A near-optimal algorithm for approximating the John Ellipsoid

Michael B. Cohen MICOHEN@MIT.EDU
MIT

Ben Cousins B.COUSINS@COLUMBIA.EDU
Columbia University

Yin Tat Lee YINTAT@UW.EDU
University of Washington

Xin Yang YX1992@CS.WASHINGTON.EDU
University of Washington

Editors: Alina Beygelzimer and Daniel Hsu

Abstract

We develop a simple and efficient algorithm for approximating the John Ellipsoid of a symmetric polytope. Our algorithm is near optimal in the sense that our time complexity matches the current best verification algorithm.

Experimental results suggest that our algorithm significantly outperforms existing algorithms. We also provide the MATLAB code for further research.

Keywords: John Ellipsoid, fixed point method, optimal design

1. Introduction

Let $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ be a polytope where P has nonzero, finite Euclidean volume. The classical theorem of John (1948) states that if $E \subseteq P$ is the ellipsoid of maximal volume contained in P, then $P \subseteq nE$, where nE represents a dilation of the ellipsoid E by a factor of n about its center. Moreover, if P is symmetric, then $P \subseteq \sqrt{n}E$. The maximal volume inscribed ellipsoid (MVIE) E is called the John Ellipsoid, and we are interested in the problem of approximating E when the polytope P is centrally symmetric, i.e. P can be expressed as $P = \{x \in \mathbb{R}^n : -\mathbf{1}_m \leq Ax \leq \mathbf{1}_m\}$ where $A \in \mathbb{R}^{m \times n}$ and A has rank n.

The problem of computing the ellipsoid of maximal volume inside polytope given by a set of inequalities has a wealth of different applications, including sampling and integration (Vempala, 2005; Chen et al., 2018), linear bandits (Bubeck et al., 2012; Hazan and Karnin, 2016), linear programming (Lee and Sidford, 2014), cutting plane methods (Khachiyan and Ehrlich, 1988) and differential privacy (Nikolov et al., 2013).

Computing the John Ellipsoid additionally has applications in the field of experimental design, a classical problem in statistics (Atwood, 1969). Specifically, the D-optimal design problem wants to maximize the determinant of the Fisher information matrix (Kiefer and Wolfowitz, 1960; Atwood, 1969), which turns out to be equivalent to finding the John Ellipsoid of a symmetric polytope. While this equivalence is known, e.g. Todd (2016), we include it in Section 2 for completeness. The problem of D-optimal design has received recent attention in the machine learning community, e.g. Allen-Zhu et al. (2017); Wang et al. (2017); Lu et al. (2018).

1.1. Our Contribution

Our main contribution is to develop an approximation algorithm to computing the John Ellipsoid inside a centrally symmetric polytope given by a set of inequalities. Previously, for solving the MVIE problem or its dual equivalent D-optimal design problem, researchers have developed various algorithms, such as first-order methods (Khachiyan, 1996; Kumar and Yildirim, 2005; Damla Ahipasaoglu et al., 2008), and second-order interior-point methods (Nesterov and Nemirovskii, 1994; Sun and Freund, 2004). Instead of using traditional optimization methods, we apply a very simple fixed point iteration. The analysis is also simple and clean, yet the convergence rate is very fast. We state our main result as follows.

Theorem 1 (Informal) *Given $A \in \mathbb{R}^{m\times n}$, let P be a centrally symmetric polytope defined as $\{x \in \mathbb{R}^n : -\mathbf{1}_m \le Ax \le \mathbf{1}_m\}$. For $\eta \in (0,1)$, there is an algorithm (Algorithm 1) that runs in time $O(\eta^{-1}mn^2\log(m/n))$, returning an ellipsoid Q so that $\frac{1}{\sqrt{1+\eta}}\cdot Q \subseteq P \subseteq \sqrt{n}\cdot Q$.*

In Lemma 5, we show that our ellipsoid is η-close to the John Ellipsoid in a certain sense. However, if we want to get $(1-\epsilon)$-approximation to the maximal volume, we shall set $\eta = \epsilon/n$[1], then Algorithm 1 runs in time $O(\epsilon^{-1}mn^3\log(\frac{m}{n}))$, and when ϵ is constant, this is comparable with the best known results $O(mn^3/\epsilon)$ (Kumar and Yildirim, 2005; Todd and Yıldırım, 2007).

Furthermore, we use sketching ideas from randomized linear algebra to speed up the algorithm so that the running time does not depend on m explicitly. This will make sense if A is a sparse matrix. Our result is stated as follows.

Theorem 2 (Informal) *Given $A \in \mathbb{R}^{m\times n}$, let P be a centrally symmetric polytope defined as $\{x \in \mathbb{R}^n : -\mathbf{1}_m \le Ax \le \mathbf{1}_m\}$. For $\eta \in (0,1)$ and $\delta \in (0,1)$, there is an algorithm (Algorithm 2) that runs within $O(\frac{1}{\eta}\log\frac{m}{\delta})$ many iterations, returning an ellipsoid Q so that with probability at least $1-\delta$, $\frac{1}{\sqrt{1+\eta}}\cdot Q \subseteq P \subseteq \sqrt{n}\cdot Q$. Moreover, each iteration involves in solving $O(\frac{1}{\eta})$ linear systems of the form $A^\top WAx = b$ where W is some diagonal matrix.*

Algorithm 2 is near optimal, because in order to verify the correctness of the result, we need to compute the leverage scores of some weighted version of A. The best known algorithm for approximating leverage scores needs to solve $\tilde{O}(\frac{1}{\eta^2})$ many linear systems (Spielman and Srivastava, 2011; Drineas et al., 2012; Clarkson and Woodruff, 2013; Nelson and Nguyên, 2013). One key advantage of our algorithm is that it reduces the problem of computing John ellipsoid to a relatively small number of linear systems. Therefore, it allows the user to apply the linear systems solver tailored for the given matrix A. For example, if A is tall, one can apply sketching technique to solve the linear systems in nearly linear time (Woodruff, 2014); if each row of A has only two non-zeros, one can apply Laplacian solvers (Daitch and Spielman, 2008; Kelner et al., 2013; Cohen et al., 2014; Kyng and Sachdeva, 2016; Kyng et al., 2016). In the code we provided, we used the Cholesky decomposition which is very fast for many sparse matrices A in practice.

Finally, we validate our algorithm on both synthetic and real data sets. Experiments show that our algorithm outperform all previous algorithms, which matches the theoretical calculation. We shall stress that our program can handle very large sparse matrices, which seems to be difficult for previous implementations. We attach our matlab code in Appendix D.

1. For details, see Lemma 5.

1.2. Related Works

There is a long line of research on computing the maximal volume ellipsoid inside polytopes given by a list of linear inequalities. We note that Khachiyan and Todd (1993) presented a linear time reduction from the problem of computing a minimum volume enclosing ellipsoid (MVEE) of a set of points to the maximal volume inscribed ellipsoid problem; therefore, these algorithms also hold for approximating the John Ellipsoid.

Using an interior-point algorithm, Nesterov and Nemirovskii (1994) showed that a $1+\epsilon$ approximation of MVEE can be computed in time $O(m^{2.5}(n^2+m)\log(\frac{m}{\epsilon}))$. Khachiyan and Todd (1993) subsequently improved the runtime to $O(m^{3.5}\log(\frac{m}{\epsilon})\cdot\log(\frac{n}{\epsilon}))$. Later on, Nemirovski (1999) and Anstreicher (2002) independently obtained an $O(m^{3.5}\log\frac{m}{\epsilon})$ algorithm. To the best of authors' knowledge, currently the best algorithms by Kumar and Yildirim (2005); Todd and Yıldırım (2007) run in time $O(mn^3/\epsilon)$. We refer readers to Todd (2016) for a comprehensive introduction and overview.

Computing the minimum volume enclosing ellipsoid of a set of points is the dual problem of D-optimal design. By generalizing smoothness condition on first order method, Lu et al. (2018) managed to solve D-optimal design problem within $O(\frac{m}{\epsilon}\log(\frac{n}{\epsilon}))$ many iterations. However, in the dense case, their iteration costs $O(mn^2)$ time, which leads to larger running time comparing to Kumar and Yildirim (2005); Todd and Yıldırım (2007). Gutman and Peña (2018) applied Bregman proximal method on the D-optimal design problem and observe accelerated convergence rate in their numerical experiments; however, they did not prove that their experimental parameter settings satisfy the assumption of their algorithm[2].

A natural version of the D-optimal design problem is to require an integral solution. The integral variant is shown to be $\mathbf{NP}$-hard (Černỳ and Hladík, 2012), although recently approximation algorithms have been developed (Allen-Zhu et al., 2017; Singh and Xie, 2018). In our context, this means the weight vector $w \in \mathbb{R}^m$ is the integral optimal solution to (2), where the sum of the weights is some specified integral parameter k.

Several Markov chains for sampling convex bodies have well understood performance guarantees based upon the roundedness of the convex body. If $B_n \subseteq K \subseteq R \cdot B_n$, then the mixing time of hit-and-run and the ball walk are both $O(n^2R^2)$ steps (Lovász and Vempala, 2006; Kannan et al., 1997). Thus, placing a convex body in John position guarantees the walks mix in $O(n^4)$ steps, and $O(n^3)$ steps if the body is symmetric; this transformation is used in practice with the convex body to be a polytope (Haraldsdóttir et al., 2017). Generating the John Ellipsoid, with a fixed center point, has also been employed as a proposal distribution for a Markov chain (Chen et al., 2018; Gustafson and Narayanan, 2018).

