A note on Approximate Minimum Volume Enclosing Ellipsoid of Ellipsoids

Sachin Jambawalikar
Stony Brook University
Department of Radiology
Stony Brook, NY, 11794-8460
sjambawa@notes.cc.sunysb.edu

Piyush Kumar*
Florida State University
Department of Computer Science
Tallahassee, FL 32306-4530
piyush@cs.fsu.edu

Abstract

We study the problem of computing the Minimum Volume Enclosing Ellipsoid (MVEE) containing a given set of ellipsoids $\mathcal{S} = \{E_1, E_2, \ldots, E_n\} \subseteq \mathbb{R}^d$. We show how to efficiently compute a small set $X \subseteq \mathcal{S}$ of size at most $\alpha = |X| = \mathcal{O}(\frac{d^2}{\epsilon})$ whose minimum volume ellipsoid is an $(1+\epsilon)$-approximation to the minimum volume ellipsoid of $\mathcal{S}$. We use an augmented real number model of computation to achieve a running time of $\mathcal{O}(\alpha(nd^\omega + d^3))$ where $\omega < 2.376$ is the exponent of square matrix multiplication. This is the best known complexity for solving the MVEE problem when $n \gg d$ and ϵ is large. The algorithm is built on the previous work by Kumar and Yıldırım [17].

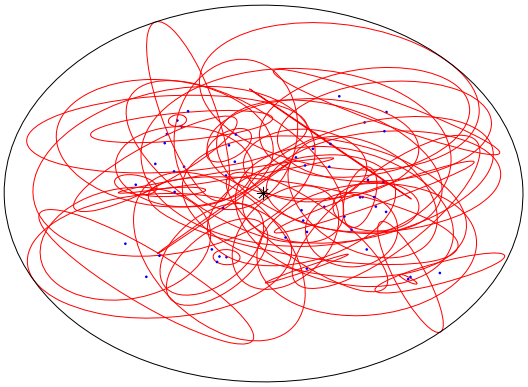


Figure 1. An approximate minimum volume ellipse containing a set of 50 randomly generated ellipses (coreset size = 4) computed using our implementation.

1 Introduction

We study the problem of computing the minimum volume enclosing ellipsoid (MVEE) of a given set of full dimensional ellipsoids $\mathcal{S} = \{E_1, \ldots, E_n\} \subseteq \mathbb{R}^d$, denoted by MVEE($\mathcal{S}$), also known as the Löwner ellipsoid for $\mathcal{S}$. Figure 1 shows an approximate minimum volume ellipse containing a set of randomly generated ellipses (Our implementation is in Matlab and took 0.4 seconds to compute the ellipse with an approximate error of $\epsilon \approx 10^{-3}$ on a Pentium IV 3.6GHz laptop).

Minimum volume enclosing ellipsoids play an important role in several diverse applications [8, 26, 30, 31, 7, 1, 18, 12, 5, 11, 25, 29, 21]. One reason that practitioners have shied away from using bounding volume ellipsoidal hierarchies is the non availability of simple algorithms to solve the enclosing ellipsoid of ellipsoids [8]. With this note, we hope to convince practitioners that a very simple algorithm could be used to compute MVEE's of ellipsoids if they can settle for approximate answers instead of exact answers.

A (full-dimensional) ellipsoid $E_{Q,c}$ in $\mathbb{R}^d$ is specified by a $d \times d$ symmetric positive definite matrix Q and a center $c \in \mathbb{R}^d$ and is defined as

$$E_{Q,c} = \{x \in \mathbb{R}^d : (x-c)^T Q(x-c) \leq 1\}. \qquad (1)$$

We denote the volume of an ellipsoid by vol $E_{Q,c}$. The volume of the ellipsoid $E_{Q,c}$ is given by [12], vol $E_{Q,c} = B \det Q^{-\frac{1}{2}}$, where B is the volume of the unit ball in $\mathbb{R}^d$.

The minimum volume enclosing ellipsoid problem for point sets has received considerable attention in the past. For applications and efficient solutions to this

*P. Kumar is partially supported by NSF CAREER Award CCF-0643593.

problem, we refer the reader to [17]. As far as we know, the problem of computing MVEE of ellipsoids has not received as much attention as the MVEE of finite point sets. It is known that the problem is a convex optimization problem with linear matrix constraints and can be solved efficiently in polynomial time [2]. We believe that a different set of fast randomized [20, 31] and deterministic [7] algorithms might also be easily adapted for the case when d is small (with running time $\mathcal{O}(d^{O(d)}n)$).

In this paper, we show an efficient way of solving this problem, which is particularly suited for applications where n is large compared to d and which can tolerate moderate values of ϵ. The algorithm extends the algorithm of Kumar and Yıldırım by using some very simple lemmas which we present in section 3. From now onwards, we refer to this algorithm as *KY* algorithm. We will use the real number model of computation unless otherwise stated [4, 24]. We augment this real number model to compute the root of a polynomial exactly in $\mathcal{O}(x)$ time when the root can be approximated in $\mathcal{O}(x \log 1/\epsilon)$, given the degree of the polynomial is x. We believe that the algorithm is practical but the error analysis complicated if we do not assume the augmented real number model. We also show a very preliminary implementation of one of our key lemmas, which is the only lemma dependent on the augmented real number model of computation (Otherwise the real number model of computation suffices). The paper is organized as follows. In section 2, we give a brief introduction to the KY algorithm [17]. Section 3 defines the modifications needed to the algorithm for it to work on ellipsoids. Section 4 presents a practical variant of our algorithm. Implementation results are presented in Section 5. We list some interesting open problems in Section 6.

2 MVEE using KY Algorithm

KY algorithm computes a $(1+\epsilon)$-approximate MVEE of a given set of affinely independent input points. One of the interesting aspects of the KY algorithm is that it involves combining two other algorithms and coming up with a practical and more efficient algorithm than either one (in at least cases where $n \gg d$ and ϵ is moderately high). To describe the algorithm we first need to review the formulation of the MVEE problem.

