EE364b Homework 4

1. *Projection onto the probability simplex.* In this problem you will work out a simple method for finding the Euclidean projection y of $x \in \mathbf{R}^n$ onto the probability simplex $\mathcal{P} = \{z \mid z \succeq 0,\ \mathbf{1}^T z = 1\}$.

   Hints. Consider the problem of minimizing $(1/2)\|y - x\|_2^2$ subject to $y \succeq 0$, $\mathbf{1}^T y = 1$. Form the partial Lagrangian

   $$L(y, \nu) = (1/2)\|y - x\|_2^2 + \nu(\mathbf{1}^T y - 1),$$

   leaving the constraint $y \succeq 0$ implicit. Show that $y = (x - \nu\mathbf{1})_+$ minimizes $L(y, \nu)$ over $y \succeq 0$.

   Solution. To find the Euclidean projection of y onto $\mathcal{P}$, we solve the problem,

   $$\begin{array}{ll} \text{minimize} & (1/2)\|y - x\|_2^2 \\ \text{subject to} & y \succeq 0, \quad \mathbf{1}^T y = 1, \end{array}$$

   with variable y. The partial Lagrangian formed by 'dualizing' the constraint $\mathbf{1}^T y = 1$, is

   $$L(y, \nu) = (1/2)\|y - x\|_2^2 + \nu(\mathbf{1}^T y - 1),$$

   with the implicit constraint $y \succeq 0$. This can also be written as

   $$L(y, \nu) = (1/2)\|y - (x - \nu\mathbf{1})\|_2^2 + \nu(\mathbf{1}^T x - 1) - n\nu^2.$$

   To minimize $L(y, \nu)$ over $y \succeq 0$ we solve the problem

   $$\begin{array}{ll} \text{minimize} & (1/2)\|y - (x - \nu\mathbf{1})\|_2^2 \\ \text{subject to} & y \succeq 0, \end{array}$$

   with variable y. This is simply the Euclidean projection of $x - \nu\mathbf{1}$ onto $\mathbf{R}^n_+$, with the solution $\tilde{y} = (x - \nu\mathbf{1})_+$. Substituting $\tilde{y}$ into $L(y, \nu)$, we obtain the dual function

   $$\begin{array}{rcl} g(\nu) & = & (1/2)\|(x - \nu\mathbf{1})_+ - (x - \nu\mathbf{1})\|_2^2 + \nu(\mathbf{1}^T x - 1) - n\nu^2 \\ & = & (1/2)\|(x - \nu\mathbf{1})_-\|_2^2 + \nu(\mathbf{1}^T x - 1) - n\nu^2. \end{array}$$

   Since ν is a scalar, we can find $\nu^\star$ by using a bisection method to maximize $g(\nu)$. The projection of x onto $\mathcal{P}$ is given by $y^\star = (x - \nu^\star\mathbf{1})_+$.

2. *Minimizing expected maximum violation.* We consider the problem of minimizing the expected maximum violation of a set of linear constraints subject to a norm bound on the variable,

   $$\begin{array}{ll} \text{minimize} & \mathbf{E} \max(b - Ax)_+ \\ \text{subject to} & \|x\|_\infty \leq 1, \end{array}$$

where the data $A \in \mathbf{R}^{m\times n}$ and $b \in \mathbf{R}^m$ are random.

We consider a specific problem instance with $m = 3$ and $n = 3$. The entries of A and b vary uniformly (and independently) ± 0.1 around their expected values,

$$\mathbf{E}\, A = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1/2 & 0 \\ 1 & 1 & 1/2 \end{bmatrix}, \qquad \mathbf{E}\, b = \begin{bmatrix} 9/10 \\ 1 \\ 9/10 \end{bmatrix}.$$

(a) *Solution via stochastic subgradient.* Use a stochastic subgradient method with step size $1/k$ to compute a solution $x^{\rm stoch}$, starting from $x = 0$, with $M = 1$ subgradient sample per iteration. Run the algorithm for 5000 iterations. Estimate the objective value obtained by $x^{\rm stoch}$ using Monte Carlo, with $M = 1000$ samples. Plot the distribution of $\max(b - Ax^{\rm stoch})$ from these samples. (In this plot, points to the left of 0 correspond to no violation of the inequalities.)

(b) *Maximum margin heuristic.* The heuristic $x^{\rm mm}$ is obtained by maximizing the margin in the inequalities, with the coefficients set to their expected values:

$$\begin{array}{ll} \text{minimize} & \max(\mathbf{E}\, b - \mathbf{E}\, Ax) \\ \text{subject to} & \|x\|_\infty \le 1. \end{array}$$

Use Monte Carlo with $M = 1000$ samples to estimate the objective value (for the original problem) obtained by $x^{\rm mm}$, and plot the distribution of $\max(b - Ax^{\rm mm})$.