We build our even faster algorithm via sketching techniques. Sketching has been successfully applied to speed up different problems, such as linear programs (Lee et al., 2019), clustering (Cohen et al., 2015a; Song et al., 2018), low rank approximations (Clarkson and Woodruff, 2013; Nelson and Nguyên, 2013; Boutsidis and Woodruff, 2014; Clarkson and Woodruff, 2015b; Razenshteyn et al., 2016; Song et al., 2017), linear regression (Clarkson and Woodruff, 2013; Nelson and Nguyên, 2013; Cohen et al., 2015b; Clarkson and Woodruff, 2015a; Price et al., 2017; Andoni et al., 2018), total least regression (Diao et al., 2019), tensor regression Li et al. (2017); Diao et al. (2018) and tensor decomposition (Wang et al., 2015; Song et al., 2016, 2019). Readers may refer to Woodruff

2. We are grateful Gutman and Peña provided us their code for testing, of which D-optimal design was only one application.

(2014) for a comprehensive survey on sketching technique. We use sketching techniques to speed up computing leverage scores. This idea was first used in Spielman and Srivastava (2011).

Previous research on the MVEE problem did take advantage of the sparsity of the input matrix A, and to the best of our knowledge, our algorithm is the first one that is able to deal with large sparse input. It would be interesting if we can apply sketching techniques to further speed up existing algorithms.

1.2.1. RELATION WITH COHEN AND PENG (2015)

We shall mention that our work is greatly inspired by Cohen and Peng (2015). The ℓ_p *Lewis Weights* $\overline{w}$ for matrix $A \in \mathbb{R}^{m\times n}$ is defined as the unique vector $\overline{w}$ so that for $i \in [m]$,

$$a_i^\top \left(A^\top \mathsf{diag}(\overline{w})^{1-2/p} A\right)^{-1} a_i = \overline{w}_i^{2/p}.$$

It is known that computing the ℓ_∞ Lewis Weight is equivalent to computing the maximal volume inscribed ellipsoid. Cohen and Peng (2015) proposes an algorithm for approximating Lewis Weights for all $p < 4$. Their algorithm is an iterative algorithm that is very similar to our Algorithm 1, and the convergence is proved by arguing the iteration mapping is contractive. The main difference is that Cohen and Peng (2015) outputs the weights in the last round, while our Algorithm 1 takes the average over all rounds and outputs the averaging weights, which allows us to conduct a convexity analysis and deal with the ℓ_∞ case.

2. Problem Formulation

In this section, we formally define the problem of computing the John Ellipsoid of a symmetric polytope. Let $P = \{x \in \mathbb{R}^n : |a_i^\top x| \leq 1, i \in [m]\}$ be a symmetric convex polytope, where $[m]$ denotes the set $\{1, 2, \cdots, m\}$. We assume $A = (a_1\, a_2\, \cdots\, a_m)^\top$ has full rank. By symmetry, we know that the maximal volume ellipsoid inside the polytope should be centered at the origin. Any ellipsoid E centered at the origin can be expressed by $x^\top G^{-2} x \leq 1$, where G is a positive definite matrix. Note that the volume of E is proportional to $\det G$, and an ellipsoid E is contained in polytope P if and only if for $i \in [m]$, $\max_{x\in E} |a_i^\top x| \leq 1$. For any $x \in E$, we can write $x = Gy$ where $\|y\|_2 \leq 1$. Hence

$$\max_{x\in E} |a_i^\top x| = \max_{\|y\|_2\leq 1} |a_i^\top G y| = \max_{\|y\|_2\leq 1} \|Ga_i\|_2 \cdot \|y\|_2 = \|Ga_i\|_2$$

Therefore, we can compute the John Ellipsoid of P by solving the following optimization program:

$$\begin{aligned} \text{Maximize} \quad & \log\det G^2, \\ \text{subject to:} \quad & G \succeq 0, \\ & \|Ga_i\|_2 \leq 1, \quad \forall i \in [m]. \end{aligned} \tag{1}$$

It turns out that the optimal ellipsoid satisfies $G^{-2} = A^\top \mathsf{diag}(w) A$, where $w \in \mathbb{R}^m_{\geq 0}$ is the optimal solution of the program

$$\begin{aligned} \text{Minimize} \quad & \sum_{i=1}^m w_i - \log\det\left(\sum_{i=1}^m w_i a_i a_i^\top\right) - n, \\ \text{subject to:} \quad & w_i \geq 0, \quad \forall i \in [m]. \end{aligned} \tag{2}$$

Actually program (2) is the Lagrange dual of program (2). Moreover, we have the following optimality criteria for w in the above program.

Lemma 3 (Optimality criteria, Proposition 2.5 in Todd (2016)) *A weight w is optimal for program (2) if and only if*

$$\begin{aligned}
&\sum_{i=1}^m w_i = n, \\
&a_i^\top \left(\sum_{i=1}^m w_i a_i a_i^\top\right)^{-1} a_i = 1, && \text{if } w_i \neq 0; \\
&a_i^\top \left(\sum_{i=1}^m w_i a_i a_i^\top\right)^{-1} a_i < 1, && \text{if } w_i = 0.
\end{aligned}$$

Computing the John Ellipsoid is closely related to D-optimal design problem (Atwood, 1969; Boyd and Vandenberghe, 2004; Todd, 2016; Gutman and Peña, 2018). For the D-optimal design problem, we are given input $X \in \mathbb{R}^{n\times m}$ where $m > n$, and we want to solve program,

$$\begin{aligned}
\text{Maximize} \quad & \log\det\left(X \mathsf{diag}(v) X^\top\right), \\
\text{subject to:} \quad & v_i \geq 0,\ \forall i \in [m] \\
& \sum_{i=1}^m v_i = 1
\end{aligned} \tag{3}$$

We emphasize that program (3) and program (2) are equivalent, in the following sense. By Lemma 3, we can rewrite program (2) as minimizing $n\log n - \log\det(A^\top \mathsf{diag}(w) A)$, subject to $w_i \geq 0$ for $i \in [m]$ and $\sum_{i=1}^m w_i = n$. By setting $v_i = \frac{w_i}{n}$, we obtain program (3). Thus, optimal solutions to programs (2) and (3) are equivalent up to a multiplicative factor n.

We can also talk about an approximate John Ellipsoid.

Definition 4 *For $\epsilon > 0$, we say $w \in \mathbb{R}^m_{\geq 0}$ is a $(1+\epsilon)$-approximation of program (2) if w satisfies*

$$\begin{aligned}
&\sum_{i=1}^m w_i = n, \\
&a_i^\top \left(\sum_{i=1}^m w_i a_i a_i^\top\right)^{-1} a_i \leq 1+\epsilon, && \forall i \in [m].
\end{aligned}$$

Lemma 5 gives a geometric interpretation of the approximation factor in Definition 4. Recall that the exact John Ellipsoid Q^* of P satisfies $Q^* \subseteq P \subseteq \sqrt{n}\cdot Q^*$.

Lemma 5 ($(1+\epsilon)$-approximation is good rounding) *Let w be a $(1+\epsilon)$-approximation of (2). Define Q as $\{x : x^\top A^\top \mathsf{diag}(w) A x \leq 1\}$. Then*

$$\frac{1}{\sqrt{1+\epsilon}}\cdot Q \subseteq P \subseteq \sqrt{n}\cdot Q.$$

Moreover, $\mathbf{vol}\left(\frac{1}{\sqrt{1+\epsilon}}Q\right) \geq e^{-n\epsilon/2}\cdot \mathbf{vol}\left(Q^*\right)$.

Proof Let $G = \left(A^\top \mathsf{diag}(w) A\right)^{-\frac{1}{2}}$ and suppose $x \in \frac{1}{\sqrt{1+\epsilon}} Q$. Then, we have that $x^\top G^{-2} x \le \frac{1}{1+\epsilon}$. So,

$$|Ax|_i = \langle a_i, x\rangle = \langle G a_i, G^{-1} x\rangle \le \|G a_i\|_2 \|G^{-1} x\|_2 \le \frac{\|G a_i\|_2}{\sqrt{1+\epsilon}}.$$

Since $\|G a_i\|_2^2 = a_i^\top (\sum_{i=1}^m w_i a_i a_i^\top)^{-1} a_i \le 1+\epsilon$, then $|Ax|_i \le 1$ and $x \in P$.

On the other hand, for $x \in P$, we have that $|Ax|_i \le 1$. Hence

$$x^\top G^{-2} x = x^\top A^\top \mathsf{diag}(w) A x = \sum_{i=1}^m w_i |Ax|_i^2 \le \sum_{i=1}^m w_i = n.$$

So $P \subseteq \sqrt{n} \cdot Q$.

Finally, since $\frac{1}{\sqrt{1+\epsilon}} \cdot Q$ is contained in P, $G' = ((1+\epsilon) A^\top \mathsf{diag}(w) A)^{-\frac{1}{2}}$ is a feasible solution to program (1). Moreover w is a feasible solution to program (2). So by duality of program (1) and (2), we have the duality gap is at most

$$\left(n - \log\det\left(\sum_{i=1}^m w_i a_i a_i^\top\right) - n\right) - \log\det\left((1+\epsilon)\sum_{i=1}^m w_i a_i a_i^\top\right)^{-1} = n\log(1+\epsilon) \le n\epsilon.$$

Let the matrix representation of Q^* be $x^\top G_*^{-2} x \le 1$, then by optimality of G_* and the duality gap, we have

$$\log\det\left((G')^{-2}\right) \ge \log\det(G_*^2) - n\epsilon.$$

Since $\mathbf{vol}(\frac{1}{\sqrt{1+\epsilon}} \cdot Q)$ is proportional to $\det(G')^{-1}$, we conclude that $\mathbf{vol}(\frac{1}{\sqrt{1+\epsilon}} Q) \ge e^{-n\epsilon/2} \mathbf{vol}(Q^*)$. ■

3. Main Algorithm

In this section, we present Algorithm 1 for approximating program (2) and analyze its performance.