2.1 Formulation

For Section II, we will assume that S is a set of points instead of full dimensional ellipsoids. Throughout the paper we will assume that S is affinely independent. We will use the superscript * to denote an optimal solution. Given a set $\mathcal{S} \subseteq \mathbb{R}^d$ of n points $p_1, \ldots, p_n$, we define a "lifting" of $\mathcal{S}$ to $\mathbb{R}^{d+1}$ via

$$\begin{aligned} \mathcal{S}' &:= \{\pm q_1, \ldots, \pm q_n\}, \\ \text{where} \quad q_i &:= \begin{bmatrix} p_i \\ 1 \end{bmatrix}, \quad i = 1, \ldots, n. \end{aligned} \tag{2}$$

It follows from the results of [15] and [22] that

$$\text{MVEE}(\mathcal{S}) = \text{MVEE}(\mathcal{S}') \cap \mathcal{H}, \tag{3}$$

where

$$\mathcal{H} := \{x \in \mathbb{R}^{d+1} : x_{d+1} = 1\}. \tag{4}$$

Since $\mathcal{S}'$ is centrally symmetric, MVEE($\mathcal{S}'$) is centered at the origin. This observation gives rise to the following convex optimization problem to compute MVEE($\mathcal{S}'$), whose solution can be used to compute MVEE($\mathcal{S}$) via (3):

$$\begin{gathered} (\mathbf{P}(\mathcal{S})) \quad \min_M \quad -\log\det M \\ \text{s.t.} \quad (q^i)^T M\, q^i \le 1, \quad i = 1, \ldots, n, \\ M \in \mathbb{R}^{(d+1)\times(d+1)} \text{ is symmetric and positive definite,} \end{gathered}$$

where $M \in \mathbb{R}^{(d+1)\times(d+1)}$ is the decision variable. The Lagrangian dual of $(\mathbf{P}(\mathcal{S}))$ is equivalent to

$$\begin{aligned} (\mathbf{D}(\mathcal{S})) \quad \max_u \quad & \log\det V(u) \\ \text{s.t.} \quad & e^T u = 1, \\ & u \ge 0, \end{aligned}$$

where $u \in \mathbb{R}^n$ is the decision variable and e is the vector of ones. Let u^* denote the u at which the optimal is reached. Here $V(u) = \Sigma_{i=1}^n u_i q_i q_i^T$. Let U^* denote the matrix with diagonal entries u^*. Let P denote the $d \times n$ matrix with points $p_1, \ldots, p_n$ being the columns of the matrix. Then

$$\begin{aligned} \text{MVEE}(\mathcal{S}) &= E_{Q^*,c^*} \\ E_{Q^*,c^*} &:= \{x \in \mathbb{R}^d : (x-c^*)^T Q^* (x-c^*) \le 1\}, \end{aligned} \tag{5}$$

where

$$Q^* := \frac{1}{d}\left(PU^*P^T - Pu^*(Pu^*)^T\right)^{-1}, \quad c^* := Pu^*. \tag{6}$$

The reader is referred to [17] for details. The notion of an approximate $MVEE(\mathcal{S})$ is defined as follows. Given $\epsilon > 0$, an ellipsoid $E_{Q,c}$ is said to be a $(1+\epsilon)$-approximation to MVEE($\mathcal{S}$) if

$$E_{Q,c} \supseteq \mathcal{S}, \quad \text{vol } E_{Q,c} \leq (1+\epsilon) \text{ vol MVEE}(\mathcal{S}). \quad (7)$$

Note that this is a much more strict definition of $(1+\epsilon)$-approximation compared to some other core set results [16, 6].

2.2 The Algorithm

We are now ready to describe the KY algorithm. The KY algorithm works in two phases. The first phase which is called the initial volume approximation runs the Betke and Henk algorithm to collect a set $\mathcal{X}_0$ of $2d$ points. In the second phase these set of $2d$ points are refined by using Khachiyan's algorithm [14] to produce a $(1+\epsilon)$-approximate MVEE of $\mathcal{S}$. Below we describe both phases in more detail since we would need to modify both phases to make the algorithm work when the input is a set of full-dimensional ellipsoids.

The first phase of KY uses an algorithm given by Betke and Henk [3]. The algorithm of Betke and Henk gives a very rough approximation to the volume of the $\mathcal{CH}(\mathcal{S})$ but at the same time it produces the approximation by using at most a set of $2d$ vertices of the $\mathcal{CH}(\mathcal{S})$. The algorithm chooses a random direction b_1 in $\mathbb{R}^d$ and finds supporting hyperplanes h_1^+ and h_1^- of $\mathcal{CH}(\mathcal{S})$. $\mathcal{CH}(X)$ refers to the convex hull of X. Let p_1^α and p_1^β be the contact points of h_1^+ and h_1^- with $\mathcal{CH}(\mathcal{S})$. The next direction b_2 is chosen perpendicular to the affine span of the supporting points already computed (in this case p_1^α and p_1^β). After d steps the d pair of points define a convex hull that is contained in $\mathcal{CH}(\mathcal{S})$.

The second phase of KY uses a modification of Khachiyan's algorithm [14] (See Algorithm 1).

In the paper by Kumar and Yıldırım it was showed that if the loop 4-8 in Algorithm 1 is run $\mathcal{O}(\frac{d^2}{\epsilon})$ times, we are guaranteed a $(1+\epsilon)$ approximate MVEE(S) at termination. Let $\mathcal{X}$ denote the points p_i for which the corresponding u_i's are non-zero. Note that $|\mathcal{X}|$ can be at most $\mathcal{O}(\frac{d^2}{\epsilon})$. They also showed the following theorem.