(c) *Direct solution of sampled problem.* Generate $M = 100$ samples of A and b, and solve the problem

$$\begin{array}{ll} \text{minimize} & (1/M)\sum_{i=1}^M \max(b^i - A^i x)_+ \\ \text{subject to} & \|x\|_\infty \le 1. \end{array}$$

The solution will be denoted $x^{\rm ds}$. Use Monte Carlo with $M = 1000$ samples to estimate the objective value (for the original problem) obtained by $x^{\rm ds}$, and plot the distribution of $\max(b - Ax^{\rm ds})$.

Hints.

- Use `x = max( min(x,1), -1 )` to project onto the ℓ_∞ norm ball.
- Use the `cvx` function `pos()` to get the positive part function $()_+$.
- The clearest code for carrying out Monte Carlo analysis uses a `for` loop. In Matlab `for` loops can be very slow, since they are interpreted. Our for-loop implementation of the solution to this problem isn't too slow, but if you find Monte Carlo runs slow on your machine, you can use the loop-free method shown below, to find the empirical distribution of $\max(b - Ax)$.

```
    % loop-free Monte Carlo with 1000 samples of data A and b
    M = 1000; noise = 0.1;
    Amtx = repmat(Abar,M,1) + 2*noise*rand(M*m,n) - noise;
    bmtx = repmat(bbar,M,1) + 2*noise*rand(M*m,1) - noise;
    % evaluate max( b - Ax ) for 1000 samples
    fvals_stoch = max( reshape(bmtx - Amtx*x_stoch,m,M) );
```

Solution.

The following code solves the problem using a stochastic subgradient method, maximum margin heuristic, and by direct solution of sampled problem.

```
n = 3; % variable size
m = 3; % number of inequalities
noise = 0.1; % amount of uncertainty

Abar = [ 1    0      0;
         1    1/2    0;
         1    1    1/2];
bbar = [9/10; 1; 9/10];

%*****************************************************************
% maximum margin heuristic: minimize max_i E(b) - E(A)x
%*****************************************************************
cvx_begin
  variable x_mm(n)
  minimize( max( bbar - Abar*x_mm ) )
  subject to
    norm( x_mm, inf ) <= 1;
cvx_end

%*****************************************************************
% direct solution of sampled problem:
%
%       minimize     1/M sum_i ( max(b^{i} - A^{i}x)_+ )
%       subject to   ||x||_inf <= 1
%*****************************************************************
M = 100;

Amtx = repmat(Abar,M,1) + 2*noise*rand(M*m,n) - noise;
bmtx = repmat(bbar,M,1) + 2*noise*rand(M*m,1) - noise;

cvx_begin
  variable x_ds(n)
  minimize( 1/M*sum( max( pos( reshape(bmtx - Amtx*x_ds,m,M) ) ) ) )
  subject to
    norm( x_ds, inf ) <= 1;
cvx_end

%*********************************************************************
% stochastic subgradient method with 1/k step size and a single sample
% used to compute a noisy (unbiased) subgradient
%*********************************************************************
MAX_ITERS = 5000;

% run subgradient method with 1/k step size
f = [+Inf]; fbest = [+Inf];

iter = 1;
x = zeros(n,1); % initial point
xhist = [x];

while iter < MAX_ITERS
  % fprintf(1,'iter = %d\n',iter);

  % noisy subgradient calculation
  M = 1;
  % generate a random realization of the data A and b
  A = repmat(Abar,M,1) + 2*noise*rand(M*m,n) - noise;
  b = repmat(bbar,M,1) + 2*noise*rand(M*m,1) - noise;
  % compute max and obtain a noisy subgradient
  [fval, ind] = max( pos(b - A*x) );
  if( fval > 0 )
    g = -A(ind,:)';
  else
    g = zeros(n,1);
  end

  % step size selection
  alpha = 1/iter;

  % store objective values
  f(end+1) = fval;
  fbest(end+1) = min( fval, fbest(end) );

  % subgradient update
  x = x - alpha*g;
  % project back onto feasible set
  x = max( min( x, 1), -1 );

  % iteration counter
  iter = iter + 1; xhist = [xhist, x];
end

x_stoch = x; % keep the last solution as the best one

%**************************************************************
% generate histograms of max(b-Ax)_+ for the three solutions
%**************************************************************
NSAMPLES = 1000;

Amtx = repmat(Abar,NSAMPLES,1) + 2*noise*rand(NSAMPLES*m,n) - noise;
bmtx = repmat(bbar,NSAMPLES,1) + 2*noise*rand(NSAMPLES*m,1) - noise;

fvals_stoch = max( reshape(bmtx - Amtx*x_stoch,m,NSAMPLES) );
fvals_mm    = max( reshape(bmtx - Amtx*x_mm,m,NSAMPLES) );
fvals_ds    = max( reshape(bmtx - Amtx*x_ds,m,NSAMPLES) );

fpvals_stoch = max( pos( reshape(bmtx - Amtx*x_stoch,m,NSAMPLES) ) );
fpvals_mm    = max( pos( reshape(bmtx - Amtx*x_mm,m,NSAMPLES) ) );
fpvals_ds    = max( pos( reshape(bmtx - Amtx*x_ds,m,NSAMPLES) ) );

fprintf('Estimated obj value for x_stoch %3.5f\n',mean(fpvals_stoch));
fprintf('Estimated obj value for x_mm    %3.5f\n',mean(fpvals_mm));
fprintf('Estimated obj value for x_ds    %3.5f\n',mean(fpvals_ds));

%**************************************************************
% histogram plots
%**************************************************************
figure(1), clf
edges = linspace(-0.4,0.4,40);
[count_1] = histc( fvals_stoch, edges )';
[count_2] = histc( fvals_mm, edges )';
[count_3] = histc( fvals_ds, edges )';

% stochastic subgradient solution
subplot(3,1,1), bar( edges, count_1, 'histc' )
title('stoch subgrad')
```

