# The Ellipsoid Method

Orestis Plevrakis

## 1 From feasibility to optimization

The "Ellipsoid method" was introduced by N. Shor in early 1970's as an iterative method for general convex optimization. We will study it in the context of linear programming:

**Problem 1: Linear Programming**

- Input: $A \in \mathbb{Q}^{m \times n}$, $b \in \mathbb{Q}^m$, $c \in \mathbb{Q}^n$.
- Output: A solution to the problem $\min\{c^\top x : Ax \geq b\}$.

Solution here means an optimal solution if there exists one. If the problem is infeasible or unbounded, then we should output corresponding messages.

At first sight, the following problem might seem strictly easier than the previous.

**Problem 2: Feasibility**

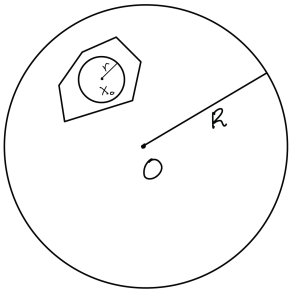
- Input: $A \in \mathbb{Q}^{m \times n}$, $b \in \mathbb{Q}^m$.
- Output: "Empty", if the polyhedron $P := \{x : \ Ax \geq b\} = \emptyset$. Else, output an $x \in P$.

It turns out that these problems are equivalent. Clearly an algorithm for Problem 1 can be directly applied to Problem 2. What about the inverse? Suppose we have a polynomial time algorithm $\mathcal{A}$ for Problem 2, and we are given an LP: $A \in \mathbb{Q}^{m \times n}$, $b \in \mathbb{Q}^m$, $c \in \mathbb{Q}^n$. First, we can use $\mathcal{A}$ to check if the LP is feasible. Now, suppose the cost is bounded, i.e., there exists an $M > 0$, such that $c^\top x \in [-M, M]$, for all $x \in P$. Suppose also that this $M$ is known to us. Then, we can use it to do binary-search, since for every $l \in [-M, M]$ we can employ $\mathcal{A}$ to check if there exists an $x$ such that $Ax \geq b$ and $c^\top x \leq l$, and thus after $\log_2(2M/\epsilon)$ calls we will have an $\hat{x} \in P$ with cost $c^\top \hat{x} \leq c^\top x_* + \epsilon$, where $x_*$ is an optimal point. To have a complete reduction, we need to answer the following: how do we get an *optimal* solution, rather than an approximately-optimal one? How do we drop the boundness assumption? These are essentially details, and at the end of the lecture, I will comment on how we can deal with them.

The following is the most important special case of Problem 2:

**Problem 3: Feasibility (with a promise)**

- Input: $A \in \mathbb{Q}^{m \times n}$, $b \in \mathbb{Q}^m$, and two positive $r, R \in \mathbb{Q}$.
- Promise: the polyhedron $P = \{x : \ Ax \geq b\}$ is either empty, or $\overline{B(x_0, r)} \subseteq P \subseteq \overline{B(0, R)}$, for some $x_0 \in \mathbb{R}^n$.
- Output: "Empty", if $P = \emptyset$. Else, output an $x \in P$.




Problem 3 is the same as Problem 2, with the exception that it requires a nonempty $P$ to be full-dimensional, i.e., it must have non-zero volume. To tackle Problem 3, it suffices to consider

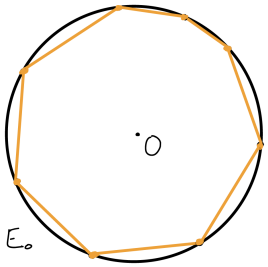
**Problem 4: Finding a feasible point (with a promise)**

- Input: $A \in \mathbb{Q}^{m \times n}$, $b \in \mathbb{Q}^m$, and two positive $r, R \in \mathbb{Q}$.
- Promise: the polyhedron $P$ is nonempty and $\overline{B(x_0, r)} \subseteq P \subseteq \overline{B(0, R)}$, for some $x_0 \in \mathbb{R}^n$.
- Output: an $x \in P$.

