Ellipsoid Method

S. Boyd
Notes for EE364b, Stanford University

April 1, 2018

These notes were taken from the book *Linear Controller Design: Limits of Performance*, by Boyd and Barratt [?], and edited (at various times over many years) by Stephen Boyd, Joelle Skaf, and Nicholas Moehle.

Contents

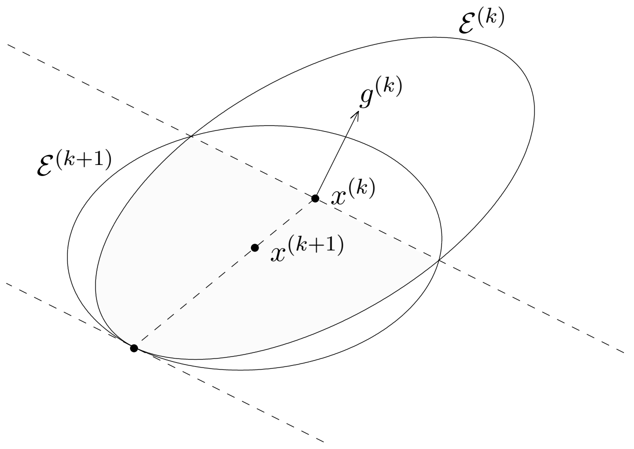

Figure 1: The shaded half ellipsoid, known to contain a minimizer, is enclosed by the ellipsoid of smallest volume, denoted $\mathcal{E}^{(k+1)}$, centered at $x^{(k+1)}$.

1 Basic ellipsoid algorithm

Ellipsoid method idea. We consider the problem of minimizing a convex function $f : \mathbf{R}^n \to \mathbf{R}$. The ellipsoid algorithm generates a "decreasing" sequence of ellipsoids in $\mathbf{R}^n$ that are guaranteed to contain a minimizing point, using the idea that given a subgradient (or quasigradient) at a point, we can find a half-space containing the point that is guaranteed not to contain any minimizers of f, *i.e.*, a cutting-plane.

Suppose that we have an ellipsoid $\mathcal{E}^{(k)}$ that is guaranteed to contain a minimizer of f. In the basic ellipsoid algorithm, we compute a subgradient $g^{(k)}$ of f at the center, $x^{(k)}$, of $\mathcal{E}^{(k)}$. We then know that the half ellipsoid

$$\mathcal{E}^{(k)} \cap \{z \mid g^{(k)T}(z - x^{(k)}) \leq 0\}$$

contains a minimizer of f. We compute the ellipsoid $\mathcal{E}^{(k+1)}$ of *minimum volume* that contains the sliced half ellipsoid; $\mathcal{E}^{(k+1)}$ is then guaranteed to contain a minimizer of f, as shown in figure 1. The process is then repeated.

We will see that the volume of the ellipsoid decreases, geometrically, which will allow us to show that the method converges, indeed with a complexity estimate that is good (at least in theory).

Special case: Bisection on R. When $n = 1$, this algorithm is the standard bisection method on $\mathbf{R}$. To see this, we note that an ellipsoid in $\mathbf{R}$ is nothing but an interval. We evaluate the derivative (or a subgradient) of f at the midpoint (which is the center), and depending on its sign, the sliced half ellipsoid is either the left or right half of the interval. In $\mathbf{R}$, a half ellipsoid is also an interval, so the minimum volume (in this case, length) ellipsoid that covers the half ellipsoid is just the interval, which has exactly half the length of the previous ellipsoid.

Ellipsoid method update. We now describe the algorithm more explicitly. An ellipsoid $\mathcal{E}$ can be described as

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

where $P \in \mathbf{S}^n_{++}$. The center of the ellipsoid is x, and the matrix P gives the size and shape or orientation of $\mathcal{E}$: the square roots of the eigenvalues of P are the lengths of the semi-axes of $\mathcal{E}$. The volume of $\mathcal{E}$ is given by

$$\mathbf{vol}(\mathcal{E}) = \beta_n \sqrt{\det P},$$

where $\beta_n = \pi^{n/2}/\Gamma(n/2+1)$ is the volume of the unit ball in $\mathbf{R}^n$.

For $n > 1$, the minimum volume ellipsoid that contains the half ellipsoid

$$\{z \mid (z-x)^T P^{-1}(z-x) \leq 1,\ g^T(z-x) \leq 0\}$$

is given by

$$\mathcal{E}^+ = \{z \mid (z-x^+)^T (P^+)^{-1}(z-x^+) \leq 1\},$$

where

$$\begin{aligned} x^+ &= x - \frac{1}{n+1} P\tilde{g}, & (1) \\ P^+ &= \frac{n^2}{n^2-1}\left(P - \frac{2}{n+1} P\tilde{g}\tilde{g}^T P\right), & (2) \end{aligned}$$

and

$$\tilde{g} = \frac{1}{\sqrt{g^T P g}} g$$

is a normalized subgradient.