Let $\sigma : \mathbb{R}^m \to \mathbb{R}^m$ be the function defined as $\sigma(v) = (\sigma_1(v), \sigma_2(v), \cdots, \sigma_m(v))$ where for $i \in [m]$,

$$\sigma_i(v) = a_i^\top \left(\sum_{j=1}^m v_j a_j a_j^\top\right)^{-1} a_i = a_i^\top (A^\top \mathsf{diag}(v) A)^{-1} a_i. \tag{4}$$

Let w^* be the optimal solution to program (2). By Lemma (3), w^* satisfies $w_i^*(1 - \sigma_i(w^*)) = 0$, or equivalently

$$w_i^* = w_i^* \cdot \sigma_i(w^*) \tag{5}$$

Inspired by (5), we use the fixed point iteration $w_i^{(k+1)} = w_i^{(k)} \cdot \sigma_i(w^{(k)})$ for $k \in [T-1]$ and $i \in [m]$. $w_i^{(k)}$ has very nice properties. Actually, by setting $B^{(k)} = \sqrt{\mathsf{diag}(w^{(k)})} \cdot A$, we can rewrite $w_i^{(k)}$ as $(B_i^{(k)})^\top \left((B^{(k)})^\top B^{(k)}\right)^{-1} B_i^{(k)}$, hence $w_i^{(k)}$ is actually the *leverage score* of the i-th row of the matrix $B^{(k)}$ Cohen et al. (2015b). From the well-known properties of leverage scores, we have

Algorithm 1: Approximate John Ellipsoid inside symmetric polytopes

```
Input: A symmetric polytope given by $-\mathbf{1}_m \leq Ax \leq \mathbf{1}_m$, where $A \in \mathbb{R}^{m \times n}$ has rank $n$.
Result: Approximate John Ellipsoid inside the polytope.
1 Initialize $w_i^{(1)} = \frac{n}{m}$ for $i = 1, \cdots, m$.
2 for $k = 1, \cdots, T-1$, do
3     for $i = 1, \cdots, m$ do
          // We can use sketch technique to further speed up.
4         $w_i^{(k+1)} = w_i^{(k)} \cdot a_i^\top (A^\top \mathsf{diag}(w^{(k)}) A)^{-1} a_i$
5     end
6 end
7 $w_i = \frac{1}{T} \sum_{k=1}^{T} w_i^{(k)}$ for $i = 1, \cdots, m$.
8 $W = \mathsf{diag}(w)$. (i.e. $W$ is a diagonal matrix with the entries of $w$)
9 return $A^\top W A$
```

Lemma 6 (Properties of leverage scores, e.g. see Section 3.3 of Cohen et al. (2015b)) *For $k \in [T]$ and $i \in [m]$, we have $0 \leq w_i^{(k)} \leq 1$. Moreover, $\sum_{i=1}^m w_i^{(k)} = n$.*

In order to show Algorithm 1 provides a good approximation of the John Ellipsoid, in the sense of Definition 4, we need to argue that for the output w of Algorithm 1, $\sigma_i(w) \leq 1 + \epsilon$. Our main result is the following theorem.

Theorem 7 (Main Result) *Let w be the output of Algorithm 1 in line (7). For all $\epsilon \in (0, 1)$, when $T = \frac{2}{\epsilon} \log \frac{m}{n}$, we have for $i \in [m]$,*

$$\sigma_i(w) \leq 1 + \epsilon.$$

Moreover,

$$\sum_{i=1}^{m} w_i = n.$$

Therefore, Algorithm 1 provides $(1+\epsilon)$-approximation to program (2).

We now analyze the running time of Algorithm 1.

Theorem 8 (Performance of Algorithm 1) *For all $\epsilon \in (0, 1)$, we can find a $(1+\epsilon)$-approximation of John Ellipsoid inside a symmetric convex polytope in time $O\left(\epsilon^{-1} mn^2 \log \frac{m}{n}\right)$.*

Proof The main loop is executed $T = O(\frac{1}{\varepsilon} \log(\frac{m}{n}))$ times, and inside each loop, we can first use $O(mn)$ time to compute $B^{(k)} := (W^{(k)})^{\frac{1}{2}} A$, then compute $(B^{(k)})^\top B^{(k)}$ in $O(mn^2)$ time. To see why we introduce $B^{(k)}$, observe that $(B^{(k)})^\top B^{(k)} = A^\top W^{(k)} A$. Now we can compute the Cholesky decomposition of $(B^{(k)})^\top B^{(k)}$ in time $O(n^3)$, and use the Cholesky decomposition to compute $c_i := ((B^{(k)})^\top B^{(k)})^{-1} a_i = (A^\top W^{(k)} A)^{-1} a_i$ in time $O(n^2)$ for each $i \in [m]$. Finally, we can compute $w_i^{(k+1)}$ by computing $w_i^{(k)} \cdot a_i^\top c_i$ in time $O(n)$. This is valid since $w_i^{(k)} \cdot a_i^\top c_i = w_i^{(k)} \cdot a_i^\top (A^\top W^{(k)} A)^{-1} a_i = w_i^{(k)} \sigma_i(w^{(k)})$. To summarize, in each iteration we use $O(mn^2 + n^3 + mn^2) = O(mn^2)$ time, hence the overall running time is as stated. ■

Now we turn to proving Theorem 7. The proof of Theorem 7 relies on the following important observation, whose proof can be found in Appendix B.

Lemma 9 (Convexity) *For $i = 1, \cdots, m$, let $\phi_i : \mathbb{R}^m \to \mathbb{R}$ be the function defined as*

$$\phi_i(v) = \log \sigma_i(v) = \log \left(a_i^\top \left(\sum_{j=1}^m v_j a_j a_j^\top \right)^{-1} a_i \right).$$

Then ϕ_i is convex.

Now that ϕ_i is convex, we can apply Jensen's inequality to get Lemma 10.

Lemma 10 (Telescoping) *Fix T as the number of main loops executed in Algorithm 1. Let w be the output in line (7) of Algorithm 1. Then for $i \in [m]$,*

$$\phi_i(w) \le \frac{1}{T} \log \frac{m}{n}$$

Proof Recall that $w = \frac{1}{T}\sum_{k=1}^T w^{(k)}$. By Lemma 9, ϕ_i is convex, and so

$$\begin{aligned}
\phi_i(w) &= \phi_i\left(\frac{1}{T}\sum_{k=1}^T w^{(k)}\right) \\
&\le \frac{1}{T}\sum_{k=1}^T \phi_i(w^{(k)}) && \text{by Jensen's inequality} \\
&= \frac{1}{T}\sum_{k=1}^T \log \sigma_i(w^{(k)}) && \text{by definition of } \phi_i \text{ function} \\
&= \frac{1}{T}\sum_{k=1}^T \log \frac{w_i^{(k+1)}}{w_i^{(k)}} \\
&= \frac{1}{T} \log \frac{w_i^{(T+1)}}{w_i^{(1)}} \\
&\le \frac{1}{T} \log \frac{m}{n} && \text{by Lemma 6 and the initialization of } w^{(1)}
\end{aligned}$$

■

Now we are ready to prove Theorem 7.
Proof [Proof of Theorem 7] Set $T = \frac{2}{\epsilon}\log\frac{m}{n}$. By Lemma 10, we have for $i \in [m]$,

$$\log \sigma_i(w) = \phi_i(w) \le \frac{1}{T}\log\frac{m}{n} = \frac{\epsilon}{2} \le \log(1+\epsilon)$$

where the last step uses the fact that when $0 < \epsilon < 1$, $\frac{\epsilon}{2} \le \log(1+\epsilon)$. This gives us $\sigma_i(w) \le 1+\epsilon$.

On the other hand, from Lemma 6 we have $\sum_{i=1}^m w_i^{(k)} = n$. Hence

$$\sum_{i=1}^m w_i = \sum_{i=1}^m \frac{1}{T}\sum_{k=1}^T w_i^{(k)} = n$$

	FixedPoint		Todd		ABPGLS		Zhang	
	Its	Time(s)	Its	Time(s)	Its	Time(s)	Its	Time(s)
Scaled Boxes	3	0.0069	1	0.028	2	4.2	3	0.045
random Gaussian	6	0.022	4.7e+02	0.12	5	1.5	4	0.99
Tangent to ellipsoid	3	0.011	5.8e+02	0.15	2	2	2	0.36
Scaled balls	4	0.025	5.9e+02	0.2	9	18	4	4.7
Two ellipsoids	15	0.088	5.8e+02	0.19	30	31	6	7.8

Table 1: Number of iterations and total time for each algorithm on the set of examples. Each tested polytope is in dimension 1000 and each algorithm was run until $\varepsilon < 0.10$.

	FixedPoint		Todd		ABPGLS		Zhang	
	Its	Time(s)	Its	Time(s)	Its	Time(s)	Its	Time(s)
Scaled Boxes	3	0.0053	1	3.8	Inf[a][b]	Inf	3	3.1
Random Gaussian	1	0.054	5e+02	8.1	2	9.9	1	0.68
Tangent to ellipsoid	1	0.26	8e+03	94	2	9.4e+02	1	17
Scaled balls	3	0.27	7.5e+02	11	Inf	Inf	3	12
Two ellipsoids	10	6.4	7.8e+03	1.2e+02	Inf	Inf	3	8.3e+02

a. If an entry has “Inf", this means the program did not terminate with 30 minutes of computational time.

b. We run all the algorithms as provided, hence we suspect their algorithms can be better by parameter tuning and/or preprocessing.