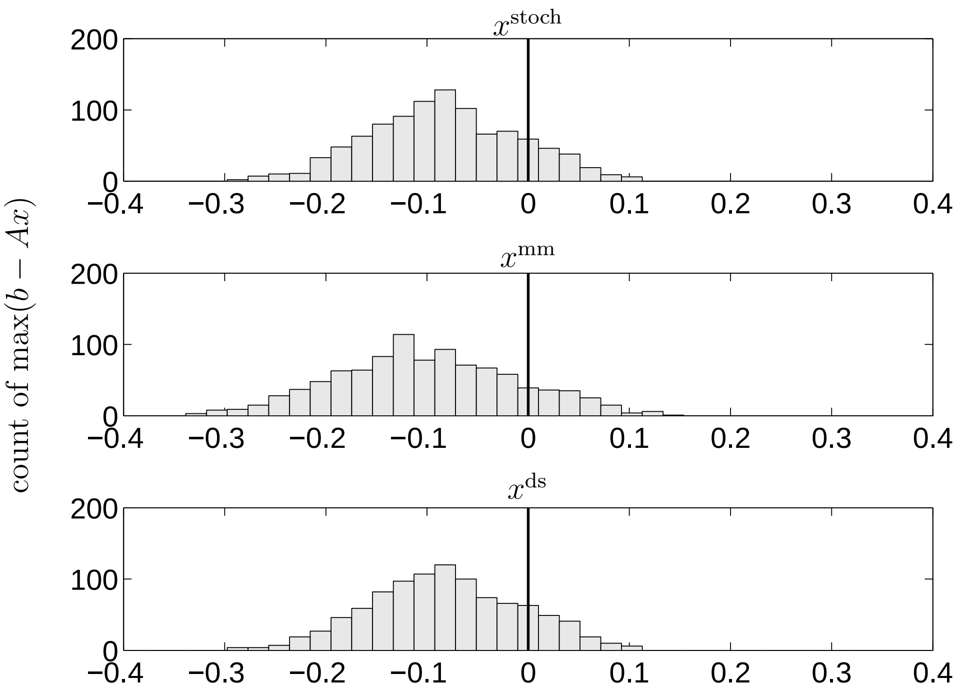
```
% maximum margin solution
subplot(3,1,2), bar( edges, count_2, 'histc' )
title('maximum margin')

% direct solution of sampled problem
subplot(3,1,3), bar( edges, count_3, 'histc' )
title('direct solution')
```

Estimated objective values for a realization of the three solutions:

```
Estimated obj value for x_stoch is 0.0051.
Estimated obj value for x_mm    is 0.0063.
Estimated obj value for x_ds    is 0.0053.
```

The empirical distributions of $\max(b - Ax)$ over $M = 1000$ Monte Carlo samples for the three solutions are shown below.