Let's give some interpretations of these equations. The step given in (1) is in the direction of the negative subgradient, in the coordinates given by the matrix P. The step $P\tilde{g}$ is the step to the boundary of $\mathcal{E}^+$, so (1) is a step that is a fraction $1/(n+1)$ to the boundary. From (2), we can think of $\mathcal{E}^{(k+1)}$ as slightly thinner than $\mathcal{E}^{(k)}$ in the direction $g^{(k)}$, and slightly enlarged over all.

Basic ellipsoid method. The basic ellipsoid algorithm is:

Ellipsoid method

```
given an initial ellipsoid $(P^{(0)}, x^{(0)})$ containing a minimizer of $f$.
$k := 0$.
repeat
  Compute a subgradient: $g^{(k)} \in \partial f(x^{(k)})$.
  Normalize the subgradient: $\tilde{g} := \frac{1}{\sqrt{g^{(k)T} P^{(k)} g^{(k)}}} g^{(k)}$.
```

```
    Update ellipsoid center: $x^{(k+1)} := x^{(k)} - \frac{1}{n+1} P^{(k)} \tilde{g}$.
    Update ellipsoid shape: $P^{(k+1)} := \frac{n^2}{n^2-1}\left(P^{(k)} - \frac{2}{n+1} P^{(k)} \tilde{g} \tilde{g}^T P^{(k)}\right)$.
    $k := k + 1$.
```

The ellipsoid method is not a descent method, so we keep track of the best point found. We define

$$f_{\rm best}^{(k)} = \min_{j=0,\ldots,k} f(x^{(j)}).$$

Proof of convergence. In this section we show that the ellipsoid algorithm converges, *i.e.*,

$$\lim_{k\to\infty} f_{\rm best}^{(k)} = f^\star,$$

provided our original ellipsoid $\mathcal{E}^{(0)}$, which we take to be a ball, *i.e.*, $x^{(0)} = 0$, $P^{(0)} = R^2 I$, contains a minimizing point in its interior. Even though $\mathcal{E}^{(k+1)}$ can be *larger* than $\mathcal{E}^{(k)}$ in the sense of maximum semi-axis ($\lambda_{\max}(P^{(k+1)}) > \lambda_{\max}(P^{(k)})$ is possible), it turns out that its volume is less:

$$\begin{array}{rcl} \mathbf{vol}(\mathcal{E}^{(k+1)}) & = & \left(\dfrac{n}{n+1}\right)^{(n+1)/2} \left(\dfrac{n}{n-1}\right)^{(n-1)/2} \mathbf{vol}(\mathcal{E}^{(k)}) \\ & < & e^{-1/2n} \mathbf{vol}(\mathcal{E}^{(k)}), \end{array}$$

by a factor that only depends on the dimension n. The first line is simple calculation from the formula for updating the matrix P that describes the ellipsoid (2) and basic determinant identities.

We suppose that $x^\star \in \mathcal{E}^{(0)}$ and that $f_{\rm best}^{(K)} \geq f^\star + \epsilon$, where $\epsilon > 0$. This means that for $k = 1, \ldots, K$, $f(x^{(k)}) > f^\star + \epsilon$. Then every point z excluded in iterations $0, \ldots, K$ has $f(z) \geq f^\star + \epsilon$, since at iteration k the function values in each excluded half-space are at least $f(x^{(k)})$.

We define

$$G = \max_{g \in \partial f(x),\ \ x \in \mathcal{E}^{(0)}} \|g\|,$$

as the maximum length of the subgradients over the initial ellipsoid. (In fact, G is a Lipschitz constant for f over the initial ellipsoid.)