■

We shall mention that it is possible to further improve Algorithm 1 by applying sketching technique from randomized linear algebra. Here we present the performance of our accelerated algorithm, and detailed analysis can be found in Appendix C.

Theorem 11 (Performance of Algorithm 2) *For all $\epsilon, \delta \in (0, 1)$, we can find a $(1+\epsilon)$-approximation of the John Ellipsoid inside a symmetric convex polytope within $O\left(\frac{1}{\epsilon}\log\frac{m}{\delta}\right)$ iterations with probability at least $1-\delta$. Moreover, each iteration involves solving $O(\frac{1}{\epsilon})$ linear systems of the form $A^\top W A x = b$ for some diagonal matrix W.*

4. Experimental Performance

In this section, we do a brief comparison of our algorithm to existing algorithms/tools for the maximum volume ellipsoid problem (or equivalently, algorithms for optimal D-design). The experimental results show that our algorithm performs favorably in the varied instances to the three tested packages, which we denote as “Todd" Todd (2016), “APBGLS" Gutman and Peña (2018), and “Zhang" Zhang and Gao (2003). We denote our algorithm as “FixedPoint", and it is available for reference in Appendix D. The theoretical gains of the algorithm appear to transfer to an improved practical algorithm. We test 7 types of instances:

1. **Scaled Boxes**: Two hypercubes of different side lengths. We tested $[-5, 5]^n \cap [-10, 10]^n$.

Table 2: Number of iterations and total time for each algorithm on the set of examples. Each tested polytope is in dimension 1000 and each algorithm was run until $\varepsilon < 0.001$.

	FixedPoint		Todd		ABPGLS		Zhang	
	Its	Time(s)	Its	Time(s)	Its	Time(s)	Its	Time(s)
Scaled Boxes	6	0.0045	1	3.2	Inf	Inf	4	4.3
Random Gaussian	5	0.25	2e+03	19	6	36	3	3.2
Tangent to ellipsoid	34	9.4	1.9e+04	2.2e+02	Inf	Inf	5	2.7e+02
Scaled balls	1.2e+02	9.7	3.8e+03	37	Inf	Inf	7	29
Two ellipsoids	2.3e+02	1.3e+02	3.2e+04	4.7e+02	Inf	Inf	Inf	Inf

Table 3: Number of iterations and total time for each algorithm on various netlib examples. If a row has multiple instances, the reported iteration counts and times are reported as mean/median of the instances. maros_r7 has $m = 9408, n = 3136$, osa_07 has $m = 25067, n = 1118$, qap12 has $m = 8856, n = 2794$. Each algorithm was run until $\varepsilon < 0.10$.

	FixedPoint		Todd		ABPGLS		Zhang	
	Its	Time(s)	Its	Time(s)	Its	Time(s)	Its	Time(s)
Netlib small	7.1/6.5	0.12/0.02	230/58	4.8/0.31	29/29	57/8.2	3.9/3.5	9.1/0.43
maros_r7	3	11.1	1	137	Inf	Inf	4	197
osa_07	12	2.6	2404	67	Inf	Inf	Inf	Inf
qap12	8	38.5	7751	570	7	610	4	145

2. **Random Gaussian**: $A \in \mathbb{R}^{m\times n}$, and each entry of A is an iid standard Gaussian. We set $m = 3n/2$.

3. **Tangent to ellipsoid**: $A \in \mathbb{R}^{m\times n}$, and each facet of A is tangent to a random ellipsoid (defined by an $n \times n$ Gaussian matrix). We set $m = 10n$.

4. **Scaled balls**: $A \in \mathbb{R}^{m\times n}$. Balls of radius r_1, r_2 each have $m/2$ facets tangent to the ball, where each facet is chosen according to the uniform measure from the respective sphere. We set $m = 3n$, $r_1 = \sqrt{n}/2$, and $r_2 = \sqrt{n}/2 + 10$.

5. **Two ellipsoids**: A consists of two independent realizations of the class **Tangent to ellipsoid**. We set $m = 20n$.

6. **Netlib small**: As a form of “real-world" and sparse examples, we consider a subset of Netlib polytopes. In this class, we consider those instances that (i) have a full-dimensional dual that only contains inequality constraints and (ii) have $m + n < 10^4$. We symmetrize the polytope

with respect to the analytic center in the dual space. 39[3] of the 109 models in the dataset satisfy these conditions.

7. **Netlib large**: We consider those Netlib polytopes that satisfy the same conditions as **Netlib small**, but have $m + n \in [10^4, 5 \cdot 10^4]$. 5 Netlib polytopes satisfy these conditions. [4]

Acknowledgments

We thank Zhao Song for his generous help on this paper.

References

Zeyuan Allen-Zhu, Yuanzhi Li, Aarti Singh, and Yining Wang. Near-optimal design of experiments via regret minimization. In *International Conference on Machine Learning*, pages 126–135, 2017.

Alexandr Andoni, Chengyu Lin, Ying Sheng, Peilin Zhong, and Ruiqi Zhong. Subspace embedding and linear regression with Orlicz norm. In *International Conference on Machine Learning*, pages 224–233, 2018.

Kurt M Anstreicher. Improved complexity for maximum volume inscribed ellipsoids. *SIAM Journal on Optimization*, 13(2):309–320, 2002.

Corwin L Atwood. Optimal and efficient designs of experiments. *The Annals of Mathematical Statistics*, pages 1570–1602, 1969.

Christos Boutsidis and David P Woodruff. Optimal CUR matrix decompositions. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*, pages 353–362. ACM, https://arxiv.org/pdf/1405.7910, 2014.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, Cambridge, 2004.

Sébastien Bubeck, Nicolo Cesa-Bianchi, and Sham Kakade. Towards minimax policies for online linear optimization with bandit feedback. In *Annual Conference on Learning Theory*, volume 23, pages 41–1. Microtome, 2012.

Michal Černỳ and Milan Hladík. Two complexity results on C-optimality in experimental design. *Computational Optimization and Applications*, 51(3):1397–1408, 2012.

Yuansi Chen, Raaz Dwivedi, Martin J Wainwright, and Bin Yu. Fast MCMC sampling algorithms on polytopes. *The Journal of Machine Learning Research*, 19(1):2146–2231, 2018.

3. However, the reported results in Table 3 are only on 35 of these 39 models. For two of the instances, our algorithm fails due to numerical issues in the Cholesky factorization; this could likely be fixed by padding the matrix $B' \cdot B$ with a small multiple of the identity matrix, but we ignored this issue for cleanliness of the code. Two other instances take too long for the other implementations. Thus, all 4 sets of results for **Netlib small** were run on the same 35 models.
4. However, similar to the **Netlib small** case, two of the cases fail our method due to numerical issues with the Cholesky factorization.

Kenneth L Clarkson and David P Woodruff. Low rank approximation and regression in input sparsity time. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 81–90. ACM, 2013.

Kenneth L Clarkson and David P Woodruff. Sketching for M-estimators: A unified approach to robust regression. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 921–939. Society for Industrial and Applied Mathematics, 2015a.

Kenneth L Clarkson and David P Woodruff. Input sparsity and hardness for robust subspace approximation. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 310–329. IEEE, 2015b.

Michael B Cohen and Richard Peng. ℓ_p row sampling by Lewis Weights. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 183–192. ACM, 2015.

Michael B. Cohen, Rasmus Kyng, Gary L. Miller, Jakub W. Pachocki, Richard Peng, Anup B. Rao, and Shen Chen Xu. Solving SDD linear systems in nearly $m \log^{1/2} n$ time. In *Proceedings of the Forty-sixth Annual ACM Symposium on Theory of Computing*, STOC '14, 2014.

Michael B Cohen, Sam Elder, Cameron Musco, Christopher Musco, and Madalina Persu. Dimensionality reduction for k-means clustering and low rank approximation. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 163–172. ACM, 2015a.

Michael B Cohen, Yin Tat Lee, Cameron Musco, Christopher Musco, Richard Peng, and Aaron Sidford. Uniform sampling for matrix approximation. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, pages 181–190. ACM, 2015b.

Samuel I Daitch and Daniel A Spielman. Faster approximate lossy generalized flow via interior point algorithms. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 451–460. ACM, 2008.

S Damla Ahipasaoglu, Peng Sun, and Michael J Todd. Linear convergence of a modified Frank–Wolfe algorithm for computing minimum-volume enclosing ellipsoids. *Optimisation Methods and Software*, 23(1):5–19, 2008.

Huaian Diao, Zhao Song, Wen Sun, and David Woodruff. Sketching for kronecker product regression and p-splines. In *International Conference on Artificial Intelligence and Statistics*, pages 1299–1308, 2018.

Huaian Diao, Zhao Song, David Woodruff, and Xin Yang. Total least squares regression in input sparsity time. In *Manuscript*, 2019.

Petros Drineas, Malik Magdon-Ismail, Michael W Mahoney, and David P Woodruff. Fast approximation of matrix coherence and statistical leverage. *Journal of Machine Learning Research*, 13 (Dec):3475–3506, 2012.

Izrail Solomonovich Gradshteyn and Iosif Moiseevich Ryzhik. *Table of integrals, series, and products*. Academic press, 2014.

Adam Gustafson and Hariharan Narayanan. John's walk. *arXiv preprint arXiv:1803.02032*, 2018.

David H Gutman and Javier F Peña. A unified framework for bregman proximal methods: subgradient, gradient, and accelerated gradient schemes. *arXiv preprint arXiv:1812.10198*, 2018.