Theorem 1 *Algorithm 1 terminates after* $\mathcal{O}(\frac{nd^3}{\epsilon})$ *operations with*

$$\begin{aligned} &\textit{vol MVEE}(\mathcal{X}) \leq \textit{vol MVEE}(\mathcal{S}) \leq \textit{vol } E_{Q^*,c^*} \\ &\leq (1+\epsilon)\textit{vol MVEE}(\mathcal{X}) \leq (1+\epsilon)\textit{vol MVEE}(\mathcal{S}), \end{aligned}$$

Algorithm 1 Khachiyan's first-order algorithm

Require: Input set of points $\mathcal{S} = \{p^1, \ldots, p^n\} \subseteq \mathbb{R}^d, \epsilon > 0, \mathcal{X}_0 \subseteq \mathcal{S}$

1: $i \leftarrow 0$
2: $u^0 \in \mathbb{R}^n$ such that $u_j^0 = \frac{1}{|\mathcal{X}_0|}$ for $p_j \in \mathcal{X}_0$ and $u_j^0 = 0$ otherwise.
3: While not converged
4: **loop**
5: $\quad j \leftarrow \arg\max_{k=1,\ldots,n}(q^k)^T V(u^i)^{-1} q^k$
6: $\quad \kappa \leftarrow \max_{k=1,\ldots,n}(q^k)^T V(u^i)^{-1} q^k.$
7: $\quad \beta \leftarrow \frac{\kappa-(d+1)}{(d+1)(\kappa-1)}$
8: $\quad u^{i+1} \leftarrow (1-\beta)u^i + \beta e_j, i \leftarrow i+1.$
9: **end loop**

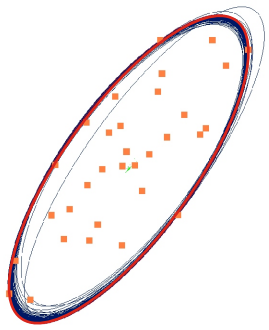


Figure 2. An approximate MVEE of points computed using KY. The blue ellipses show the iterates.

Recently, we have implemented KY in C++ with approximately 300 lines of code (but the code has not been optimized in any way). The figure 2 shows an approximate MVEE of points computed using the KY algorithm. It also draws the ellipses corresponding to V^{-1} at each iteration. The green points show the movement of the centers for each iteration. The total time taken for this toy problem was a fraction of a second (0.11 sec with $\epsilon = 0.001$ on a 3.6GHz notebook). This is work in progress and we soon plan to extend this implementation to compute MVEE of ellipsoids instead of points. Note that KY algorithm already works for a set of triangles or polytopes in $\mathbb{R}^d$ as input. It also seems that for most practical needs (for example in games), KY could be just run for a constant number of iterations instead of being driven by a particular ϵ.

3 Extension to Ellipsoids

We will need the following lemmas to extend the KY algorithm so that it works efficiently when $\mathcal{S}$ is a set of full-dimensional ellipsoids instead of points. In this section $\mathcal{S}$ will mean a set of n full-dimensional ellipsoids.

Lemma 1 (**LW48** *[13]*)

$$\frac{1}{d}MVEE(\mathcal{S}) \subseteq \mathcal{CH}(\mathcal{S}) \subseteq MVEE(\mathcal{S}) \tag{8}$$

where $\mathcal{CH}(\mathcal{S})$ denotes the convex hull of $\mathcal{S}$ and the ellipsoid on the left-hand side is obtained by scaling $MVEE(\mathcal{S})$ around its center by a factor of $1/d$.

The second lemma helps us to optimize linear objective functions over ellipsoids.

Lemma 2 (**GLS88** *[12]*) *Given a linear objective function $w^T x$ and an ellipsoid $E_{Q,c}$, let p^α and p^β be the points that maximize and minimize $w^T x$ over $E_{Q,c}$ respectively. Then*

$$p^\alpha = c + \frac{Qw}{\sqrt{w^T Q w}} \text{ and } p^\beta = c - \frac{Qw}{\sqrt{w^T Q w}}$$

Proof. It is easy to derive this lemma from the fact that $w \neq 0 \max w^T x$ such that $x \in E_{I,0}$ is achieved at vector $c + \frac{w}{||w||}$. □

Note that this lemma alone is enough to show that phase one of the KY algorithm can be run for a set of ellipsoids compared to points. The total time taken in this case is $\mathcal{O}(nd^3)$. For the complexity, we can just calculate the singular value decompositions of each of the matrices associated with the ellipsoids and store them. In each iteration, for each ellipsoid we can translate it to the origin and rotate it to make it axis aligned before we compute the extreme points p^α and p^β for that particular ellipsoid in $\mathcal{O}(d^2)$ (note that this only involves rotation of the linear objective direction). Rotating and translating back p^α and p^β to get the real optimal points takes another $\mathcal{O}(d^2)$ and hence the total complexity is given by the initial SVD computation which takes $\mathcal{O}(nd^3)$. Note that combining Betke and Henk's result which states that vol $\mathcal{CH}(\mathcal{S}) \leq d!vol\mathcal{CH}(\mathcal{X}_0)$ combined with lemma 1 can easily show that

vol MVEE$(\mathcal{X}_0) \leq$ vol MVEE$(\mathcal{S}) \leq d^{2d}$vol MVEE$(\mathcal{X}_0)$

(This was already shown in Kumar and Yıldırım [17]). The only thing remaining to convert in the KY algorithm is adapting Algorithm 1 to make it work with ellipsoids. Note that we are using the fact that ellipsoids are nothing but infinite union of points. If we can implement both phases of KY to run on these infinite point sets, the guarantees of KY will also extend to infinite point sets [16].