Then in the ball

$$\mathcal{B} = \{z \mid \|z - x^\star\| \leq \epsilon/G\}.$$

we have $f(z) \leq f^\star + \epsilon$. We assume without loss of generality that $\mathcal{B} \subseteq \mathcal{E}^{(0)}$, and consequently no point of $\mathcal{B}$ was excluded in iterations $1, \ldots, K$, so we have

$$\mathcal{B} \subseteq \mathcal{E}^{(k)}.$$

Thus, $\mathbf{vol}(\mathcal{E}^{(k)}) \geq \mathbf{vol}(\mathcal{B})$, so

$$e^{-K/2n} \mathbf{vol}(\mathcal{E}^{(0)}) \geq (\epsilon/G)^n \beta_n.$$

For $\mathcal{E}^{(0)} = \{z \mid \|z\| \leq R\}$ we have $\mathbf{vol}(\mathcal{E}^{(0)}) = R^n \beta_n$, so, taking logs,

$$-\frac{K}{2n} + n \log R \geq n \log \frac{\epsilon}{G},$$

and therefore

$$K \leq 2n^2 \log \frac{RG}{\epsilon}.$$

Thus to compute $f^\star$ with error at most ϵ, it takes no more than $2n^2 \log RG/\epsilon$ iterations of the ellipsoid algorithm; this number grows slowly with both dimension n and accuracy ϵ (at least, from the point of view of complexity theory).

Interpretation. From the initial ellipsoid (a ball of radius R) and the Lipschitz bound on f (*i.e.*, G), we know that our original point is no more than RG suboptimal. After K steps we have produced a point that is ϵ suboptimal. Thus the ratio RG/ϵ is the fractional reduction in our ignorance about the number $f^\star$. The complexity analysis above says that the number of steps required is proportional to the log of this ratio. This is *exactly* like the bisection method in $\mathbf{R}$ (and indeed, it is the bisection method for $n = 1$). We might say that the ellipsoid method is a generalization of the bisection method to higher dimensions.

Stopping criterion. Since we always know that there is a minimizer $x^\star \in \mathcal{E}^{(k)}$, we have

$$f^\star = f(x^\star) \geq f(x^{(k)}) + g^{(k)T}(x^\star - x^{(k)})$$

for some $x^\star \in \mathcal{E}^{(k)}$, and hence

$$\begin{array}{rcl} f(x^{(k)}) - f^\star & \leq & -g^{(k)T}(x^\star - x^{(k)}) \\ & \leq & \max_{z \in \mathcal{E}^{(k)}} -g^{(k)T}(z - x^{(k)}) \\ & = & \sqrt{g^{(k)T} P^{(k)} g^{(k)}}. \end{array}$$

Thus the simple stopping criterion

$$\sqrt{g^{(k)T} P^{(k)} g^{(k)}} \leq \epsilon$$

guarantees that on exit, $f(x^{(k)})$ is within ϵ of $f^\star$. A more sophisticated stopping criterion is

$$u^{(k)} - l^{(k)} \leq \epsilon,$$

where

$$u^{(k)} = \min_{1 \leq i \leq k} f(x^{(i)}), \qquad l^{(k)} = \max_{1 \leq i \leq k} \left(f(x^{(i)}) - \sqrt{g^{(i)T} P^{(i)} g^{(i)}} \right).$$

While the ellipsoid algorithm works for quasiconvex functions (with the $g^{(k)}$'s quasigradients), this stopping criterion does not.

Ellipsoid update — Hessian form. We can express the ellipsoid algorithm in terms of $H = P^{-1}$ instead of P. The update equations become (using basic linear algebra)

$$\begin{aligned} x^+ &= x - \frac{1}{n+1} H^{-1} \tilde{g}, \\ P^+ &= \left(1 - \frac{1}{n^2}\right)\left(H + \frac{2}{n-1} \tilde{g}\tilde{g}^T\right), \end{aligned}$$

where

$$\tilde{g} = \frac{1}{\sqrt{g^T H^{-1} g}} g$$

is the normalized subgradient.

In this form we can interpret H as an approximation of the 'Hessian' of f, and the ellipsoid algorithm then looks like a quasi-Newton method. But remember that f need not even be differentiable, let alone have a second derivative, *i.e.*, a Hessian.

2 Deep-cut ellipsoid algorithm

If an upper bound on the objective function is available at each iteration, *deep cuts* can be used to improve performance over the basic ellipsoid algorithm. Suppose that at iteration k, we have an upper bound on the optimal objective value: $f^\star \leq f_{\rm best}^{(k)} \leq f(x^{(k)})$. (One such bound is the best point found so far, since the iterates $x^{(k)}$ do not in general produce monotonically decreasing objective values). Given $g \in \partial f(x)$, we have for all z

$$f(z) \geq f(x) + g^T(z - x),$$

so for every optimizer $x^\star$, we have

$$g^T(x^\star - x) + f(x) \leq f^\star \leq f_{\rm best}^{(k)}$$

We can therefore exclude from consideration the half-space

$$\{z \mid g^T(z - x) > f_{\rm best}^{(k)} - f(x)\},$$

which is bigger than the half-space $\{z \mid g^T(z-x) > 0\}$ excluded in the ellipsoid algorithm, provided $f(x^{(k)}) > f_{\rm best}^{(k)}$, as shown in figure 2.