Hulda S. Haraldsdóttir, Ben Cousins, Ines Thiele, Ronan M. T. Fleming, and Santosh Vempala. CHRR: coordinate hit-and-run with rounding for uniform sampling of constraint-based models. *Bioinformatics*, 33(11), 1 2017.

Elad Hazan and Zohar Karnin. Volumetric spanners: an efficient exploration basis for learning. *The Journal of Machine Learning Research*, 17(1):4062–4095, 2016.

GJO Jameson. Inequalities for gamma function ratios. *The American Mathematical Monthly*, 120 (10):936–940, 2013.

Fritz 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 Publishers, Inc., New York, N. Y., 1948.

Ravi Kannan, László Lovász, and Miklós Simonovits. Random walks and an $O^*(n^5)$ volume algorithm for convex bodies. *Random Structures Algorithms*, 11(1):1–50, 1997.

Jonathan A Kelner, Lorenzo Orecchia, Aaron Sidford, and Zeyuan Allen Zhu. A simple, combinatorial algorithm for solving SDD systems in nearly-linear time. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 911–920. ACM, 2013.

Leonid G Khachiyan. Rounding of polytopes in the real number model of computation. *Mathematics of Operations Research*, 21(2):307–320, 1996.

Leonid G Khachiyan and Michael J Todd. On the complexity of approximating the maximal inscribed ellipsoid for a polytope. *Mathematical Programming*, 61(1):137–159, 1993.

Tarasov S. Khachiyan, L. and I. Ehrlich. The method of inscribed ellipsoids. *Soviet Math. Doklady*, 1988.

Jack Kiefer and Jacob Wolfowitz. The equivalence of two extremum problems. *Canadian Journal of Mathematics*, 12(363-366):234, 1960.

Piyush Kumar and E Alper Yildirim. Minimum-volume enclosing ellipsoids and core sets. *Journal of Optimization Theory and Applications*, 126(1):1–21, 2005.

Rasmus Kyng and Sushant Sachdeva. Approximate gaussian elimination for laplacians-fast, sparse, and simple. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 573–582. IEEE, 2016.

Rasmus Kyng, Yin Tat Lee, Richard Peng, Sushant Sachdeva, and Daniel A Spielman. Sparsified cholesky and multigrid solvers for connection laplacians. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 842–850. ACM, 2016.

Beatrice Laurent and Pascal Massart. Adaptive estimation of a quadratic functional by model selection. *Annals of Statistics*, pages 1302–1338, 2000.

Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in $O(\sqrt{rank})$ iterations and faster algorithms for maximum flow. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 424–433. IEEE, 2014.

Yin Tat Lee, Zhao Song, and Qiuyi Zhang. Solving empirical risk minimization in the current matrix multiplication time. In *COLT*, 2019.

Xingguo Li, Jarvis Haupt, and David Woodruff. Near optimal sketching of low-rank tensor regression. In *Advances in Neural Information Processing Systems*, pages 3466–3476, 2017.

László Lovász and Santosh Vempala. Hit-and-run from a corner. *SIAM J. Comput.*, 35(4):985–1005, 2006.

Haihao Lu, Robert M Freund, and Yurii Nesterov. Relatively smooth convex optimization by first-order methods, and applications. *SIAM Journal on Optimization*, 28(1):333–354, 2018.

Jelani Nelson and Huy L Nguyên. OSNAP: Faster numerical linear algebra algorithms via sparser subspace embeddings. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 117–126. IEEE, 2013.

Arkadi Nemirovski. On self-concordant convex–concave functions. *Optimization Methods and Software*, 11(1-4):303–384, 1999.

Yurii Nesterov and Arkadii Nemirovskii. *Interior-point polynomial algorithms in convex programming*, volume 13. Siam, 1994.

Aleksandar Nikolov, Kunal Talwar, and Li Zhang. The geometry of differential privacy: the sparse and approximate cases. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 351–360. ACM, 2013.

Eric Price, Zhao Song, and David P. Woodruff. Fast regression with an ℓ_∞ guarantee. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, 2017.

Ilya Razenshteyn, Zhao Song, and David P Woodruff. Weighted low rank approximations with provable guarantees. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 250–263. ACM, 2016.

Mohit Singh and Weijun Xie. Approximate positive correlated distributions and approximation algorithms for D-optimal design. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2240–2255. Society for Industrial and Applied Mathematics, 2018.

Zhao Song, David P. Woodruff, and Huan Zhang. Sublinear time orthogonal tensor decomposition. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems (NIPS) 2016, December 5-10, 2016, Barcelona, Spain*, pages 793–801, 2016.

Zhao Song, David P Woodruff, and Peilin Zhong. Low rank approximation with entrywise ℓ_1-norm error. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 688–701. ACM, 2017.

Zhao Song, Lin F Yang, and Peilin Zhong. Sensitivity sampling over dynamic geometric data streams with applications to k-clustering. *arXiv preprint arXiv:1802.00459*, 2018.

Zhao Song, David P Woodruff, and Peilin Zhong. Relative error tensor low rank approximation. In *SODA*. https://arxiv.org/pdf/1704.08246, 2019.

Daniel A Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011.

Peng Sun and Robert M Freund. Computation of minimum-volume covering ellipsoids. *Operations Research*, 52(5):690–706, 2004.

Michael J. Todd. *Minimum-volume ellipsoids : theory and algorithms*. MOS-SIAM series on optimization. Philadelphia, 2016.

Michael J Todd and E Alper Yıldırım. On khachiyan's algorithm for the computation of minimum-volume enclosing ellipsoids. *Discrete Applied Mathematics*, 155(13):1731–1744, 2007.

Santosh Vempala. Geometric random walks: a survey. In *Combinatorial and computational geometry*, volume 52 of *Math. Sci. Res. Inst. Publ.*, pages 577–616. Cambridge Univ. Press, Cambridge, 2005.

Yining Wang, Hsiao-Yu Tung, Alexander J Smola, and Anima Anandkumar. Fast and guaranteed tensor decomposition via sketching. In *Advances in Neural Information Processing Systems (NIPS)*, pages 991–999. https://arxiv.org/pdf/1506.04448, 2015.

Yining Wang, Adams Wei Yu, and Aarti Singh. On computationally tractable selection of experiments in measurement-constrained regression models. *The Journal of Machine Learning Research*, 18(1):5238–5278, 2017.

David P Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and Trends® in Theoretical Computer Science*, 10(1–2):1–157, 2014.

Yin Zhang and Liyan Gao. On numerical solution of the maximum volume ellipsoid problem. *SIAM Journal on Optimization*, 14(1):53–76, 2003.

Appendix A. Preliminaries

In this section we introduce notations and preliminaries used in the appendix. We use $N(\mu, \sigma^2)$ to represent the normal distribution with mean μ and variance σ^2.

A.1. Multivariate Calculus

Let $f : \mathbb{R}^m \to \mathbb{R}^n$ be a differentiable function. The *directional derivative* of f in the direction h is defined as

$$Df(x)[h] := \frac{\mathrm{d} f(x + th)}{\mathrm{d} t}\Big|_{t=0}.$$

We can also define a high order directional derivative as

$$D^k f(x)[h_1, \cdots, h_k] := \frac{\mathrm{d}^k f(x + \sum_{i=1}^k t_i h_i)}{\mathrm{d}t_1 \mathrm{d}t_2 \cdots \mathrm{d}t_k}\Big|_{t_1=0,\ldots,t_k=0}.$$

The following two properties of directional derivatives will be useful.

Proposition 12

- *(Chain rule)* $Df(g(x))[h] = f'(g(x)) \cdot Dg(x)[h]$.
- *Let* $f(X) = X^{-1}$ *where* $X \in \mathbb{R}^{n \times n}$. *For* $H \in \mathbb{R}^{n \times n}$, $Df(X)[H] = X^{-1}HX^{-1}$.

A.2. Gamma Function

Γ function is a well-known math object. It is defined as

$$\Gamma(z) = \int_0^{+\infty} x^{z-1} e^{-x} \mathrm{d}x.$$

We need the following result on Gamma function.

Lemma 13 (Corollary 1 of Jameson (2013)) *For all* $x > 0$ *and* $0 \le y \le 1$,

$$x(x+y)^{y-1} \le \frac{\Gamma(x+y)}{\Gamma(x)} \le x^y.$$

A.3. Tail Bound for χ^2 Distribution

We need the following version of concentration for χ^2 distribution.

Lemma 14 (Lemma 1 in Laurent and Massart (2000)) *Let* $X \sim \chi^2(n)$ *be a* χ^2 *distribution with* n *degree of freedom. Then for* $t > 0$,

$$\Pr[X - n \ge 2\sqrt{nt} + 2t] \le e^{-t}.$$

Appendix B. Proof of Lemma 9

In this section we provide the proof of Lemma 9.

Proof We first prove a strengthened result: for fixed $a \in \mathbb{R}^n$, the function $f : S^n_{++} \to \mathbb{R}$ defined as $f(M) = \log(a^\top M^{-1} a)$ is convex. Here S^n_{++} is the set of all positive definite $n \times n$ matrices. Notice that S^n_{++} is an open set, so we can differentiate f.

We argue that it is sufficient to show for all $M \in GL_n$ and all $H \in \mathbb{R}^{n \times n}$, the second order directional derivative $D^2 f(M)[H, H]$ is non-negative. This is because $D^2 f(M)[H,H] = H^\top \nabla^2 f(M) H$. So if for all H we have $H^\top \nabla^2 f(M) H \ge 0$, then $\nabla^2 f(M) \succeq 0$, which is precisely the convex condition.