3. *Subgradient method for inequality form SDP.* Describe how to implement a subgradient method to solve the inequality form SDP

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & x_1 A_1 + \cdots + x_n A_n \preceq B, \end{array}$$

with variable $x \in \mathbf{R}^n$, and problem data $c \in \mathbf{R}^n$, $A_1, \ldots, A_n \in \mathbf{S}^m$, $B \in \mathbf{S}^m$.

Generate a small instance of the problem (say, with $n = 4$ and $m = 10$) and solve it using your subgradient method. Check your solution using CVX.

Solution. The objective is affine, so a subgradient is simply c. We can express the constraint as

$$f(x) = \lambda_{\max}(x_1A_1 + \cdots + x_nA_n - B) \leq 0.$$

When an iterate is not feasible, *i.e.*, $f(x) > 0$, we need to find a subgradient $g \in \partial f(x)$. We can write,

$$f(x) = \sup_{\|v\|\leq 1} v^T(x_1A_1 + \cdots + x_nA_n - B)v.$$

A subgradient is given by

$$g = (\tilde{v}^T A_1 \tilde{v}, \ldots, \tilde{v}^T A_n \tilde{v}),$$

where $\tilde{v}$ is the normalized eigenvector corresponding to the largest eigenvalue of $(x_1A_1 + \cdots + x_nA_n - B)$.

When an iterate $x^{(k)}$ is feasible, we use a diminishing non-summable step size, *e.g.*, $\alpha_k = 1/k$. When an iterate is not feasible, we can use Polyak's step size, *i.e.*,

$$\alpha_k = \frac{f(x^{(k)}) + \epsilon}{\|g^{(k)}\|_2^2},$$

where ϵ is small (*e.g.*, $\epsilon = 10^{-6}$), and $g^{(k)} \in \partial f(x^{(k)})$. Thus the algorithm is as follows,

$$x^{(k+1)} = \begin{cases} x^{(k)} - (1/k)c & f(x^{(k)}) \leq 0 \\ x^{(k)} - \dfrac{f(x^{(k)}) + \epsilon}{\|g^{(k)}\|_2^2} g^{(k)} & f(x^{(k)}) > 0 \end{cases}.$$

The following code solves the inequality form SDP.

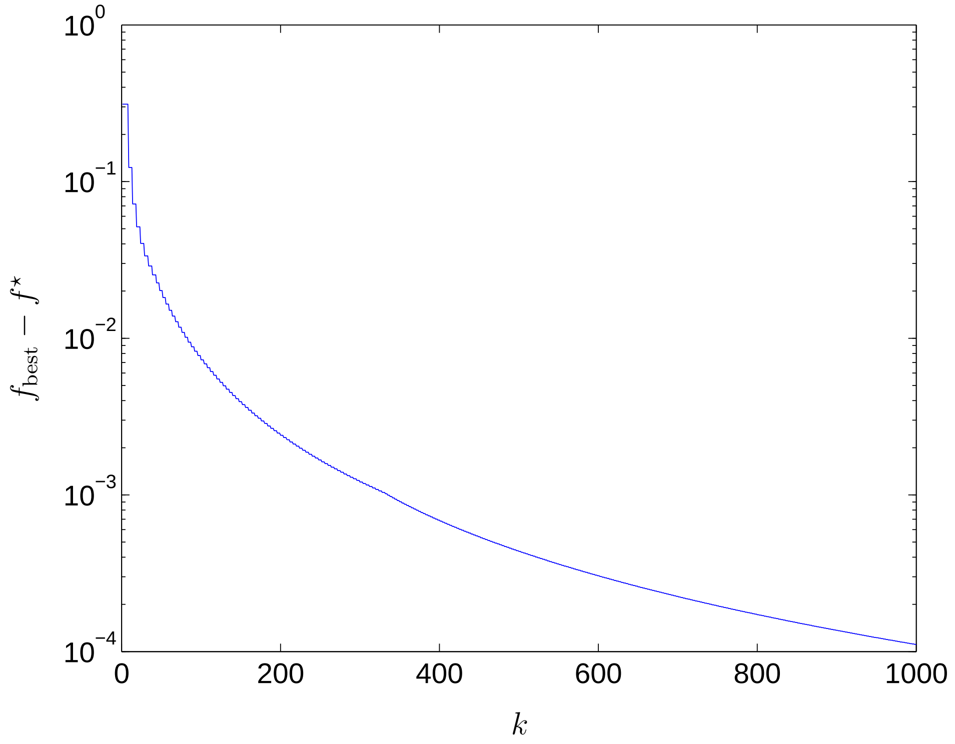
```
% generate the problem data
randn('state',0);
n = 4; m = 10; maxiter = 1000; eps = 1e-6;
A = randn(m,m,n); B = randn(m,m); B = B*B'; c = randn(n,1);
for j = 1:n A(:,:,j) = A(:,:,j)+A(:,:,j)'; end;

% solve using subgradient method
x = zeros(n,1); g = zeros(n,1); fbest = (c'*x)*ones(maxiter,1);
for k = 1:maxiter
    F = -B;
    for j = 1:n F = F+x(j)*A(:,:,j); end;
    [V,D] = eig(F);
    if (D(m,m) > 0)
        for j = 1:n g(j) = V(:,m)'*A(:,:,j)*V(:,m); end;
        x = x-((D(m,m)+eps)/(g'*g))*g;
    else
        if (c'*x < fbest(k))
            fbest(k:end) = c'*x;
        end
        x = x-(1/k)*c;
    end
end
x_subg = x;

% check solution using CVX
cvx_begin sdp
variable x(n)
F = -B;
for j = 1:n
    F = F+x(j)*A(:,:,j);
end
F <= 0;
minimize (c'*x)
cvx_end
x_cvx = x;

set(gca,'Fontsize', 16);
semilogy(fbest-cvx_optval);
xlabel('k'); ylabel('fbestmf');
print('-depsc', 'subgrad_sdp.eps');
```