In the deep-cut method, we choose $\mathcal{E}^{(k+1)}$ to be the minimum volume ellipsoid that contains the set

$$\mathcal{S}^{(k)} = \mathcal{E}^{(k)} \cap \{z \mid g^{(k)T}(z - x^{(k)}) \leq f_{\rm best}^{(k)} - f(x^{(k)})\}.$$

Then $\mathcal{E}^{(k+1)}$ is given by

$$\begin{aligned} x^{(k+1)} &= x^{(k)} - \frac{1 + n\alpha}{n+1} P^{(k)} \tilde{g}^{(k)}, \\ P^{(k+1)} &= \frac{n^2(1-\alpha^2)}{n^2 - 1}\left(P^{(k)} - \frac{2(1+n\alpha)}{(n+1)(1+\alpha)} P^{(k)} \tilde{g}^{(k)} \tilde{g}^{(k)T} P^{(k)}\right), \end{aligned}$$

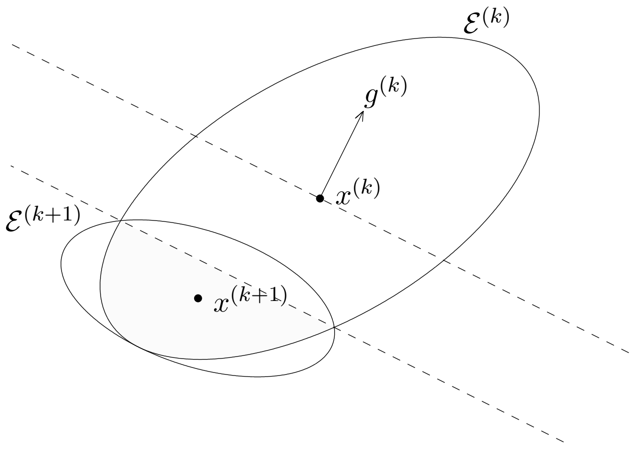

Figure 2: The shaded sliced ellipsoid, which is known to cotnain a minimizer, is enclosed by the ellipsoid of smallest volume, denoted $\mathcal{E}^{(k+1)}$, and centered at $x^{(k+1)}$.

where

$$\begin{aligned} \alpha &= \frac{f(x^{(k)})}{\sqrt{g^{(k)T}P^{(k)}g^{(k)}}}, \\ \tilde{g}^{(k)} &= \frac{1}{\sqrt{g^{(k)T}P^{(k)}g^{(k)}}}g^{(k)}. \end{aligned}$$

The deep-cut ellipsoid algorithm is thus:

```
*Deep-cut ellipsoid method*

**given** an initial ellipsoid $(P^{(0)}, x^{(0)})$ containing a minimizer of $f$, and a sequence
of upper bounds $f_{\text{best}}^{(k)}$.
$k := 0$.
**repeat**
   Compute a subgradient $g^{(k)} \in \partial f(x^{(k)})$.
   Update upper bound $f_{\text{best}}^{(k)}$.
   $\alpha := \frac{f(x^{(k)})-f_{\text{best}}^{(k)}}{\sqrt{g^{(k)T}P^{(k)}g^{(k)}}}$.
   Normalize the subgradient: $\tilde{g}^{(k)} := \frac{1}{\sqrt{g^{(k)T}P^{(k)}g^{(k)}}}g^{(k)}$.
   Update ellipsoid center: $x^{(k+1)} := x^{(k)} - \frac{1+n\alpha}{n+1}P^{(k)}\tilde{g}^{(k)}$.
   Update ellipsoid shape: $P^{(k+1)} := \frac{n^2}{n^2-1}(1-\alpha^2)\left(P^{(k)} - \frac{2(1+n\alpha)}{(n+1)(1+\alpha)}P^{(k)}\tilde{g}^{(k)}(\tilde{g}^{(k)})^T P^{(k)}\right)$.
   $k := k + 1$.
```

3 Ellipsoid algorithm with constraints

The (deep-cut) ellipsoid algorithm is readily modified to solve the constrained problem

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \le 0, \quad i = 1, \dots, m. \end{array}$$

In this section we describe one such modification.

Once again, we generate a sequence of ellipsoids of decreasing volume, each of which is guaranteed to contain a feasible minimizer. If $x^{(k)}$ is feasible ($f_i(x^{(k)}) \le 0$ for $i = 1, \dots, m$) then we form $\mathcal{E}^{(k+1)}$ exactly as in the (deep-cut) ellipsoid algorithm; we call this an *objective* iteration. If $x^{(k)}$ is infeasible ($f_i(x^{(k)}) > 0$ for some i) then we form $\mathcal{E}^{(k+1)}$ as in the ellipsoid algorithm, but using a subgradient of a violated constraint function f_i instead of the objective. We call this a *constraint* iteration.