To see why it suffices to solve Problem 4, suppose we have a polynomial-time algorithm for it, that runs in time at most $p(\langle I \rangle)$, where $p$ is a polynomial, $I$ is the instance $(A, b, r, R)$ in binary and $\langle I \rangle$ is the length of the instance. Now, we can apply the algorithm to Problem 3. If it takes more than $p(\langle I \rangle)$ steps to complete, we stop it and output "Empty". If it terminates on time, but its output does not satisfy the inequalities $Ax \geq b$, we return "Empty". Else, the algorithm finds an $x \in P$ in at most $p(\langle I \rangle)$ steps.

## 2 The Ellipsoid Method

The Ellipsoid method solves Problem 4. The idea is to iteratively construct ellipsoids $E_0 := \overline{B(0, R)}$, $E_1, E_2, \ldots$ such that for all $i$, $E_i \supseteq P$ and $vol(E_{i+1}) < vol(E_i)$. We need to say 1) how we will construct the ellipsoids and 2) how we will eventually find an $x \in P$. Let's consider first an easy case: suppose we know that $vol(E_0) \approx vol(P)$, like in the figure:




How can find a point a point in $P$? Choose 0! If $0 \notin P$, then it violates some inequality: $A_i \cdot 0 < b_i \leq A_i x$, for all $x \in P$ ($A_i$ is the $i^{th}$ row of $A$). Thus, all of $P$ is above the hyperplane passing through 0 and having $A_i$ as orthogonal vector (Figure 1a).

Since $P$ lies inside a hemisphere, its volume is at most half the volume of $E_0$, contradiction. Even though this case was easy, it taught us something crucial: if we check whether $0 \in P$, then we either solve the problem, or we halve our search-space. Now, for $E_1$, we choose an ellipsoid that contains this hemisphere, and has volume smaller than $vol(E_0)$ (Figure 1b). In a moment, we will show how to construct $E_1$. For $E_2$ we follow the same process: we check if the center $c_1$ of $E_1$ lies in $P$, by checking the linear inequalities. If it does we are done, otherwise there is a violated constraint $A_j \cdot c_1 < b_j \leq A_j \cdot x$, for all $x \in P$, and so for $E_2$ we should choose an ellipsoid that contains the half-ellipsoid $E_1 \cap \{x : \ A_j \cdot x \geq A_j \cdot c_1\}$, and has $vol(E_2) < vol(E_1)$.

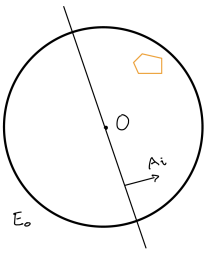
The heart of the ellipsoid method lies in Theorem 1 that follows. We will need some notation: for a point $c \in \mathbb{R}^n$ and a PD matrix $Q$, we denote by $E(c, Q)$ the ellipsoid $\{x : \ (x-c)^\top Q^{-1}(x-c) \leq 1\}$

**Theorem 1.** *Let $E(c, Q)$ be an ellipsoid, and let $a \in \mathbb{R}^n$ be a nonzero vector. Then, the ellipsoid $E(c', Q')$, with*

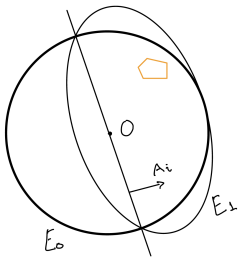
$$c' := c + \frac{1}{n+1} \frac{Qa}{\sqrt{a^\top Qa}}$$

$$Q' := \frac{n^2}{n^2-1} \left( Q - \frac{2}{n+1} \frac{Qaa^\top Q}{a^\top Qa} \right)$$

*satisfies $E(c', Q') \supseteq E(c, Q) \cap \{x : \ a^\top x \geq a^\top c\}$, and $vol(E(c', Q')) \leq e^{-\frac{1}{2(n+1)}} vol(E(c, Q))$.*




(a) $P$ belongs to the hemisphere determined by $A_i$.




(b) The new ellipsoid $E_1$ contains the hemisphere.

Figure 1: Choosing $E_1$.