Let us do some computation with Proposition 12.

$$Df(M)[H] = -\frac{a^\top M^{-1} H M^{-1} a}{a^\top M^{-1} a},$$

and

$$D^2 f(M)[H,H] = \frac{2a^\top M^{-1}HM^{-1}HM^{-1}a \cdot a^\top M^{-1}a - (a^\top M^{-1}HM^{-1}a)^2}{(a^\top M^{-1}a)^2}.$$

By Cauchy-Schwarzt inequality, we have

$$\begin{aligned} a^\top M^{-1}HM^{-1}HM^{-1}a \cdot a^\top M^{-1}a =& \|M^{-\frac{1}{2}}HM^{-1}a\|_2^2 \cdot \|M^{-\frac{1}{2}}a\|_2^2 \\ \geq& (\langle M^{-\frac{1}{2}}HM^{-1}a, M^{-\frac{1}{2}}a\rangle)^2 \\ =& (a^\top M^{-1}HM^{-1}a)^2. \end{aligned}$$

Hence $D^2 f(M)[H,H] \geq \frac{a^\top M^{-1}HM^{-1}HM^{-1}a \cdot a^\top M^{-1}a}{(a^\top M^{-1}a)^2} \geq 0$ for all $M \in GL_n$ and all $H \in \mathbb{R}^{n\times n}$.

Now we are ready to work on ϕ_i. For all v, v' in the domain of ϕ_i, let $M = \sum_{i=1}^m v_i a_i a_i^\top$ and $M' = \sum_{i=1}^m v_i' a_i a_i^\top$. Then for all $\lambda \in [0,1]$,

$$\begin{aligned} \phi_i(\lambda v + (1-\lambda)v') &= \log a_i^\top \left(\sum_{i=1}^m (\lambda v_i + (1-\lambda)v_i')a_i a_i^\top\right)^{-1} a_i \\ &= \log a_i^\top \left(\lambda \sum_{i=1}^m v_i a_i a_i^\top + (1-\lambda)\sum_{i=1}^m v_i' a_i a_i^\top\right)^{-1} a_i \\ &= f(\lambda M + (1-\lambda)M') \\ &\leq \lambda f(M) + (1-\lambda) f(M') \qquad \text{because } f \text{ is convex} \\ &= \lambda \phi_i(v) + (1-\lambda)\phi_i(v'). \end{aligned}$$

So ϕ_i is also convex. ■

Appendix C. Faster Algorithm for Computing John Ellipsoid for Sparse Matrix

In this section we present our accelerated algorithm, Algorithm 2 and analyze its performance. Recall that Algorithm 1 uses the iterating rule $w \leftarrow w \cdot \sigma(w)$ where $\sigma_i(w) = a_i^\top (A^\top \mathsf{diag}(w) A)^{-1} a_i$. With our setting of $B^{(k)}$, we have

$$(B^{(k)})^\top B^{(k)} = A^\top \sqrt{W^{(k)}} \cdot \sqrt{W^{(k)}} A = A^\top W^{(k)} A.$$

So $\hat{w}_i^{(k+1)} = \|B^{(k)}((B^{(k)})^\top B^{(k)})^{-1}(\sqrt{w_i^{(k)}} a_i)\|_2^2 = w_i \cdot \sigma_i(w)$ is just what we do in Algorithm 1. Hence from Lemma 6, we obtain the following properties about $\hat{w}_i^{(k)}$.

Proposition 15 (Bound on $\hat{w}^{(k)}$) *For completeness we define $\hat{w}^{(1)} = w^{(1)}$. For $k \in [T]$ and $i \in [m]$, $0 \leq \hat{w}_i^{(k)} \leq 1$. Moreover, $\sum_{i=1}^m \hat{w}_i^{(k)} = n$.*

However, $B^{(k)}$ is a m by n matrix, so it is computationally expensive to compute $\hat{w}_i^{(k+1)}$. The trick we use here, which is initially introduced first by Spielman and Srivastava (2011), is to introduce a random Gaussian matrix S with s rows to speed up the computation. Of course, this will introduce extra error, however we can prove that the overall result has good concentration.

Algorithm 2: Faster Algorithm for approximating John Ellipsoid inside symmetric polytopes

```
Input: A symmetric polytope given by $-\mathbf{1}_m \le Ax \le \mathbf{1}_m$, where $A \in \mathbb{R}^{m\times n}$
Result: Approximate John Ellipsoid inside the polytope
1  initialize $w_i^{(1)} = \frac{n}{m}$ for $i = 1, \cdots, m$.
2  for $k = 1, \cdots, T-1$, do
3      $W^{(k)} = \mathsf{diag}(w^{(k)})$.
4      $B^{(k)} = \sqrt{W^{(k)}}A$.
5      Let $S^{(k)} \in \mathbb{R}^{s\times m}$ be a random matrix where each entry is chosen i.i.d from $N(0,1)$, i.e. the
        standard normal distribution.
6      for $i = 1, \cdots, m$ do
           // Ideally we want to compute
              $\hat{w}_i^{(k+1)} = \|B^{(k)}((B^{(k)})^\top B^{(k)})^{-1}(\sqrt{w_i^{(k)}}a_i)\|_2^2$.
           // But this is expensive, so we use sketching technique to
              speed up.
7          $w_i^{(k+1)} = \frac{1}{s}\cdot\|S^{(k)}B^{(k)}((B^{(k)})^\top B^{(k)})^{-1}(\sqrt{w_i^{(k)}}a_i)\|_2^2$.
8      end
9  end
10 $w_i = \frac{1}{T}\sum_{k=1}^T w_i^{(k)}$ for $i = 1, \cdots, m$.
11 $v_i = \frac{n}{\sum_{j=1}^m w_j} w_i$ for $i = 1, \cdots, m$.
12 $V = \mathsf{diag}(v)$.
13 return $A^\top V A$
```

C.1. Approximation Guarantee

Since Algorithm 2 is a randomized algorithm, we need to argue that for the output v of Algorithm 2, $\sigma_i(v) \le 1+\epsilon$ with high probability. Our main result in this section is

Theorem 16 (Main result) *Let w be the output in line (10) of Algorithm 2. For all $\epsilon, \delta \in (0,1)$, when $T = \frac{10}{\epsilon}\log\frac{m}{\delta}$ and $s = \frac{80}{\epsilon}$, we have*

$$\Pr[\forall i \in [m], \sigma_i(w) \le 1+\epsilon] \ge 1-\delta.$$

Moreover, before rescaling at the end,

$$\Pr\left[\sum_{i=1}^m w_i \le (1+\epsilon)n\right] \ge 1-\delta.$$

By scaling w so that $\sum_{i=1}^m w_i = n$, we have

Theorem 17 (Approximation guarantee) *For all $\epsilon, \delta \in (0,1)$, When $T = O(\frac{1}{\epsilon}\log\frac{m}{\delta})$ and $s = O(\frac{1}{\epsilon})$, Algorithm 2 provides a $(1+\epsilon)^2$-approximation to program (2) with probability at least $1-2\delta$.*

Proof On line (11) of Algorithm 2 we set $v = \frac{n}{\sum_{i=1}^m w_i}w$, hence $\sum_{i=1}^m v_i = n$.

From Theorem 16, with probability at least $1-2\delta$, $\forall i \in [m], \sigma_i(w) \le 1+\epsilon$, and $\sum_{i=1}^m w_i \le (1+\epsilon)n$. Therefore for all $i \in [m]$,

$$\sigma_i(v) = a_i^\top (A^\top \mathsf{diag}(v) A)^{-1} a_i = a_i^\top \left(\frac{n}{\sum_{i=1}^m w_i} \cdot A^\top \mathsf{diag}(w) A \right)^{-1} a_i = \frac{\sum_{i=1}^m w_i}{n} \sigma_i(w) \le (1+\epsilon)^2.$$

■

From now on we focus on proving Theorem 16. Recall that $\phi_i(w) = \log \sigma_i(w)$ for $i \in [m]$. Similar to Lemma 10, we can prove the following lemma with the convexity of ϕ_i.

Lemma 18 (Telescoping) *Fix T as the number of main loops executed in Algorithm 2. Let w be the output at line (10) of Algorithm 2. Then for $i \in [m]$,*

$$\phi_i(w) \le \frac{1}{T} \log \frac{m}{n} + \frac{1}{T} \sum_{k=1}^T \log \frac{\hat{w}_i^{(k)}}{w_i^{(k)}}.$$

Proof Recall that $w = \frac{1}{T}\sum_{k=1}^T w^{(k)}$. By Lemma 9, ϕ_i is convex, so we can apply Jensen's inequality to obtain

$$\begin{aligned}
\phi_i(w) &\le \frac{1}{T}\sum_{k=1}^T \phi_i(w^{(k)})) && \text{Jensen's inequality} \\
&= \frac{1}{T}\sum_{k=1}^T \log \sigma_i(w^{(k)}) && \text{by definition of } \phi_i \text{ function} \\
&= \frac{1}{T}\sum_{k=1}^T \log \frac{\hat{w}_i^{(k+1)}}{w_i^{(k)}} && \hat{w}_i^{(k+1)} = w_i^{(k)} \sigma_i(w^{(k)}) \\
&= \frac{1}{T}\sum_{k=1}^T \log \frac{\hat{w}_i^{(k+1)}}{\hat{w}_i^{(k)}} \frac{\hat{w}_i^{(k)}}{w_i^{(k)}} \\
&= \frac{1}{T}(\sum_{k=1}^T \log \frac{\hat{w}_i^{(k+1)}}{\hat{w}_i^{(k)}} + \sum_{k=1}^T \log \frac{\hat{w}_i^{(k)}}{w_i^{(k)}}) \\
&= \frac{1}{T}\log \frac{\hat{w}_i^{(T+1)}}{\hat{w}_i^{(1)}} + \frac{1}{T}\sum_{k=1}^T \log \frac{\hat{w}_i^{(k)}}{w_i^{(k)}} \\
&\le \frac{1}{T}\log \frac{m}{n} + \frac{1}{T}\sum_{k=1}^T \log \frac{\hat{w}_i^{(k)}}{w_i^{(k)}}. && \text{by Proposition 15 and the initialization of } w^{(1)}
\end{aligned}$$

■

From Lemma 18, we can bound the expectation of ϕ_i directly.