The algorithm is thus:

```
*Ellipsoid method with constraints*

**given** an initial ellipsoid $(P^{(0)}, x^{(0)})$ containing feasible minimizers.
$k := 0$.
**repeat**
  If $f_i(x^{(k)}) > 0$ for some $i$,        // $x^{(k)}$ *is infeasible.*
    Compute a subgradient $g^{(k)} \in \partial f_i(x^{(k)})$.
    Normalize the subgradient: $\tilde{g}^{(k)} := \frac{1}{\sqrt{g^{(k)T} P^{(k)} g^{(k)}}} g^{(k)}$.
    $\alpha := \frac{f_i(x^{(k)})}{\sqrt{g^{(k)T} P^{(k)} g^{(k)}}}$.
  Else,        // $x^{(k)}$ *is feasible.*
    Compute a subgradient $g^{(k)} \in \partial f(x^{(k)})$.
    Normalize the subgradient: $\tilde{g}^{(k)} := \frac{1}{\sqrt{g^{(k)T} P^{(k)} g^{(k)}}} g^{(k)}$.
    Update upper bound $f_{\text{best}}^{(k)}$.
    $\alpha := \frac{f(x^{(k)}) - f_{\text{best}}^{(k)}}{\sqrt{g^{(k)T} P^{(k)} g^{(k)}}}$.
  Update ellipsoid center: $x^{(k+1)} := x^{(k)} - \frac{1+n\alpha}{n+1} P^{(k)} \tilde{g}^{(k)}$.
  Update ellipsoid shape: $P^{(k+1)} := \frac{n^2}{n^2-1}(1-\alpha^2)\left(P^{(k)} - \frac{2(1+n\alpha)}{(n+1)(1+\alpha)} P^{(k)} \tilde{g}^{(k)} (\tilde{g}^{(k)})^T P^{(k)}\right)$.
  $k := k + 1$.
```

In a constraint iteration, all points we discard are infeasible. In an objective iteration, all points we discard have objective value greater than or equal to the current, feasible point. Thus in each case, we do not discard any minimizers, so that the ellipsoids will always contain any minimizers that are in the initial ellipsoid.

The same proof as for the basic ellipsoid algorithm shows that this modified algorithm works provided the set of feasible ϵ-suboptimal points has positive volume. (This can be related to Slater's condition, assuming the constraint functions are Lipschitz.)

Stopping criterion. For an objective iteration, we can use the stopping criterion

$$\sqrt{g^{(k)T}P^{(k)}g^{(k)}} \leq \epsilon, \quad f_i(x^{(k)}) \leq 0.$$

For a constraint iteration with violated constraint i, we terminate the algorithm if all points in the new ellipsoid are known to violate constraint i:

$$f_i(x^{(k)}) > \sqrt{g^{(k)T}P^{(k)}g^{(k)}}.$$

Alternatively, if we allow slightly violated constraints, we can use the combined stopping criterion

$$\sqrt{g^{(k)T}P^{(k)}g^{(k)}} \leq \epsilon_{\text{obj}}, \quad f(x^{(k)}) \leq \epsilon_{\text{feas}}.$$

With this stopping criterion, the modified algorithm will work even when the set of feasible points with nearly optimal objective value does not have positive volume, *e.g.*, with equality constraints.

4 Numerical examples

Unconstrained example. We consider the problem of minimizing a piecewise affine function:

$$\text{minimize} \quad f(x) = \max_{i=1,\ldots,m}(a_i^T x + b_i), \tag{3}$$

with variable $x \in \mathbf{R}^n$. We use $n = 20$ variables and $m = 100$ terms, with problem data a_i and b_i generated from a unit normal distribution.

Basic ellipsoid algorithm. We use the basic ellipsoid algorithm described in §1, starting with $P^{(0)} = I$, and $x^{(0)} = 0$.

Figure 3 shows the convergence of $f^{(k)}$ to the optimal value, as well as $f^{(k)} - \sqrt{g^{(k)T}P^{(k)}g^{(k)}}$ (which is a lower bound on $f^\star$). Figure 4 shows the convergence of $f_{\text{best}}^{(k)} - f^\star$, where $f_{\text{best}}^{(k)} = \min_{i=1,\ldots,k} f(x^{(i)})$ is the best point found at iteration k. The slow convergence shown in this example is in fact typical for this method.