This theorem immediately gives us an algorithm:

**Algorithm 1** Ellipsoid Method

$c_0 \leftarrow 0$
$Q_0 \leftarrow R^2 I$
**for** $k = 0, 1, 2, \ldots$ **do**
  **if** $c_k \in P$ **then**
    output $c_k$
  **else**
    Let $i$ be such that $A_i \cdot c_k < b_i$.
    $c_{k+1} \leftarrow c_k + \frac{1}{n+1} \frac{Q_k A_i}{\sqrt{A_i^\top Q_k A_i}}$
    $Q_{k+1} \leftarrow \frac{n^2}{n^2-1} \left( Q_k - \frac{2}{n+1} \frac{Q_k A_i A_i^\top Q_k}{A_i^\top Q_k A_i} \right)$
  **end if**
**end for**

## 2.1 Running time

How many iterations will be executed? Since $E(c_k, Q_k) \supseteq P$ for all $k$, we have $vol(E(c_k, Q_k)) \geq vol(P) \geq vol(\overline{B(x_0, r)})$. At the same time, $vol(E(c_k, Q_k)) \leq e^{-\frac{k}{2(n+1)}} vol(\overline{B(0, R)})$. Thus,

$$e^{\frac{k}{2(n+1)}} \leq \frac{vol(\overline{B(0,R)})}{vol(\overline{B(x_0,r)})} = \frac{R^n}{r^n}$$

where the last equality comes from the fact that for any $c \in R^n, \rho > 0$, $\overline{B(c,\rho)} = \overline{B(0,\rho)} = (\rho I)(\overline{B(0,1)})$, and since $det(\rho I) = \rho^n$, we have $vol(\overline{B(c,\rho)}) = \rho^n vol(\overline{B(0,1)})$. Thus, $k = O(n^2 \log \frac{R}{r})$. This is a bound on the number of iterations. Each iteration takes $O(mn)$ time to scan the inequalities, and $O(n^2)$ time to compute $c_{k+1}, Q_{k+1}$. In total, ellipsoid method takes $O\left(\max(n,m) n^3 log \frac{R}{r}\right)$ time.

**The algorithm for Problem 3.** Note that in order to solve Problem 3, we can apply the Ellipsoid method and halt it after $O\left(n^2 \log \frac{R}{r}\right)$ iterations. If it hasn't returned a point by then, we can conclude that $P$ is empty.

The only thing left is to prove Theorem 1.

## 2.2 Proof of Theorem 1

The proof will justify, where the formula for $c', Q'$ comes from. To simplify notation, we use $B$ to denote the unit ball $E(0, I)$. Now, from Problem 7, we know that Theorem 1 holds for $c = 0, Q = I, a = e_1$. We call $c_*, Q_*$ the corresponding $c', Q'$:

$$c_* = \frac{1}{n+1} e_1$$

$$Q_* = \frac{n^2}{n^2-1} \left( I - \frac{2}{n+1} e_1 e_1^\top \right)$$

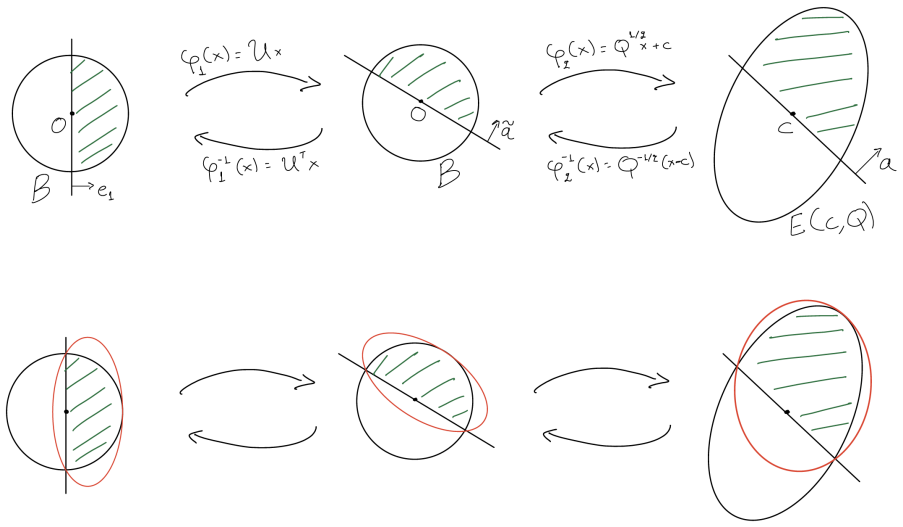



Figure 2: The reduction

Let $E_* := E(c_*, Q_*)$. We will reduce the general case to this special case, using the fact that any ellipsoid can be transformed to the unit ball by applying an affine function. The reduction is depicted in Figure 2, which we proceed explaining. First of all, $E(c,Q) = Q^{1/2}(B) + c = \phi_2(B)$, where we define $\phi_2(x) := Q^{1/2}x + c$. Clearly, $\phi_2^{-1}(x) = Q^{-1/2}(x-c)$ and $\phi_2^{-1}(E(c,Q)) = B$. How does $\phi_2^{-1}$ act on halfspaces? As we now show, any invertible affine function, maps halfspaces to halfspaces, while sending the boundary hyperplane of the input halfspace, to the boundary hyperplane of the output halfspace.