The following figure shows the progress of the subgradient method with iteration number k.



4. *Kelley's cutting-plane algorithm.* We consider the problem of minimizing a convex function $f : \mathbf{R}^n \to \mathbf{R}$ over some convex set C, assuming we can evaluate $f(x)$ and find a subgradient $g \in \partial f(x)$ for any x. Suppose we have evaluated the function and a subgradient at $x^{(1)}, \ldots, x^{(k)}$. We can form the piecewise-linear approximation

$$\hat{f}^{(k)}(x) = \max_{i=1,\ldots,k} \left(f(x^{(i)}) + g^{(i)T}(x - x^{(i)}) \right),$$

which satisfies $\hat{f}^{(k)}(x) \le f(x)$ for all x. It follows that

$$L^{(k)} = \inf_{x \in C} \hat{f}^{(k)}(x) \le p^\star,$$

where $p^\star = \inf_{x \in C} f(x)$. Since $\hat{f}^{(k+1)}(x) \ge \hat{f}^{(k)}(x)$ for all x, we have $L^{(k+1)} \ge L^{(k)}$.

In Kelley's cutting-plane algorithm, we set $x^{(k+1)}$ to be any point that minimizes $\hat{f}^{(k)}$ over $x \in C$. The algorithm can be terminated when $U^{(k)} - L^{(k)} \le \epsilon$, where $U^{(k)} = \min_{i=1,\ldots,k} f(x^{(i)})$.

Use Kelley's cutting-plane algorithm to minimize the piecewise-linear function $f(x) = \max_{i=1,\ldots,m}(a_i^T x + b_i)$ that we have used for other numerical examples, with C the unit cube, *i.e.*, $C = \{x \mid \|x\|_\infty \le 1\}$. The data that defines the particular function can be found in the Matlab directory of the subgradient notes on the course web site. You can start with $x^{(1)} = 0$ and run the algorithm for 40 iterations. Plot $f(x^{(k)})$, $U^{(k)}$, $L^{(k)}$ and the constant $p^\star$ (on the same plot) versus k.

Repeat for $f(x) = \|x - c\|_2$, where c is chosen from a uniform distribution over the unit cube C. (The solution to this problem is, of course, $x^\star = c$.)

Solution. First we consider the case where f is piecewise-linear. Finding a subgradient of f is easy: given x, we first find an index j for which

$$a_j^T x + b_j = \max_{i=1,\ldots,m} (a_i^T x + b_i).$$

Then we can take as subgradient $g = a_j$.

