xline)),x_c(2) +
 yline,'r','LineWidth',1.5); hold on;
    plot(x_c(1)-r*ones(size(xline)),x_c(2) +
 yline,'r','LineWidth',1.5);
else %represent x^p + y^p = r^p as a graph for 0<x<r, rotate 4 times
    xp = linspace(0,r); rot = [0 -1; 1 0]; %rotation by pi/2
    y = abs((r^p - xp.^p).^(1/p));
```

```
    for i=0:3
        temp = (rot^i)*[xp;y];
        plot(x_c(1) + temp(1,:), x_c(2) +
 temp(2,:),'r','LineWidth',1.5);
        hold on;
    end
end
plot(x_c(1),x_c(2),'x','MarkerSize',8,'LineWidth',1.5)
xlabel('x_1')
ylabel('x_2')
title(['Largest ',num2str(p),'-norm ball lying in a 2D polyhedron']);
axis([-1 1 -1 1])
axis equal
set(gca,'FontSize',12);
end
```

Published with MATLAB® R2019a

```
clear all; close all;
% Code by Frederick Law (except 'CVX Example Code' from CVX)
% cheby_test.m       for part (a)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%% CVX Example Code %%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
% Boyd & Vandenberghe, "Convex Optimization"
% Joëlle Skaf - 08/16/05
% (a figure is generated)
%
% The goal is to find the largest Euclidean ball (i.e. its center and
% radius) that lies in a polyhedron described by linear inequalites in
 this
% fashion: P = {x : a_i'*x <= b_i, i=1,...,m} where x is in R^2

% Generate the input data
a1 = [ 2;  1];
a2 = [ 2; -1];
a3 = [-1;  2];
a4 = [-1; -2];
b = ones(4,1);

% Create and solve the model
cvx_begin
    variable r(1)
    variable x_c(2)
    maximize ( r )
    a1'*x_c + r*norm(a1,2) <= b(1);
    a2'*x_c + r*norm(a2,2) <= b(2);
    a3'*x_c + r*norm(a3,2) <= b(3);
    a4'*x_c + r*norm(a4,2) <= b(4);
cvx_end

% Generate the figure
x = linspace(-2,2);
theta = 0:pi/100:2*pi;
plot( x, -x*a1(1)./a1(2) + b(1)./a1(2),'b-');
hold on
plot( x, -x*a2(1)./a2(2) + b(2)./a2(2),'b-');
plot( x, -x*a3(1)./a3(2) + b(3)./a3(2),'b-');
plot( x, -x*a4(1)./a4(2) + b(4)./a4(2),'b-');
plot( x_c(1) + r*cos(theta), x_c(2) + r*sin(theta), 'r');
plot(x_c(1),x_c(2),'k+')
xlabel('x_1')
ylabel('x_2')
title('Largest Euclidean ball lying in a 2D polyhedron');
axis([-1 1 -1 1])
axis equal
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%% My Code %%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% verification that cheby_center(A,b) does the same thing as CVX
 example
A = [a1 a2 a3 a4]; %put the same inequalities into matrix A
figure(2);
[my_x_c, my_r] = cheby_cent(A,b,2);
disp(['difference between example and my x_c in 2-norm is:
 ',num2str(norm(my_x_c-x_c))]);
disp(['difference between example and my r in magnitude is:
 ',num2str(abs(my_r-r))]);

% test on a different set of inequalities where the polyhedron is non
% empty, namely the strip defined by x+y >= -1 and x+y <= 1, contains
 at
% least the l1 ball, known that the max radius is sqrt(2)/2;
A = [-1 1; -1 1]; b = [1, 1];
figure(3);
[new_x_c,new_r] = cheby_cent(A,b);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Published with MATLAB® R2019a

```
clear all; close all;
% Code by Frederick Law
% cheby_test2.m       for parts (b),(c),(d)

% part (b)
% choose A,b so polyhedron is empty, p=2
A = [-1 1; -1 1]; b = [-1;-1]; % corresponds to x+y >= 1, x+y <=-1
figure(5);
[x_c,r] = cheby_cent(A,b);

% part (c)
% test for Chebyshev center with generic p-norm, 0 < p <= Inf
a1 = [ 2;  1];
a2 = [ 2; -1];
a3 = [-1;  2];
a4 = [-1; -2];
b = ones(4,1);

A = [a1 a2 a3 a4]; %assemble matrix

figure(1); %1-norm
[x_1, r_1] = cheby_cent(A,b,1);

figure(2); %1.5-norm
[x_1p5, r_1p5] = cheby_cent(A,b,1.5);

figure(3); %Inf-norm
[x_inf,r_inf] = cheby_cent(A,b,Inf);

% part (d)
figure(4); %p-norm, p<1
[x_p,r_p] = cheby_cent(A,b,7/10);
```

Published with MATLAB® R2019a