**Notation.** Let $a \in \mathbb{R}^n, a \neq 0, d \in \mathbb{R}$. We use $H(a,d)$ to denote the hyperplane $\{x : \ a^\top x = d\}$, and $H_{\geq}(a,d)$ to denote the halfspace $\{x : \ a^\top x \geq d\}$.

**Proposition 2.** *Let $a \in \mathbb{R}^n, a \neq 0, d \in \mathbb{R}$. Consider an affine function $\phi : \mathbb{R}^n \to \mathbb{R}^n$ such that $\phi(x) = Mx + c$, where $M \in \mathbb{R}^{n \times n}$ is invertible, and $c \in \mathbb{R}^n$. Let*

$$\tilde{a} := \left(M^{-1}\right)^\top a$$

$$\tilde{d} := d + a^\top M^{-1} c$$

*Then, $\phi(H(a,d)) = H(\tilde{a}, \tilde{d})$ and $\phi(H_{\geq}(a,d)) = H_{\geq}(\tilde{a}, \tilde{d})$*

*Proof.* We prove it for the hyperplane. For the halfspace the steps are identical.

$$x \in H(a,d) \iff a^\top x = d \iff a^\top M^{-1} M x = d \iff \left(\left(M^{-1}\right)^T a\right)^\top (Mx + c - c) = d$$
$$\iff \tilde{a}^\top \phi(x) = \tilde{d} \iff \phi(x) \in H(\tilde{a}, \tilde{d})$$

This implies that $\phi(H(a,d)) = H(\tilde{a}, \tilde{d})$ (why?). □

Thus, $\phi_2^{-1}(H_{\geq}(a, a^\top c)) = H_{\geq}(\tilde{a}, 0)$, where $\tilde{a} = Q^{1/2}a$. After applying $\phi_2^{-1}$ to both our ellipsoid $E(c,Q)$ and our halfspace $H_{\geq}(a, a^\top c)$, we don't get exactly the setting of the special case, since $\tilde{a}$ might not have the same direction with $e_1$. To alleviate this, we apply an orthogonal transformation:

let $U$ be an orthogonal matrix such that $Ue_1 = \tilde{a}/\|\tilde{a}\|$, and consider the affine function $\phi_1(x) := Ux$. We know that $\phi_{-1}$ maps $B$ to itself, and also by Proposition 2, $\phi_1^{-1}(H_{\geq}(\tilde{a}, 0)) = H_{\geq}(e_1, 0)$. Let $\phi := \phi_2 \circ \phi_1(E_*)$. This is an ellipsoid due to the following proposition.

**Proposition 3.** *Let $\phi_1, \phi_2 : \mathbb{R}^n \to \mathbb{R}^n$ be two affine functions. Then, their composition is affine.*

*Proof.* Let $\phi_1(x) = M_1x + c_1$, $\phi_2(x) = M_2x + c_2$. Then, $\phi_2(\phi_1(x)) = M_2(M_1x + c_1) + c_2 = M_2M_1x + (M_2c_1 + c_2)$. □

Now, we will show that $\phi(E_*)$ has the desired properties. First of all, since $E_* \supseteq B \cap H_{\geq}(e_1, 0)$,

$$\phi(E_*) \supseteq \phi\left(B \cap H_{\geq}(e_1, 0)\right) = \phi\left(B\right) \cap \phi\left(H_{\geq}(e_1, 0)\right) = E(c, Q) \cap H_{\geq}(a, a^\top c)$$

where the second-to-last equality holds because $\phi$ is invertible. Secondly, note that $\phi(x) = Q^{1/2}Ux + c$, and thus

$$\frac{vol(\phi(E_*))}{vol(E(c,Q)} = \frac{vol(\phi(E_*))}{vol(\phi(B))} = \frac{|det(Q^{1/2}U)|vol(E_*)}{|det(Q^{1/2}U)|vol(B)} = \frac{vol(E_*)}{vol(B)} \leq e^{-\frac{1}{2(n+1)}}.$$