The following MATLAB script implements Kelley's cutting-plane method for minimizing f. It generates figure 4.

```
cvx_quiet(1);

% number of variables n and linear functions m
n = 20; m = 100;

%*****************************************************************
% generate an example
%*****************************************************************
randn('state',1)
A = randn(m,n);
b = randn(m,1);

%*****************************************************************
% compute pwl optimal point using CVX
%*****************************************************************
cvx_begin
    variable x(n)
    minimize max(A*x + b)
cvx_end
f_star = cvx_optval;

%*****************************************************************
% cutting plane method
%*****************************************************************
% number of iterations
niter = 40;
% initial localization polyhedron is the cube {x| norm(x, inf) <= R}
R = 1;
% initial point
x = zeros(n,1);
G = []; h = [];
```

```
f_save = []; L = []; U = [];
for i = 1:niter
    % find active functions at current x
    [f, idx] = max(A*x + b);
    f_save = [f_save; f];
    U = [U; min(f_save)];
    % subgradient at current x
    g = A(idx(1),:)';
    G = [G; g'];
    h = [h; f - g'*x];
    cvx_begin
        variable x(n)
        minimize max(G*x + h)
        norm(x, inf) <= 1
    cvx_end
    L = [L; cvx_optval];
end

%*********************************************************************
% plots
%*********************************************************************
figure
set(gca, 'FontSize',18);
plot(1:niter, f_save);
hold on;
plot(1:niter, L, 'r-.');
plot(1:niter, U, 'g--');
plot(1:niter, f_star*ones(1,niter),'k')
xlabel('k');
legend('f','L','U','pstar', 'Location','Southeast')
```

Now we consider the case where $f(x) = \|x - c\|$. For this f, a subgradient g at $x \neq c$ is given by

$$g = (x - c)/\|x - c\|.$$

The following MATLAB script implements Kelley's cutting-plane method, and generates figure 4.

```
% number of variables n
n = 20;

%*********************************************************************
% generate an example
```

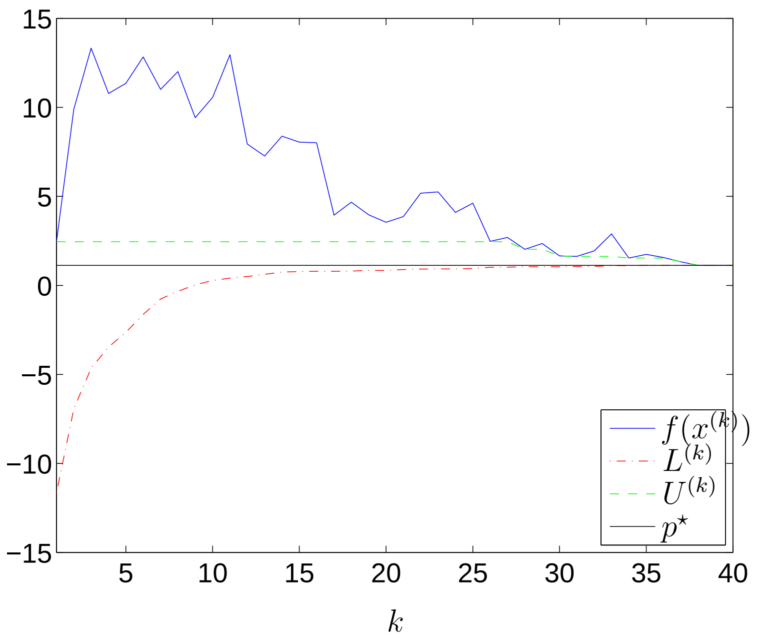


Figure 1: Convergence of Kelley's cutting-plane algorithm for minimizing a piecewise-linear function.

```
%*********************************************************************
rand('state',1)
c = 2*rand(n,1)-1;

% optimal solution
x_star = c;
f_star = 0;

%*********************************************************************
% cutting plane method
%*********************************************************************
% number of iterations
niter = 40;
% initial localization polyhedron is the cube {x| norm(x, inf) <= R}
R = 1;
% initial point
x = zeros(n,1);
G = []; h = [];
f_save = []; L = []; U = [];
for i = 1:niter
```

```
    % f and g at current x
    f = norm(x-c);
    g = (x-c)/f;
    f_save = [f_save; f];
    U = [U; min(f_save)];
    % update PWL underestimator
    G = [G; g'];
    h = [h; f - g'*x];
    cvx_begin
        variable x(n)
        minimize max(G*x + h)
        norm(x, inf) <= 1
    cvx_end
    L = [L; cvx_optval];
end

%*******************************************************************
% plots
%*******************************************************************
figure
set(gca, 'FontSize',18);
plot(1:niter, f_save);
hold on;
plot(1:niter, L, 'r-.');
plot(1:niter, U, 'g--');
plot(1:niter, f_star*ones(1,niter),'k')
xlabel('k');
legend('f','L','U','pstar', 'Location','Southeast')
```

5. *Minimum volume ellipsoid covering a half-ellipsoid.* In this problem we derive the update formulas used in the ellipsoid method, *i.e.*, we will determine the minimum volume ellipsoid that contains the intersection of the ellipsoid

$$\mathcal{E} = \{x \in \mathbf{R}^n \mid (x - x_c)P^{-1}(x - x_c) \leq 1\}$$

and the halfspace

$$\mathcal{H} = \{x \mid g^T(x - x_c) \leq 0\}.$$

We'll assume that $n > 1$, since for $n = 1$ the problem is easy.