Before we move to our next lemma, we note that the only lines in algorithm 1 that touches the input are line 5,6. If we could implement line 5 to run on the infinite point set resulting from union of ellipsoids, KY would start accepting ellipsoids as input (since line 6 can easily be evaluated for the j-th point computed). Let us look at line 5 more closely. It finds out the furthest outlier in the ellipsoidal norm. So we need to solve the following optimization problem for each $E_{Q,c} \in \mathcal{S}$.

$$\begin{aligned} (\mathbf{FO}) \quad & \max_x \quad x^T V^{-1} x \\ & \text{s.t.} \quad (x-c)^T Q (x-c) = 1, \end{aligned}$$

where V and Q are both positive definite. Note that the furthest outlier computed in each iteration (one run of loop 4-8 in Algorithm 1) will help us run Algorithm 1 to completion. We know that the new $V(u)$ should be

$$V(u^{i+1}) = (1-\beta)V(u^i) + \beta p_j p_j^T$$

Here β refers to the value of β at the i-th iteration. Let $W = (1-\beta)V(u^i)$. Then $W^{-1} = \frac{1}{1-\beta}V(u^i)^{-1}$. Since we maintain $V(u^i)^{-1}$ from the last iteration, hence $V(u^{i+1})^{-1} = (W + \beta p_j p_j^T)^{-1} = W^{-1} - \beta W^{-1} p_j p_j^T W^{-1}$. This takes $\mathcal{O}(d^2)$ time to compute. Later we will see that the cost of solving n (**FO**) problems dominates the cost of updating $V(u^i)^{-1}$ in each iteration. Note that we now do not need to keep the vector u explicitly in KY. We can just replace line 7 with our new update rule.

We also know that Algorithm 1 terminates in $\mathcal{O}(d^2/\epsilon)$ iterations for any point set [17]. Since a set of ellipsoids is nothing but a set of points, this still remains true. So the only part of the algorithm left to extend is solving the problem (**FO**). For this we first transform the problem by multiplying the whole space with $V^{-1/2}$. This changes the problem to

$$\begin{aligned} (\mathbf{FO}') \quad & \max_x \quad x^T x \\ & \text{s.t.} \quad (x-c')^T Q' (x-c') = 1, \end{aligned}$$

This effect can be thought of as converting the ellipsoidal norm defined by V^{-1} to the euclidean norm.

Now we can translate the whole space by c' such that the constraining ellipsoid Q' becomes centered at the origin. This also changes the objective function to $\max_x (x + c')^T (x + c')$. Again we apply a rotation to the entire space so that Q' becomes axis aligned (diagonal, Please see figure 3). So the new problem we want

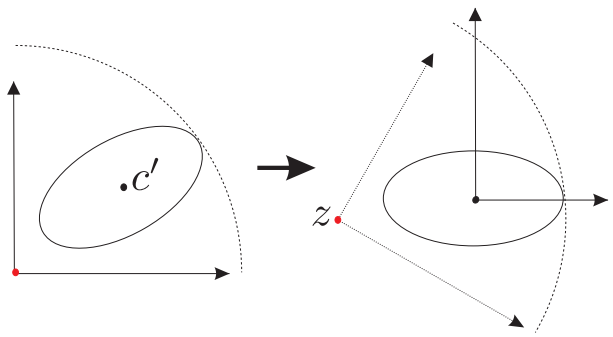


Figure 3. Change of objective function.

to solve is:

$$(\mathbf{FO''}) \quad \begin{array}{ll} \max_x & (x-z)^T(x-z) \\ \text{s.t.} & x^T D x = 1, \end{array}$$

Note that if we solve our problem in this space, it only takes $O(d^2)$ time to recover the real solution (which involves translation, rotation and scaling of the current solution to the original space). But to convert our original problem to this form, we already need $\mathcal{O}(d^\omega)$ where $\omega \leq 2.36$ is the cost of matrix multiplication of two $d \times d$ matrices.

To solve ($\mathbf{FO''}$), we first assume that $D = \{1/r_1^2, \ldots, 1/r_d^2\}$. W.l.o.g. assume that $0 < r_1 \leq r_2 < \ldots \leq r_d$. $E_{D,0}$ intersects many levels of $(x-z)^T(x-z) = l$. We want to compute the point that attains l_{max} for algorithm 1. Finding l_{max} is a constrained optimization problem that can be reduced to polynomial root finding by using Lagrange multipliers. The Lagrangian here is $L(x, \lambda) = (x-z)^T(x-z) - \lambda(x^T D x - 1)$. We now can find the maximizing x from the optimality condition

$$\nabla_x L(x, \lambda) = 2(x - z) - 2\lambda D x = 0. \qquad (12)$$

Geometrically, this condition means that the normals of the sphere and the axis-aligned ellipsoid centered at the origin are aligned at the furthest point from z on the ellipsoid. We are now ready to show the following lemma.

Lemma 3 *Let λ_1 and λ_2 be two roots satisfying the first order condition (equation 12) with corresponding solutions x and y. If $\lambda_1 > \lambda_2$ then x is at least as far as y from c.*

Proof. We want to show that

$$\sum_{i=1}^{d} \left((x_i - z_i)^2 - (y_i - z_i)^2\right)$$

$$= \sum_{i=1}^{d} \left(x_i^2 - y_i^2 + 2z_i(y_i - x_i)\right) \geq 0$$

Using 12 we now need to show,

$$\sum_{i=1}^{d} \left(x_i^2 - y_i^2 + \left(\left(x_i - x_i\lambda_1/r_i^2\right)\right.\right.$$
$$\left.\left. + \left(y_i - y_i\lambda_2/r_i^2\right)\right)(y_i - x_i)\right)$$

is ≥ 0 which simplifies to:

$$\sum_{i=1}^{d} \left(-x_i y_i \lambda_1/r_i^2 + x_i y_i \lambda_2/r_i^2\right) \geq 0$$

$$\text{or} \quad (\lambda_1 - \lambda_2)(1 - \textstyle\sum_{i=1}^{d}(x_i y_i)/r_i^2) \quad \geq 0 \quad (13)$$

Since $\lambda_1 > \lambda_2$, we only need to show that:

$$\left(\frac{1}{2}\sum_{i=1}^{d} x_i^2/r_i^2 + \frac{1}{2}\sum_{i=1}^{d} y_i^2/r_i^2 - \sum_{i=1}^{d}(x_i y_i)/r_i^2\right)$$

$$= \sum_{i=1}^{d} \left(\frac{x_i - y_i}{\sqrt{2} r_i}\right)^2 \geq 0$$

which is trivially true. □

We are now ready to prove the following theorem.