It remains to show that $\phi(E_*) = E(c', Q')$.

$$\phi(E_*) = Q^{1/2}U(E_*) + c = Q^{1/2}U(Q_*^{1/2}(B) + c_*) + c = Q^{1/2}UQ_*^{1/2}(B) + \left(Q^{1/2}Uc_* + c\right)$$

and so the center is $c + Q^{1/2}Uc_* = c + Q^{1/2}U\frac{1}{n+1}e_1 = c + \frac{1}{n+1}\frac{Qa}{\sqrt{a^\top Qa}} = c'$. Now, for $Q'$, remember that in the previous lecture we showed that if an ellipsoid $E$ is given in the form $E = M(B)+c$, where $M$ is invertible, then we can write it in the alternative form: $E = \{x : (x-c)^\top(MM^\top)^{-1}(x-c)\}$ (this appears in the proof of Theorem 3 of lecture 6). Since

$$Q^{1/2}UQ_*^{1/2}\left(Q^{1/2}UQ_*^{1/2}\right)^\top = Q^{1/2}UQ_*U^\top Q^{1/2}$$

and

$$UQ_*U^\top = \frac{n^2}{n^2-1}\left(I - \frac{2}{n+1}Ue_1e_1^\top U^\top\right) = \frac{n^2}{n^2-1}\left(I - \frac{2}{n+1}\frac{Q^{1/2}aa^\top Q^{1/2}}{a^\top Qa}\right)$$

By multiplying with $Q^{1/2}$ from both sides, we get $Q'$. □

## 3 From Feasibility (with a promise) to solving LPs

We gave a polynomial-time algorithm for the problem "Feasibility (with a promise)". However, we did not fully show how this implies that we can solve LPs in polynomial time. There are two things that need to be established to have a full algorithm for LPs:

1. How the ellipsoid method can be adapted to solve the Feasibility problem (without any promise).
2. How we can use a poly-time algorithm for Feasibility, to solve LPs in poly-time. In the beginning of the lecture, we saw how to perform such a reduction in order to approximate within $\epsilon$ the optimal solution, under the assumption that the LP is bounded. However, it is possible to a) get an exact optimal solution (if one exists) in poly-time, and b) remove the boundness condition.

I will only sketch the main ideas for resolving 1 and 2. The complete answers contain many details, which are not so interesting conceptually.[1] For 1), the main difficulty is that the polytope $P$ can be nonempty, but contained inside some hyperplane. If we run the ellipsoid method for such a $P$, after some iterations, there will be a direction along which the ellipsoid will very flat, this direction will be almost orthogonal to the hyperplane where $P$ lies. Using this (and some other ideas), we can identify this hyperplane, and "jump into it", in the sense that we continue the ellipsoid method inside it, reducing the dimension of our problem by 1 (and this cannot happen forever). For 2), I will only comment on why it possible to get an exact optimal solution. This might seem strange, since we are doing continuous optimization and (as in the analysis of gradient descent) the natural guarantee is "after $T_\epsilon$ steps, we reach a solution with suboptimality gap at most $\epsilon$". This is reasonable, because computers have finite precision and if the optimal point has irrational coordinates, we can only hope to approximate it. But remember that for bounded polyhedra, there is always a vertex that is an optimal solution. Now, the fact that $A$ and $b$ have rational entries implies that every vertex has rational coordinates. Here is a proof: every vertex is the solution of a linear system, consisting of $n$ constraints that are set to be satisfied with equality. The coefficients of this linear system are all rationals, and the solution can be produced with Gaussian elimination, where all arithmetic operations are addition, subtraction, multiplication and division, and so the solution will be rational. Thus, in principle we can output an exact optimal solution.[2]

## References

[1] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric algorithms and combinatorial optimization*, volume 2. Springer Science & Business Media, 2012.

[1] The curious reader can find all these details here [1].

[2] Remember that in a computer we can have an exact representation of a rational number as a pair of two integers.