Lemma 19 (Expectation of $\log \sigma_i$) *If s is even, then*

$$\mathbf{E}[\phi_i(w)] = \mathbf{E}\left[\log a_i^\top \left(\sum_j w_j a_j a_j^\top \right)^{-1} a_i \right] \le \frac{1}{T}\log\frac{m}{n} + \frac{2}{s}.$$

where the randomness is taken over the sketching matrices $\{S^{(k)}\}_{k=1}^{T-1}$.

Proof Recall the update rule

$$w_i^{(k+1)} = \frac{1}{s} \cdot \|S^{(k)} B^{(k)} ((B^{(k)})^\top B^{(k)})^{-1} (\sqrt{w_i^{(k)}} a_i)\|_2^2.$$

Let $y_i^{(k)} = B^{(k-1)}((B^{(k-1)})^\top B^{(k-1)})^{-1}(\sqrt{w_i^{(k-1)}} a_i)$ be a vector of size m. Then $\hat{w}_i^{(k)} = \|y_i^{(k)}\|_2^2$, and $w_i^{(k)} = \frac{1}{s}\|S^{(k)} y_i^{(k)}\|_2^2$, where each entry of $S^{(k)}$ is chosen i.i.d from $N(0,1)$.

Fix $y_i^{(k)}$. Let us consider the distribution of $w_i^{(k)}$. We first consider 1 coordinate of $S^{(k)} y_i^{(k)}$, which is $(S^{(k)} y_i^{(k)})_j = \sum_{t=1}^m S_{jt}^{(k)} (y_i^{(k)})_t$. Since each $S_{jt}^{(k)}$ is chosen from $N(0,1)$, $S_{jt}^{(k)}(y_i^{(k)})_t$ follows the distribution $N(0, (y_i^{(k)})_t^2)$, and $(S^{(k)} y_i^{(k)})_j$ follows the distribution $N(0, \sum_{t=1}^m (y_i^{(k)})_t^2) = N(0, \|y_i^{(k)}\|_2^2) = \|y_i^{(k)}\|_2 \cdot N(0,1)$. Hence $w_i^{(k)}$ follows the distribution of $\frac{1}{s}\|y_i^{(k)}\|_2^2 \cdot \chi^2(s)$ where $\chi^2(s)$ is χ^2-distribution with s degree of freedom.

Hence if we only consider the randomness of matrix $S^{(k)}$, then we have

$$\mathop{\mathbf{E}}_{S}\left[\log \frac{\hat{w}_i^{(k)}}{w_i^{(k)}}\right] = \mathop{\mathbf{E}}_{z\sim\chi^2(s)}\left[\log \frac{\|y_i^{(k)}\|_2^2}{\frac{1}{s}\|y_i^{(k)}\|_2^2 \cdot z}\right] = \mathop{\mathbf{E}}_{z\sim\chi^2(s)}[\log \frac{s}{z}].$$

Hence by the pdf of the χ^2-distribution, assuming s is even, we have that

$$\mathop{\mathbf{E}}_{z\sim\chi^2(s)}[\log z] = \int_0^\infty \frac{1}{2^{s/2}\Gamma(s/2)} x^{s/2-1} e^{-x/2} \log x \mathrm{d}x = \sum_{i=1}^{s/2-1} \frac{1}{i} - \gamma + \log 2.$$

The last equation use 4.352-2 of the 5th edition of Gradshteyn and Ryzhik (2014), and γ is the Euler constant. Hence

$$\mathop{\mathbf{E}}_{z\sim\chi^2(s)}[\log \frac{s}{z}] = \log s - \left(\sum_{i=1}^{s/2-1} \frac{1}{i} - \gamma + \log 2\right) \le \log s - (\log \frac{s}{2} + \gamma - \gamma + \log 2 - \frac{2}{s}) = \frac{2}{s}.$$

by the fact that $\sum_{i=1}^k \frac{1}{i} \ge \log k + \gamma$.

Therefore we have that

$$\mathbf{E}[\phi_i(w)] \le \frac{1}{T} \log \frac{m}{n} + \frac{2}{s}.$$

■

Since $\phi_i(w) = \log \sigma_i(w)$, Lemma 19 already provides some concentration results on $\sigma_i(w)$. We can also prove stronger type of concentration by bounding the moments of $\sigma_i(w)$ directly.

Lemma 20 (Moments of σ_i) *For* $\alpha > 0$, *if* $\frac{s}{2} > \frac{\alpha}{T}$, *then*

$$\mathbf{E}[\sigma_i(w)^\alpha] = \mathbf{E}\left[(a_i^\top (\sum_j w_j a_j a_j^\top)^{-1} a_i)^\alpha\right] \le (\frac{m}{n})^{\frac{\alpha}{T}} \cdot (1 + \frac{2\alpha}{sT - 2\alpha})^T.$$

Proof By Lemma 18 we have

$$\sigma_i(w)^\alpha \le (\frac{m}{n})^{\frac{\alpha}{T}} \cdot \prod_{k=1}^{T} (\frac{\hat{w}^{(k)}}{w^{(k)}})^{\frac{\alpha}{T}}.$$

Fix k and $\hat{w}^{(k)}$. Similarly as proof of Lemma 19, for the moment let us only consider the randomness of $S^{(k)}$, then we have

$$\mathop{\mathbf{E}}_{S^{(k)}}\left[(\frac{\hat{w}^{(k)}}{w^{(k)}})^{\frac{\alpha}{T}}\right] = \mathop{\mathbf{E}}_{z\sim\chi^2(s)}\left[(\frac{s}{z})^{\frac{\alpha}{T}}\right].$$

Hence we have

$$\begin{aligned}\mathop{\mathbf{E}}_{S^{(k)}}\left[(\frac{\hat{w}^{(k)}}{w^{(k)}})^{\frac{\alpha}{T}}\right] &= \int_0^\infty (\frac{s}{x})^{\frac{\alpha}{T}} \cdot \frac{1}{2^{s/2}\Gamma(s/2)} x^{s/2-1}e^{-x/2}\mathsf{d}x \\ &= \frac{s^{\frac{\alpha}{T}}}{2^{s/2}\Gamma(s/2)} \int_0^\infty x^{\frac{s}{2}-\frac{\alpha}{T}-1}e^{-x/2}\mathsf{d}x \\ &= \frac{s^{\frac{\alpha}{T}}}{2^{s/2}\Gamma(s/2)} \cdot \frac{\Gamma(s/2-\alpha/T)}{(1/2)^{s/2-\alpha/T}} \\ &= (s/2)^{\frac{\alpha}{T}} \cdot \frac{\Gamma(s/2-\alpha/T)}{s/2}.\end{aligned}$$

where the third line uses 3.381-4 in Gradshteyn and Ryzhik (2014) and the condition that $\frac{s}{2} > \frac{\alpha}{T}$.

By Lemma 13,

$$\frac{\Gamma(\frac{s}{2})}{\Gamma(\frac{s}{2}-\frac{\alpha}{T})} \ge \frac{(\frac{s}{2}-\frac{\alpha}{T})}{(\frac{s}{2})^{1-\frac{\alpha}{T}}},$$

which gives us

$$\mathop{\mathbf{E}}_{S^{(k)}}\left[(\frac{\hat{w}^{(k)}}{w^{(k)}})^{\frac{\alpha}{T}}\right] \le (\frac{s}{2})^{\frac{\alpha}{T}} \cdot \frac{(\frac{s}{2})^{1-\frac{\alpha}{T}}}{(\frac{s}{2}-\frac{\alpha}{T})} = \frac{\frac{s}{2}}{\frac{s}{2}-\frac{\alpha}{T}}.$$

Because each $S^{(k)}$ matrix is independent to each other, we have

$$\mathbf{E}[\sigma_i(w)^\alpha] \le (\frac{m}{n})^{\frac{\alpha}{T}} \cdot (\frac{\frac{s}{2}}{\frac{s}{2}-\frac{\alpha}{T}})^T = (\frac{m}{n})^{\frac{\alpha}{T}} \cdot (1+\frac{2\alpha}{sT-2\alpha})^T.$$

■

Now we are ready to prove Theorem 16.