Theorem 2 *The objective function $(x-z)^T(x-z)$ can be maximized for the constraint $p \in E_{D,0}$ in time $\mathcal{O}(d^2)$*
.

Proof. Assume $z \neq 0$ since that case is trivial to solve. We prove this lemma by splitting the proof in two cases. *Case I:*

$$x = (I - \lambda D)^{-1} z$$

$$= \text{diag}\left(\frac{z_1}{1-\frac{\lambda}{r_1^2}}, \frac{z_2}{1-\frac{\lambda}{r_2^2}}, \ldots, \frac{z_d}{1-\frac{\lambda}{r_1^d}}\right). \quad (14)$$

In this case, we can plug x in $E_{D,0}$ to get:

$$\sum_{k=1}^{d} \frac{x_k^2}{r_k^2} = 1$$

$$= \sum_{k=1}^{d} \frac{1}{r_k^2}\left(\frac{z_k r_k^2}{r_k^2 - \lambda}\right)^2 = \sum_{k=1}^{d}\left(\frac{z_k^2 r_k^2}{(r_k^2-\lambda)^2}\right)$$

or the secular equation

$$\mathcal{P}(\lambda) = 1 - \sum_{k=1}^{d}\left(\frac{z_k^2 r_k^2}{(r_k^2-\lambda)^2}\right) = 0$$

$$= 1 - \sum_{k=1}^{d}\left(\frac{w_k^2}{(r_k^2-\lambda)^2}\right) \quad (15)$$

where $w_k = z_k r_k$. When $z \neq 0$, by lemma 3 we are only interested in $\lambda_{\max}$ since that is the one that maximizes $(x-z)^T(x-z)$. It's easy to notice that $\lambda_{\max} \in (r_d^2, +\infty) \subset \mathbb{R}$. Also there is only one real root in $(r_d^2, +\infty)$ since the derivative of this function is positive and hence the function is strictly increasing in $(r_d^2, +\infty)$; As $x \to \infty$, $\mathcal{P}(x) \to 1$ and as $x \to r_d$, $\mathcal{P}(x) \to -\infty$. It is also easy to show that $r_d^2 \leq \lambda_{\max} < r_d^2 + \sqrt{\sum_{k=1}^{d} w_k^2}$. The isolated root $\lambda_{\max}$ can now be found using any root finding method or the problem can be formulated as an eigenvalue problem and the largest root can be computed using QR factorization in $\mathcal{O}(d^3)$ operations [23]. With the result in [19] we can get a running time of $\mathcal{O}(d^2)$ (Note that [19] assumes that all the radii are distinct for the secular equation, luckily our equation can easily be transformed into their formulation). It was also noted in [27] that our problem is an instance of the *semidefinite programming* problem for which polynomial time and effective methods exist[1].

Case II: $\lambda = r_i^2$ for some i. We will search for the optimal answer by setting λ to the radii of E, one at a time. So now we can assume that λ is known. Let $I = \{j | r_j^2 = \lambda\}$ and $\overline{I} = \{j | r_j^2 \neq \lambda\}$. Then using the first order equation, we can write

$$\sum_{k\in I} \frac{x_k^2}{r_k^2} + \sum_{k\in\overline{I}} \frac{1}{r_k^2}\left(\frac{c_k r_k^2}{r_k^2-\lambda}\right)^2 = 1$$

The only unknown variables are x_k for $k \in I$. Note that x_k forms a hyper-ellipsoid. In this case we can pick any set of x_k on this hyper-ellipsoid which is given by:

$$\sum_{k\in I} \frac{x_k^2}{r_k^2} = 1 - \sum_{k\in\overline{I}} \frac{1}{r_k^2}\left(\frac{c_k r_k^2}{r_k^2-\lambda}\right)^2$$

Note that for each $\lambda = r_i^2$, we are spending $O(d)$ time to find a corresponding solution. In the end, in $O(d^2)$ time, we can find the furthest of the solutions found for this case.

The total running time for the algorithm that does the search for the correct λ is hence $O(d^2)$. Note that Case II is indeed a degeneracy. If all z_i's are non-zero, we do not need to take care of Case II[2] (Which is the case, most of the time in practice).

□

Recently we have implemented a solution to this lemma in Matlab using maximum eigenvalue computation and a sample result is shown in figure 4(a) (Figure 4(b) shows a zoom in of the furthest computation done by our current implementation). In the near future we plan to replace the maximum eigenvalue computation with the largest root computation of a secular equation. The total time taken to compute this solution was a fraction of a second.

For finite precision arithmetic model, solving our problem gets messy [9]. Note that an error in the calculation of $\lambda_{\max}$ also incurs an error in the calculation of x^*. A small error in calculating $\lambda_{\max}$ could possibly result in a very large error in x^*. Fortunately, its very easy to show that this does not happen.