Figure 5 compares the convergence of the basic ellipsoid method and the deep-cut ellipsoid method. You might guess that the deep-cut method would perform better than the basic ellipsoid method, since it cuts more volume out of each ellipsoid. But in most cases it does not converge much faster.

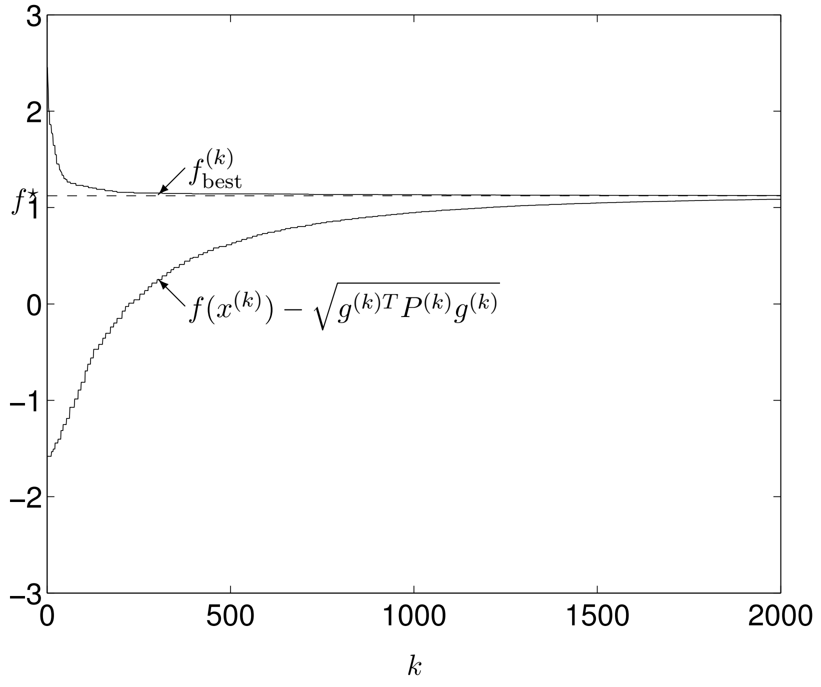

Figure 3: Convergence of $f_{\text{best}}^{(k)}$ and $f(x^{(k)}) - \sqrt{g^{(k)T}P^{(k)}g^{(k)}}$ (a lower bound on $f^\star$), to $f^\star$ with the iteration number k.

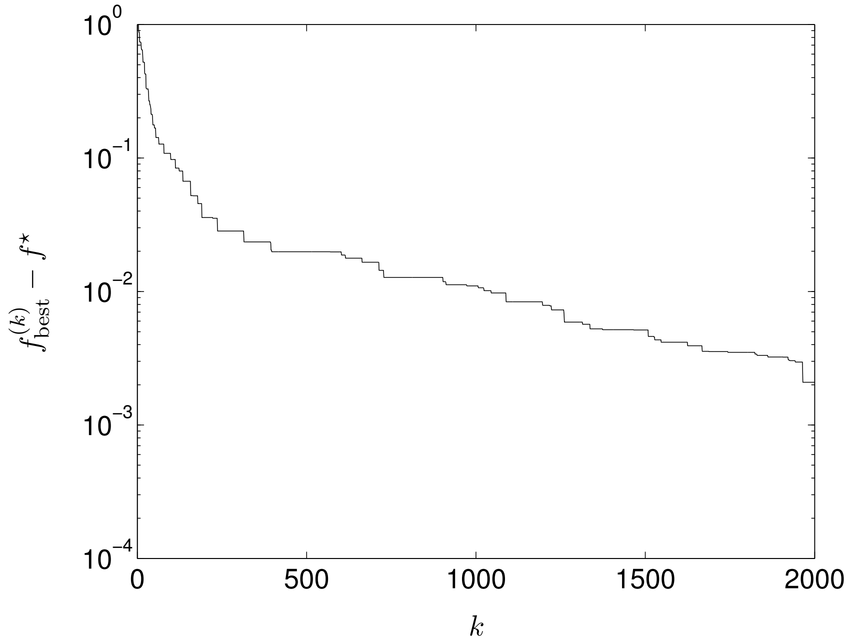

Figure 4: The suboptimality gap $f_{\text{best}}^{(k)} - f^\star$ versus iteration k.