(a) We first consider a special case: $\mathcal{E}$ is a ball centered at the origin ($P = I$, $x_c = 0$), and $g = -e_1$ (e_1 the first unit vector), so $\mathcal{E} \cap \mathcal{H} = \{x \mid x^T x \leq 1,\ x_1 \geq 0\}$.

Let

$$\tilde{\mathcal{E}} = \{x \mid (x - \tilde{x}_c)^T \tilde{P}^{-1}(x - \tilde{x}_c) \leq 1\}$$

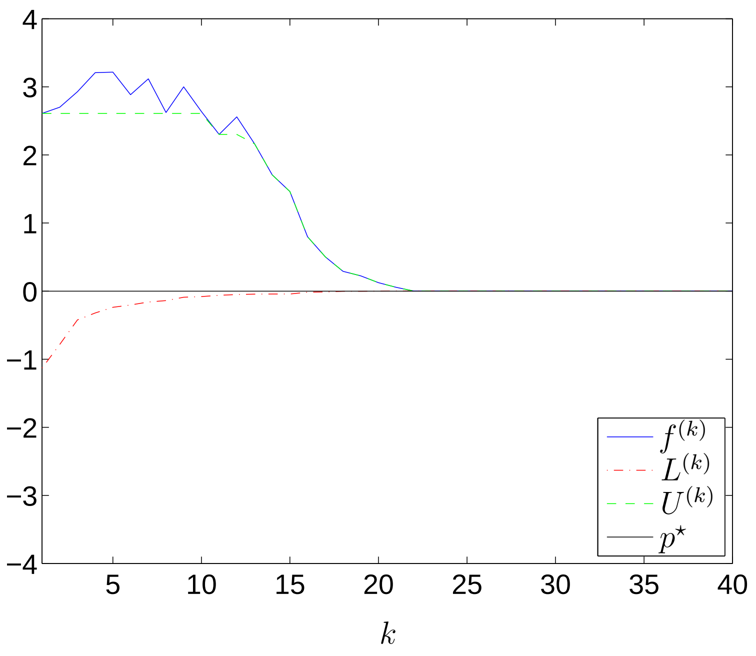


Figure 2: Convergence of Kelley's cutting-plane algorithm for minimizing $\|x - c\|$.

denote the minimum volume ellipsoid containing $\mathcal{E} \cap \mathcal{H}$. Since $\mathcal{E} \cap \mathcal{H}$ is symmetric about the line through first unit vector e_1, it is clear (and not too hard to show) that $\tilde{\mathcal{E}}$ will have the same symmetry. This means that the matrix $\tilde{P}$ is diagonal, of the form $\tilde{P} = \mathbf{diag}(\alpha, \beta, \beta, \ldots, \beta)$, and that $x_c = \gamma e_1$.

So now we have only three variables to determine: α, β, and γ. Express the volume of $\tilde{\mathcal{E}}$ in terms of these variables, and also the constraint that $\tilde{\mathcal{E}} \supseteq \mathcal{E} \cap \mathcal{H}$. Then solve the optimization problem directly, to show that

$$\alpha = \frac{n^2}{(n+1)^2}, \quad \beta = \frac{n^2}{n^2-1}, \quad \gamma = \frac{1}{n+1}$$

(which agrees with the formulas we gave, for this special case).

(b) Now consider the general case, stated at the beginning of this problem. Show how to reduce the general case to the special case solved in part (a).

Hint. Find an affine transformation that maps the original ellipsoid to the unit ball, and g to $-e_1$. Explain why minimizing the volume in these transformed coordinates also minimizes the volume in the original coordinates.

Solution.

(a) The volume of $\tilde{\mathcal{E}}$ is

$$\mathbf{vol}\,\tilde{\mathcal{E}} = \kappa_n (\det \tilde{P})^{1/2} = \kappa_n (\alpha \beta^{n-1})^{1/2},$$

where $\kappa_n > 0$ is a constant (the volume of the unit ball in $\mathbf{R}^n$). The constraint $\mathcal{E} \cap \mathcal{H} \subseteq \tilde{\mathcal{E}}$ can be written as

$$x_1^2 + \cdots + x_n^2 \leq 1, \;\; x_1 \geq 0 \Longrightarrow (x_1 - \gamma)^2/\alpha + (x_2^2 + \cdots + x_n^2)/\beta \leq 1.$$