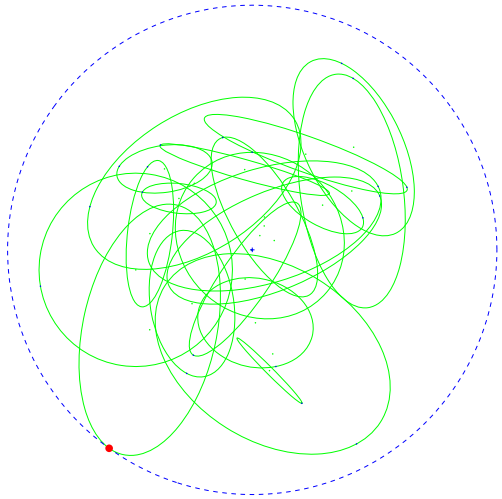
$$x_k^* + \delta_{x_k^*} = \frac{c_k r_k^2}{r_k^2 - (\lambda_{\max} + \delta_{\lambda_{\max}})} \quad \forall k = 1 \ldots d$$

using Taylor's expansion

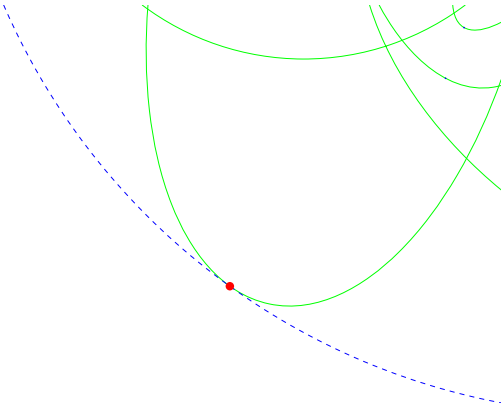
$$x_k^* + \delta_{x_k^*} = \frac{c_k r_k^2}{r_k^2 - \lambda_{\max}} + \frac{c_k r_k^2}{(r_k^2-\lambda_{\max})^2}\delta_{\lambda_{\max}} + O(\delta_{\lambda_{\max}}^2)$$

[1] The only problem with using polynomial time solvers for SDP problems is that this increases the running time of our algorithm by $\mathcal{O}(d^{O(1)})$. The advantage of using this method is that one does not need the augmented real number model of computation but assuming that one can store and do exact computations with real numbers in $\mathcal{O}(1)$ suffices.

[2] Another way to solve this *degenerate* case is by perturbation. For $z_i = 0$ we can perturb z_i infinitesimally to make it nonzero. Using the triangle inequality, it can be shown that the furthest point does not move too much in this case.



(a) Furthest point computation example



(b) Translation and Scaling

Figure 4. Furthest point computation figures for a set of 20 randomly generated ellipses in 2D. Zooming in on the left figure also shows the furthest point computed by our algorithm on each of the individual ellipses. Note that our current implementation in Matlab takes $\mathcal{O}(nd^3)$ to compute the furthest point. We hope to reduce it to at least $\mathcal{O}(nd^2)$ in the near future.

which shows that $\delta_{x_k^*}$ is linearly dependent on $\delta_{\lambda_{\max}}$. Thus solving $\lambda_{\max}$ to machine precision almost guarantees a machine precision solution to x^*. Note that this gives us the power to list the bits of x^* efficiently. Now we need to multiply the solution with rotation and scaling matrices. For a more detailed error analysis in the finite precision model, the reader is referred to [9, 23].

Note that both [9, 23] solve the the minimization version of (**FO**). Since we have already showed that the secular equation $\mathcal{P}(\lambda)$ has it's maximum root isolated, their results apply to our problem by replacing the minimum root with the maximum root.

The conversion of KY algorithm to work with ellipsoids is hence complete. Now we are ready to prove the main complexity result of this paper.

Theorem 3 *The total running time for the modified KY algorithm is $\mathcal{O}(nd^{2+\omega}/\epsilon)$ where $\omega < 2.376$ is the exponent of square matrix multiplication. On termination it gives a $(1+\epsilon)$-approximate MVEE of the input point set.*

Proof. Since we run KY as if it were running on a finite point set, it terminates with the same guarantees on the infinite point set as it would on the finite point set. This trick was also used in [16] to compute the minimum enclosing ball of a set of balls.

For the running time of the algorithm, the major time taken is solving (**FO**) in each iteration of algorithm 1. This takes $\mathcal{O}(nd^\omega + d^3)$ time per iteration. The total number of iterations is bounded by $\alpha = \mathcal{O}(d^2/\epsilon)$ and hence the total running time of our algorithm is $\mathcal{O}(\frac{d^2}{\epsilon}(nd^\omega + d^3))$. Note that there is only a $\mathcal{O}(d^{\omega-1})$ overhead to run KY on ellipsoids compared to points when $n > d$. □

Remark: Very recently and parallel to this work, Alper Yıldırım has announced a MVEE algorithm for computing MVEE of ellipsoids with a running time slightly larger than ours [32]. It also seems that our algorithm is simpler to implement. We plan to explore the practicality of this algorithm in the near future and report our results. We also hope to post the source code for both KY and its extensions on the web soon.

4 A Practical Variant

The first order algorithm, even though nice in theory turned out to be slow in practice when we implemented it (Please see next section). This motivated the design of a core-set/active-set based algorithm for solving our problem where we try to intelligently guess the important points in the input in each iteration and discard the inactive points. From our experiments it was clear that in each iteration we could gain a lot if we relied on a second order method compared to the first order method (as in the theoretical result). Algorithm 2 presents a practical variant of our theoretical method that we presented in the last section.