Proof [Proof of Theorem 16] Set $\alpha = \frac{3}{\epsilon}\log\frac{m}{\delta}$. We can verify that

$$\alpha \ge \frac{\log(m/\delta)}{\log\frac{1+\epsilon}{1+\epsilon/4}}. \tag{6}$$

Notice that in this setting, we have $sT \geq 4\alpha$. Therefore, for $i \in [m]$, by Markov's inequality, we have

$$\begin{aligned}
\Pr[\sigma_i(w) \geq 1+\epsilon] &= \Pr[\sigma_i(w)^\alpha \geq (1+\epsilon)^\alpha] \\
&\leq \frac{\mathbf{E}[\sigma_i(w)^\alpha]}{(1+\epsilon)^\alpha} \\
&\leq \frac{(\frac{m}{n})^{\frac{\alpha}{T}} \cdot (1+\frac{2\alpha}{sT-2\alpha})^T}{(1+\epsilon)^\alpha} && \text{by Lemma 20} \\
&\leq \frac{(\frac{m}{n})^{\frac{\alpha}{T}} \cdot (1+\frac{2\alpha}{sT/2})^T}{(1+\epsilon)^\alpha} && \text{because } sT \geq 4\alpha \\
&\leq \frac{(\frac{m}{n})^{\frac{\alpha}{T}} e^{\frac{4\alpha}{s}}}{(1+\epsilon)^\alpha} && \text{Here we use } 1+x \leq e^x
\end{aligned}$$

With our choice of s, T, we can check that for sufficiently large m, n,

$$(\frac{m}{n})^{\frac{1}{T}} = (\frac{m}{n})^{\frac{\epsilon/10}{\log(m/\delta)}} \leq 1+\epsilon/10,$$

and

$$e^{\frac{4}{s}} = e^{\frac{\epsilon}{20}} \leq 1+\epsilon/10.$$

Hence

$$\Pr[\sigma_i(w) \geq 1+\epsilon] \leq \left(\frac{(1+\epsilon/10)^2}{1+\epsilon}\right)^\alpha \leq \left(\frac{1+\epsilon/4}{1+\epsilon}\right)^\alpha.$$

Then by (6), we have

$$\Pr[\sigma_i(w) \geq 1+\epsilon] \leq \frac{\delta}{m}.$$

By union bound, we have that

$$\Pr[\exists i \in [m], \sigma_i(w) \geq 1+\epsilon] \leq \delta.$$

Then let us prove the second part in Theorem 16. Fix k. Recall that $B^{(k)} = \sqrt{W^{(k)}}A$. Let $D^{(k)} = B^{(k)}((B^{(k)})^\top B^{(k)})^{-1}(B^{(k)})^\top$, then we can check that $D^{(k)}$ is an orthogonal projection matrix, because

$$(D^{(k)})^2 = \Big(B^{(k)}((B^{(k)})^\top B^{(k)})^{-1}(B^{(k)})^\top\Big) \cdot \Big(B^{(k)}((B^{(k)})^\top B^{(k)})^{-1}(B^{(k)})^\top\Big) = D^{(k)}.$$

Since $\mathsf{rank}(D^{(k)}) = n$, we can diagonalize $D^{(k)}$ as $D^{(k)} = \Lambda^{-1}E_n\Lambda$ where Λ is an $m \times m$ orthogonal matrix, and $E_n \in \mathbb{R}^{m\times m}$ is a diagonal matrix where the first n diagonal entries are 1 and all the other entries are 0. So we can rewrite the update rule as

$$\begin{aligned}
w_i^{(k+1)} &= \frac{1}{s} \cdot \|S^{(k)}B^{(k)}((B^{(k)})^\top B^{(k)})^{-1}(\sqrt{w_i^{(k)}}a_i)\|_2^2 \\
&= \frac{1}{s} \cdot \Big((S^{(k)}D^{(k)})^\top (S^{(k)}D^{(k)})\Big)_{ii}.
\end{aligned}$$

Therefore

$$\begin{aligned}
\sum_{i=1}^m w_i^{(k+1)} =& \frac{1}{s} \cdot \sum_{i=1}^m \Big((S^{(k)} D^{(k)})^\top (S^{(k)} D^{(k)}) \Big)_{ii} \\
=& \frac{1}{s} \cdot \mathsf{Tr} \left[((S^{(k)} D^{(k)})^\top (S^{(k)} D^{(k)}) \right] \\
=& \frac{1}{s} \cdot \mathsf{Tr} \left[(D^{(k)})^\top (S^{(k)})^\top S^{(k)} D^{(k)} \right] \\
=& \frac{1}{s} \cdot \mathsf{Tr} \left[(S^{(k)})^\top S^{(k)} (D^{(k)})^2 \right] && D^{(k)} \text{ is symmetric} \\
=& \frac{1}{s} \cdot \mathsf{Tr} \left[(S^{(k)})^\top S^{(k)} D^{(k)} \right] && (D^{(k)})^2 = D^{(k)} \\
=& \frac{1}{s} \cdot \mathsf{Tr} \left[S^{(k)} \Lambda^{-1} E_n \Lambda (S^{(k)})^\top \right]. && \text{diagonalization of } D^{(k)}
\end{aligned}$$

Let $\hat{S}^{(k)} = S^{(k)} \Lambda^{-1}$. Here Λ depends on previous randomness. Notice that Gaussian distribution is invariant under orthogonal transform, and $S^{(k)}$ is independent to previous S matrices. We conclude that $\hat{S}^{(k)}$ also has i.i.d entries that follows the distribution $N(0,1)$, and $\hat{S}^{(k)}$ is independent to previous randomness.

So we have

$$\sum_{i=1}^m w_i^{(k+1)} = \frac{1}{s} \cdot \mathsf{Tr}[\hat{S}^{(k)} E_n (\hat{S}^{(k)})^\top] = \frac{1}{s} \sum_{p=1}^s \sum_{q=1}^n (\hat{S}^{(k)})_{pq}^2.$$

Namely, the distribution of $\sum_{i=1}^m w_i^{(k+1)}$ is $\frac{1}{s}\chi^2(ns)$. Because for different k, the randomness are independent, we have $\sum_{i=1}^m w_i = \frac{1}{T}(\sum_{k=1}^T \sum_{i=1}^m w_i^{(k)})$ follows the distribution $\frac{1}{sT}\chi^2(nsT)$. So we can set $t = \frac{1}{16}\epsilon^2 \cdot nsT = \Theta(n \log \frac{m}{\delta})$ in Lemma 14 to see that for sufficiently large m, n,

$$\begin{aligned}
\Pr\left[\sum_{i=1}^m w_i^{(k+1)} \geq (1+\epsilon)n \right] =& \Pr_{z \sim \frac{1}{sT}\chi^2(nsT)} [z \geq (1+\epsilon)n] && \text{set } z = \sum_{i=1}^m w_i^{(k+1)} \\
=& \Pr_{z \sim \chi^2(nsT)} [z \geq (1+\epsilon)nsT] && \text{rescale } z \\
=& \Pr_{z \sim \chi^2(nsT)} [z - nsT \geq \epsilon \cdot nsT] \\
\leq& \Pr_{z \sim \chi^2(nsT)} \left[z - nsT \geq 2t + 2\sqrt{nsT \cdot t} \right] \\
\leq& e^{-t} \leq \delta. && \text{by Lemma 14}
\end{aligned}$$

■

C.2. Runtime analysis

We now analyze the running time of Algorithm 2. The main loop is executed T times, and inside each loop, we can first use $O(s \cdot \mathsf{nnz}(B^{(k)}))$ time to compute $S^{(k)} B^{(k)}$, then solve s linear systems to compute $F^{(k)} := S^{(k)} B^{(k)} ((B^{(k)})^\top B^{(k)})^{-1}$. Finally we can compute all of $\|F^{(k)} \cdot (w_i^k)^{1/2} \cdot a_i\|_2^2$ in $O(s \cdot \mathsf{nnz}(A))$ time by first computing matrix $S^{(k)} \cdot \sqrt{W^{(k)}} \cdot A^\top$. Recall that $B^{(k)} = \sqrt{W^{(k)}} A$, so $\mathsf{nnz}(B^{(k)}) \leq \mathsf{nnz}(A)$. Notice that it takes at least $\mathsf{nnz}(A)$ time to solve the linear system $A^\top W^{(x)} A x = b$. Together with Theorem 17, this gives us

Theorem 21 (Restatement of Theorem 11) *For all $\epsilon, \delta \in (0,1)$, we can find a $(1+\epsilon)$-approximation of John Ellipsoid inside a symmetric polytope within $O\left(\frac{1}{\epsilon} \log \frac{m}{\delta}\right)$ iterations with probability at least $1 - \delta$. Moreover, each iteration involves solving $O(\frac{1}{\epsilon})$ linear systems of the form $A^\top W A x = b$ for some diagonal matrix W.*

Appendix D. Matlab Code

```
% Compute the John ellipsoid weight for a full rank matrix A
function [w_avg, iter] = FixedPoint(A,tol,w)
m = size(A,1); n = size(A,2);
exactTime = Inf; useJL = false;
if ~exist('tol', 'var'), tol = 0.01; end
if ~exist('w', 'var'), w = ones(m,1)*n/m; end
if issparse(A), A = A(:,colamd(A)); useJL = true; end

w_avg = w;
for iter = 1:ceil(10*log(m/n)/tol)
    tau = computeTau(w, 10+iter); %increase the JLdim

    if max(tau./w) < 1 + tol
        w_avg = w; %use only the current w if we detect it has convgerged
        break;
    end
    w = tau; %update w

    if useJL
        w_avg = (1-1/iter)*w_avg + w/iter;
        if randi(iter) == 1 %with prob. 1/iter, test if we have converged
            tau_avg = computeTau(w_avg, 10+iter^2); %check conv of running avg
            if max(tau_avg./w_avg) < 1 + tol, break; end
        end
    end
end

% compute tau = diag(B*inv(B'*B)*B') with B = sqrt(W) A
function tau = computeTau(w, JLdim)
    tstart = tic; %record times to decide when to use sketching
    B = spdiags(sqrt(w),0,m,m)*A;
    R = chol(B' * B); % In rare cases, chol fails due to numerical error

    if nnz(R) > n^2/20, A = full(A); useJL = false; end % use full if dense

        % remove if too slow
    if (exactTime == Inf || ~useJL) % use JL only if it is much faster
        S = R' \ B';
        tau = full(sum(S.^2,1)'); % compute tau exactly
        exactTime = toc(tstart);
    else
        S = B * (R\randn(n, JLdim));
        tau = full(sum(S.^2,2)/JLdim);  % compute tau using JL
        JLTime = toc(tstart);
        if JLTime*(10+iter) > exactTime, useJL = false; end
    end
end
end
```