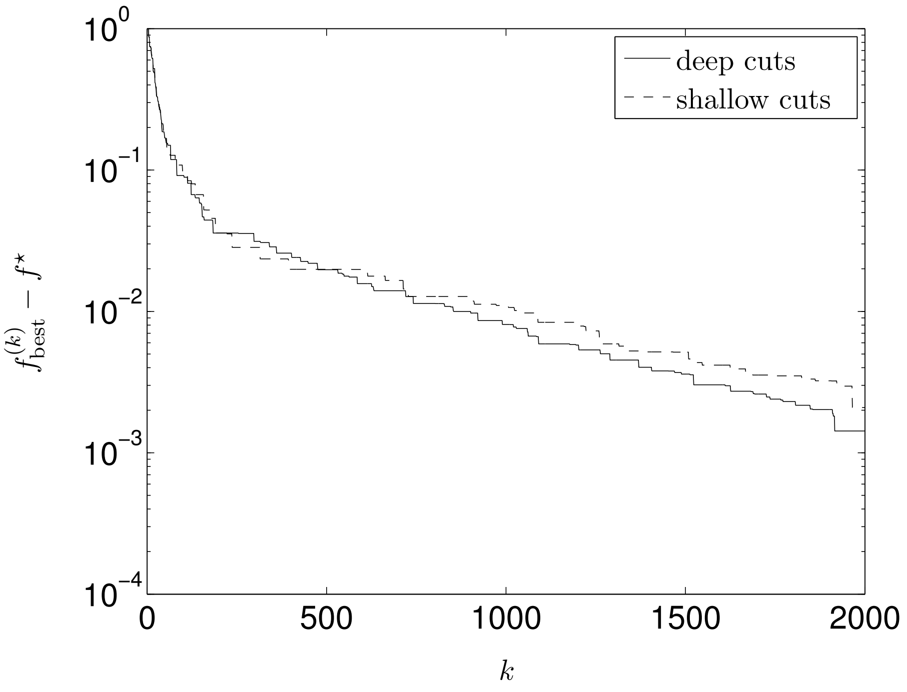

Figure 5: The suboptimality gap $f_{\text{best}}^{(k)} - f^\star$ versus iteration k for the basic ellipsoid algorithm and the deep-cut ellipsoid algorithm.

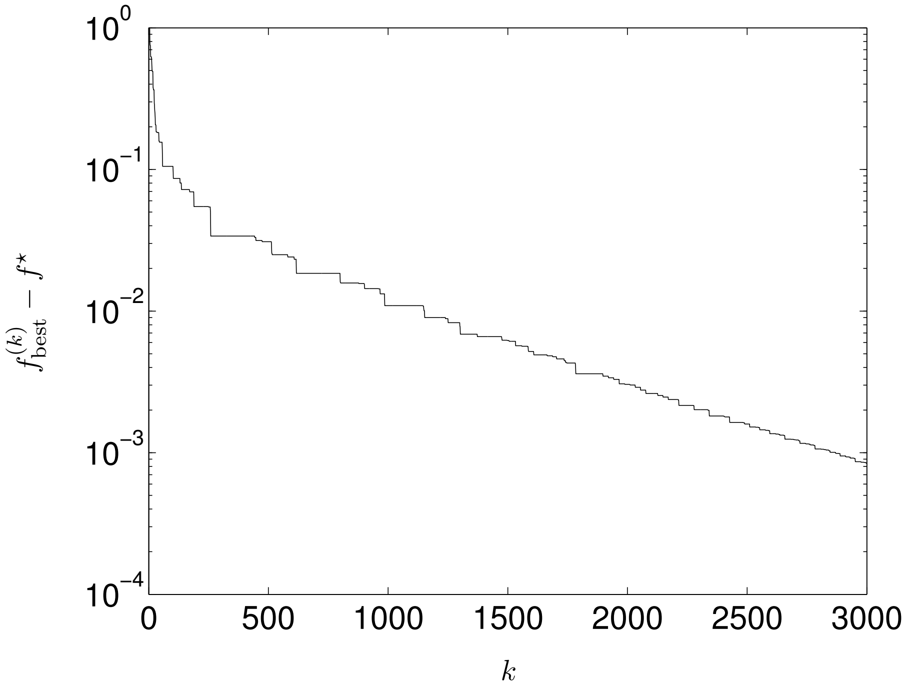

Figure 6: The suboptimality gap $f_{\rm best}^{(k)} - f^\star$ versus iteration k for the constrained ellipsoid algorithm.

Constrained example. Here we address a box-constrained version of (3):

$$\begin{array}{ll} \text{minimize} & f(x) = \max_{i=1,\dots,m}(a_i^T x + b_i), \\ \text{subject to} & |x_i| \leq 0.1, \end{array}$$

with problem data generated as in the first example. We use the constrained ellipsoid method the solve the problem. Figure 6 shows the convergence of $f_{\rm best}^{(k)} - f^\star$.