Algorithm 2 Second-order algorithm

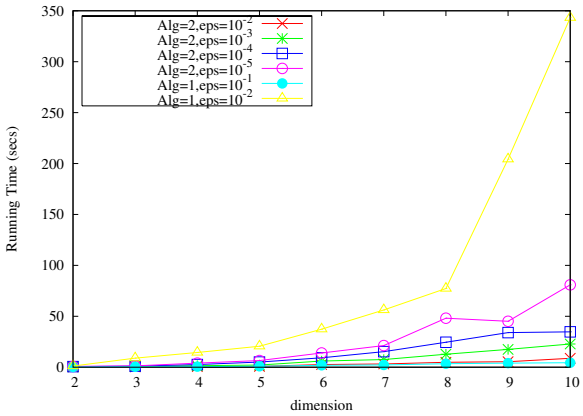
Require: Input set of ellipsoids,
$\mathcal{S} = \{E_1, E_2, \ldots, E_n\} \subseteq \mathbb{R}^d, \epsilon > 0, \mathcal{X} \subseteq \mathcal{S}$

1: **loop**
2: $\mathcal{E} \leftarrow MVE(X)$ (using [28])
3: $p_i = \max_{x \in E_i} ||x - center(\mathcal{E})||_{\mathcal{E}}\ \forall i \in 1 \ldots n$
4: $\mathcal{P} = \{p_i | p_i$ not contained in $(1+\epsilon)\mathcal{E}\}$.
5: **if** $|\mathcal{P}| == 0$ **then** break;
6: $\mathcal{P} =$ Project $p_i \in \mathcal{P}$ on corresponding $E_i \in \mathcal{S}$.
7: $X = X \cup$ Betke-Henk$(\mathcal{P})$
8: $X = X \setminus ((1-\epsilon_1)\mathcal{E} \cap X)$
9: **end loop**

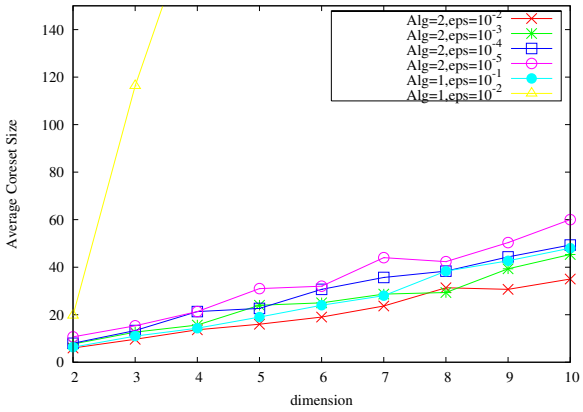
Finite termination of active set methods can be usually proved assuming that each of the subproblems/iterations are solved exactly [10, 6]. Without an exact solver, the proof of termination with gurantees on running times becomes rather complicated even for very simple problems [16]. For our practical variantriant, we chose to give up on guarantees for finite termination.

5 Implementation Results

A common drawback of first order algorithms is that they cannot achieve high accuracy. Although the algorithm we presented in this paper is the fastest known algorithm to solve the MVEE problem (when $n \gg d$ and ϵ is large), in practice, our experiments with KY motivated us to look for better algorithms, especially because we would like to solve problems with small ϵ. Our current implementation efforts suggest that depending on the value of ϵ, first or second order algorithms should be dynamically chosen in practice (or a combination used). Hence we have recently started exploring second order algorithms for implementation [16, 28]. The idea of the implementation is essentially the same but we solve each subproblem using a second order algorithm instead of just updating the first order solution. We believe that in practice this is the way to go even for moderately small ϵ although the complexity results for these algorithms do not look as promising as the first order algorithms. Even justifying termination or correctness for our practical variant seems tough unless we rely heavily on the augmented real number model of computation. Since the analysis and presentation of the implementation differs significantly from the algorithm presented in this paper, we decided not to include our second order implementation results in this paper. Figure 1 shows an example output of our current implementation that uses a second order algorithm to solve the subproblems (similar to [16]).



(a) Running time for 100 randomly generated ellipsoids

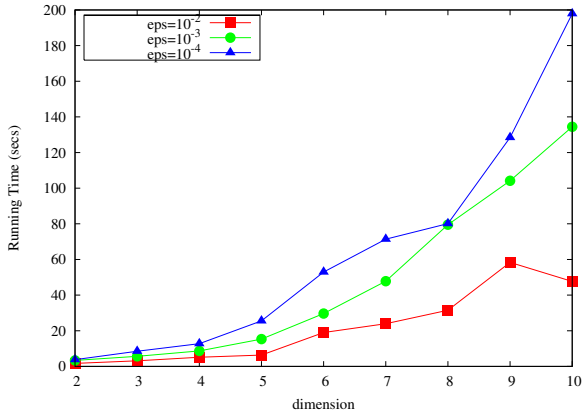


(b) Translation and Scaling

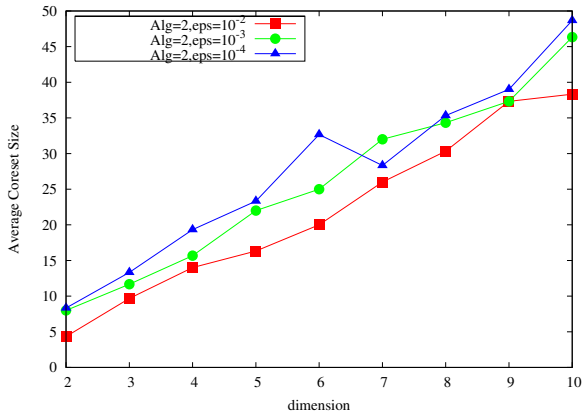
Figure 5. These graphs compare the running time and coreset sizes for 100 randomly generated ellipsoids in various dimension for various approximations.

6 Future Work

A very interesting open problem that this work opens up is how to analyze our algorithm in finite precision arithmetic and bound the error of the overall algorithm. It might be the case that the error analysis for our algorithm is complicated enough that there is a simpler way to design the entire algorithm and analyze the new algorithm. We plan to work on this in the near future. We also plan to make available our MVEE of ellipsoids code on the web.



(a) Running time for 100 randomly generated ellipsoids



(b) Translation and Scaling

Figure 6. These graphs compare the running time and coreset sizes for 100 randomly generated ellipsoids in various dimension for various approximations.