By considering the point $x_1 = 1$, $x_2 = \cdots = x_n = 0$, we immediately see that

$$\alpha \geq (1-\gamma)^2 \tag{1}$$

is a necessary condition for $\mathcal{E} \cap \mathcal{H} \subseteq \tilde{\mathcal{E}}$. Considering points with $x_1 = 0$, $x_2^2 + \cdots + x_n^2 = 1$ we find a second necessary condition

$$\alpha^{-1}\gamma^2 + \beta^{-1} \leq 1. \tag{2}$$

It is easy to see that conditions (1) and (2) are also sufficient for $\mathcal{E} \cap \mathcal{H} \subseteq \tilde{\mathcal{E}}$: assume (1) and (2) hold, then

$$(x_1 - \gamma)^2/\alpha + (x_2^2 + \cdots + x_n^2)/\beta \leq (x_1 - \gamma)^2/\alpha + (1 - \gamma^2/\alpha)(x_2^2 + \cdots + x_n^2). \tag{3}$$

We need only consider points on the boundary of $\mathcal{E} \cap \mathcal{H}$, *i.e.*, points with either

$$x_1 = 0, \quad x_2^2 + \cdots + x_n^2 \leq 1, \tag{4}$$

or

$$x_1 \geq 0, \quad x_1^2 + x_2^2 + \cdots + x_n^2 = 1. \tag{5}$$

In the first case, inequality (3) reduces to

$$(x_1 - \gamma)^2/\alpha + (x_2^2 + \cdots + x_n^2)/\beta \leq \gamma^2/\alpha + 1/\beta \leq 1.$$

In the second case, we obtain

$$\begin{aligned}
(x_1 - \gamma)^2/\alpha + (x_2^2 + \cdots + x_n^2)/\beta &\leq (x_1 - \gamma)^2/\alpha + (1 - \gamma^2/\alpha)(1 - x_1^2) \\
&= (x_1^2 - 2x_1\gamma + \gamma^2 x_1^2)/\alpha + 1 - x_1^2 \\
&\leq x_1^2(1 - 2\gamma + \gamma^2)/\alpha + 1 - x_1^2 \\
&= x_1^2(1-\gamma)^2/\alpha + 1 - x_1^2 \\
&\leq x_1^2 + 1 - x_1^2 = 1.
\end{aligned}$$

To minimize the volume we must choose α and β that satisfy (1) and (2) with equality, *i.e.*,

$$\alpha = (1-\gamma)^2, \quad \beta = \frac{(1-\gamma)^2}{1-2\gamma}.$$

Substitute α and β in the volume expression,

$$(\mathbf{vol}\, \tilde{\mathcal{E}}/\kappa)^2 = \alpha\beta^{n-1} = \frac{(1-\gamma)^{2n}}{(1-2\gamma)^{n-1}}.$$

and set the derivative equal to zero. We obtain

$$\frac{(1-\gamma)^{2n-1}}{(1-2\gamma)^n}((2n+2)\gamma - 2) = 0.$$

Therefore, $\gamma = 1/(n+1)$ and finally

$$\alpha = \frac{n^2}{(n+1)^2}, \quad \beta = \frac{n^2}{n^2-1}.$$

(b) Assume $\mathcal{E}$ is not degenerate (hence $P^{-1} \succ 0$). Let R denote any square root of P^{-1}, *i.e.*, $P^{-1} = R^T R$. From the fact that $R^{-T}g \neq 0$ (R non-singular and $g \neq 0$) and $\mathcal{H}$ depends only on the direction of g, we can assume that $\|R^{-T}g\| = 1$ without loss of generality. Thus there exists an orthogonal matrix Q ($Q^{-1} = Q^T$) such that $QR^{-T}g = -\mathbf{e}_1$. (q is a Householder reflection $Q = I - 2vv^T/v^Tv$, with $v = R^{-T}g + \mathbf{e}_1$.)

Now let $x' = QR(x - x_c)$ be the affine transformation, then

$$(x-x_c)^T R^T R (x - x_c) \leq 1 \Longleftrightarrow x'^T x' \leq 1$$

hence the image of $\mathcal{E}$ is the unit ball. For the halfspace $\mathcal{H}$ we obtain

$$g^T R^{-1} R(x - x_c) \leq 0 \Longleftrightarrow -\mathbf{e}_1^T x' \leq 0$$

so $\mathcal{H}$ is mapped to the halfspace considered in the special case.

Since an affine transformation only changes the volume of an ellipsoid by a constant multiple (the constant is $\det(QR)$ and depends only on the transformation), minimizing the volume of an ellipsoid in the transformed coordinates is equivalent to minimizing the volume in the original coordinates.