5 Ellipsoid method for other problems

The ellipsoid and deep-cut ellipsoid methods only require the ability to find a neutral or deep cutting-plane at any given point. This problem comes up in several contexts beyond minimizing a convex function. We give two examples in this section.

Fixed-point of nonexpansive operator. Suppose that $F : \mathbf{R}^n \to \mathbf{R}^n$ is nonexpansive, *i.e.*, satisfies

$$\|F(x) - F(y)\|_2 \leq \|x - y\|_2$$

for all $x, y \in \mathbf{R}^n$. We seek a point $x^\star$ that is a fixed point of F, *i.e.*, $F(x^\star) = x^\star$. Assuming such a point exists, we can find it using the ellipsoid method.

We observe that

$$\|F(x) - x^\star\|_2^2 = \|F(x) - F(x^\star)\|_2^2 \leq \|x - x^\star\|_2^2,$$

which we expand as

$$\|F(x)\|_2^2 - 2F(x)^T x^\star + \|x^\star\|_2^2 \leq \|x\|_2^2 - 2x^T x^\star + \|x^\star\|_2^2,$$

which we rewrite as

$$2(x - F(x))^T x^\star \leq \|x\|_2^2 - \|F(x)\|_2^2,$$

and finally,

$$(x - F(x))^T (x^\star - x) + (1/2)\|x - F(x)\|_2^2 \leq 0.$$

But this is a deep cut: it says that any fixed point of F must satisfy

$$g^T (x^\star - x) + h \leq 0,$$

where

$$g = x - F(x), \quad h = (1/2)\|x - F(x)\|_2^2 > 0.$$

So we can directly use the deep-cut ellipsoid method to find a fixed point of F (if it exists).

Zero of monotone operator. Suppose that $T : \mathbf{R}^n \to \mathbf{R}^n$ is a monotone operator, *i.e.*, it satisfies

$$(T(x) - T(y))^T (x - y) \geq 0$$

for all $x, y \in \mathbf{R}^n$. We seek a zero of T, *i.e.*, a point $x^\star$ with $T(x^\star) = 0$ (assuming such a point exists). Here too we can generate a cutting-plane at a point x, and then we can use the ellipsoid method.

To do this, we start with

$$0 \leq (T(x^\star) - T(x))^T (x^\star - x) = -T(x)^T (x^\star - x),$$

so any zero of F satisfies

$$g^T (x^\star - x) \leq 0,$$

where $g = T(x)$. This is a neutral cutting-plane.

Let's apply this to the monotone operator

$$T(x, y) = \begin{bmatrix} \nabla_x L(x, y) \\ -\nabla_y L(x, y) \end{bmatrix} = \begin{bmatrix} \nabla f(x) + A^T y \\ b - Ax \end{bmatrix},$$

where $L(x, y) = f(x) + y^T (Ax - b)$ is the Lagrangian of the problem of minimizing $f(x)$ subject to $Ax = b$. (We can use any subgradient in place of $\nabla f(x)$ here when f is not differentiable.) Any zero of T is a primal-dual solution of the constrained problem.

At the point (x^k, y^k), we will use the condition

$$\begin{bmatrix} \nabla f(x) + A^T y \\ b - Ax \end{bmatrix}^T \begin{bmatrix} x^\star - x \\ y^\star - y \end{bmatrix} \leq 0$$

in the ellipsoid method, where $(x^\star, y^\star)$ is any primal-dual solution.

A Notes and references

Two early articles that give algorithms for minimizing a quasiconvex function using cutting-planes are Newman [?] and Levin [?]. In these precursors of the ellipsoid algorithm, the localization set is a polyhedron, and its complexity (number of inequalities) increases as the algorithm proceeds, so that the computation per iteration grows.

The ellipsoid algorithm was developed in the 1970's in the Soviet Union by Shor, Yudin, and Nemirovsky. A detailed history of its development, including English and Russian references, appears in chapter 3 of Akgül [?]. It was used in 1979 by Khachiyan in his famous proof that linear programs can be solved in polynomial time; an English translation appears in [?] (see also Gács and Lovázs [?]).

The 1981 survey by Bland, Goldfarb, and Todd [?] contains a very clear description of the method and has extensive references on its development and early history, concentrating on its application to linear programming. Our exposition follows Chapter 3 of the book by Grötschel, Lovász, and Schrijver [?]. In [?, ?], Goffin gives an interpretation of the ellipsoid algorithm as a variable-metric descent algorithm.