Acknowledgments: The authors are grateful to Alper Yıldırım and Kyle Gallivan for various discussions on this problem.

References

[1] G. Barequet and S. Har-Peled. Efficiently approximating the minimum-volume bounding box of a point set in three dimensions. *J. Algorithms*, 38:91–109, 2001. http://valis.cs.uiuc.edu/~sariel/research/papers/98/bbox.pdf.

[2] A. Ben-Tal and A. Nemirovski. *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*. MPS-SIAM Series on Optimization. SIAM, 2001.

[3] U. Betke and M. Henk. Approximating the volume of convex bodies. *Discrete Computational Geometry*, 10:15–21, 1993.

[4] L. Blum, F. Cucker, M. Shub, and S. Smale. *Complexity and real computation*. Springer-Verlag New York, Inc., 1998.

[5] C. Bouville. Bounding ellipsoid for ray-fractal intersection. In *SIGGRAPH*, pages 45–52, 1985.

[6] M. Bădoiu, S. Har-Peled, and P. Indyk. Approximate clustering via core-sets. *Proc. 34th Annu. ACM Sympos. on Theory of Computing*, pages 250–257, 2002. http://valis.cs.uiuc.edu/~sariel/papers/02/coreset/coreset.pdf.

[7] B. Chazelle and J. Matoušek. On linear-time deterministic algorithms for optimization problems in fixed dimension. In *SODA: ACM-SIAM Symposium on Discrete Algorithms*, pages 281–290, 1993.

[8] D. Eberly. *3D Game Engine Design*. Morgan Kaufmann, 2001.

[9] W. Gander, G. H. Golub, and U. von Matt. A constrained eigenvalue problem. *Linear Algebra and its Applications*, 114-115(1):815–839, 1989.

[10] P. E. Gill and W. Murray. *Numerical Methods for Constrained Optimization*. Academic Press, 1974.

[11] F. Glineur. Pattern separation via ellipsoids and conic programming. Master's thesis, Faculté Polytechnique de Mons, Belgium, 1998.

[12] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer, New York, 1988.

[13] F. John. Extremum problems with inequalities as subsidiary conditions. In *Studies and Essays, presented to R. Courant on his 60th birthday January 8, 1948*, pages 187–204. Interscience, New York, 1948. Reprinted in: *Fritz John, Collected Papers Volume 2* (J. Moser, ed), Birkhäuser, Boston, 1985, pp. 543–560.

[14] L. G. Khachiyan. Rounding of polytopes in the real number model of computation. *Mathematics of Operations Research*, 21:307–320, 1996.

[15] L. G. Khachiyan and M. J. Todd. On the complexity of approximating the maximal inscribed ellipsoid for a polytope. *Mathematical Programming*, 61:137–159, 1993.

[16] P. Kumar, J. S. B. Mitchell, and A. Yıldırım. Approximate minimum enclosing balls in high dimensions using core-sets. *The ACM Journal of Experimental Algorithmics*, 8(1), 2003. http://www.compgeom.com/meb/.

[17] P. Kumar and A. Yıldırım. Minimum volume enclosing ellipsoids and core sets. *Journal of Optimization Theory and Applications*, 126, 2005. To appear, http://www.ams.sunysb.edu/~piyush/papers/mve.pdf.

[18] H. W. Lenstra, Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8:538–548, 1983.

[19] O. E. Livne and A. Brandt. N roots of the secular equation in O(N) operations. *SIAM Journal of Matrix Anal. Appl.*, 24:439–453, 2002.

[20] J. Matousek, M. Sharir, and E. Welzl. A subexponential bound for linear programming. In *Proceedings of the eighth annual symposium on Computational geometry*, pages 1–8, 1992.

[21] D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. Quantile approximation for robust statistical estimation and k-enclosing problems. *International Journal of Computational Geometry and Applications*, 10(6):593–608, 2000.

[22] Y. E. Nesterov and A. S. Nemirovskii. *Interior Point Polynomial Methods in Convex Programming*. SIAM Publications, Philadelphia, 1994.

[23] E. Rimon and S. P. Boyd. Obstacle collision detection using best ellipsoid fit. *J. Intell. Robotics Syst.*, 18(2):105–126, 1997.

[24] A. Rybalov. On the P-NP problem over real matrix rings. *Theor. Comput. Sci.*, 314(1):281–285, 2004.

[25] B. W. Silverman and D. M. Titterington. Minimum covering ellipses. *SIAM J. Sci. Statist. Comput.*, 1:401–409, 1980.

[26] S. Silvey and D. Titterington. A geometric approach to optimal design theory. *Biometrika*, 62:21–32, 1973.

[27] J. F. Sturm and S. Zhang. On cones of nonnegative quadratic functions. *Math. Oper. Res.*, 28(2):246–267, 2003. http://www.optimization-online.org/DB_FILE/2001/05/324.pdf.

[28] P. Sun and R. M. Freund. Computation of minimum volume covering ellipsoids. Technical report, MIT Operations Research Center, 2002.

[29] S. P. Tarasov, L. G. Khachiyan, and I. I. Erlikh. The method of inscribed ellipsoids. *Soviet Mathematics Doklady*, 37:226–230, 1988.

[30] D. M. Titterington. Optimal design: some geometrical aspects of D-optimality. *Biometrika*, 62(2):313–320, 1975.

[31] E. Welzl. Smallest enclosing disks (balls and ellipsoids). In H. Maurer, editor, *Proceedings of New Results and New Trends in Computer Science*, volume 555 of *LNCS*, pages 359–370, Berlin, Germany, June 1991. Springer.

[32] A. Yildirim. On the minimum volume covering ellipsoid of ellipsoids. Unpublished Manuscript, http://www.optimization-online.org/DB_FILE/2005/01/1043.pdf.