Math. Prog. Comp. (2016) 8:133–160
DOI 10.1007/s12532-015-0097-z

FULL LENGTH PAPER

A practical volume algorithm

Ben Cousins[1] · **Santosh Vempala**[1]

Received: 2 December 2013 / Accepted: 9 October 2015 / Published online: 28 October 2015

Abstract We present a practical algorithm for computing the volume of a convex body with a target relative accuracy parameter $\varepsilon > 0$. The convex body is given as the intersection of an explicit set of linear inequalities and an ellipsoid. The algorithm is inspired by the volume algorithms in Lovász and Vempala (J Comput Syst Sci 72(2):392–417, 2006) and Cousins and Vempala (SODA, pp. 1215–1228, 2014), but makes significant departures to improve performance, including the use of empirical convergence tests, an adaptive annealing scheme and a new rounding algorithm. We propose a benchmark of test bodies and present a detailed evaluation of our algorithm. Our results indicate that that volume computation and integration might now be practical in moderately high dimension (a few hundred) on commodity hardware.

Keywords Volume computation · Convex geometry · Random walks · Hit-and-run · Simulated annealing

Mathematics Subject Classification 65D18 · 52A38 · 52-04

1 Introduction

High-dimensional integration and sampling is a fundamental problem of interest across the sciences and engineering [1–6]. In 1989, Dyer et al. [7,8] found an $O^*(n^{23})$

✉ Ben Cousins
bcousins3@gatech.edu

✉ Santosh Vempala
vempala@cc.gatech.edu; vempala@gatech.edu

[1] School of Computer Science, Algorithms and Randomness Center, Georgia Institute of Technology, Atlanta, GA, USA

algorithm for computing the volume of a convex body (the O^* notation suppresses dependences on log factors and error parameters). Their approach was extended to polynomial time algorithms for integrating logconcave functions. Since then, over the past quarter century, the theoretical efficiency of algorithms for computing the volume and for integration has gone down from $O^*(n^{23})$ membership tests to $O^*(n^4)$ such tests [9]. Recently, we found an $O^*(n^3)$ algorithm for computing the Gaussian measure of a convex body [10] and an $O^*(n^3)$ algorithm for computing the volume of a well-rounded convex body [11].

Together with today's computation speeds, this progress suggests the possibility of practical multi-dimensional integration. Indeed, the first such implementation of the Lovász-Vempala $O^*(n^4)$ algorithm was reported in 2012 [12]; however, it could only compute the volume of cubes of dimension up to 9, with higher dimensional cubes taking prohibitively long. There are several reasons for this, including (a) the complexities above are for the number of membership tests; each test typically takes quadratic or higher number of arithmetic operations, (b) the theoretical bounds have large constants (e.g., 10^{10} and multiple logarithmic factors), and (c) the algorithms are designed for the the worst-case, i.e., they include routines such as: "run for $f(n)$" steps, where $f(n)$ is independent of the input body. These aspects had to be addressed to some extent in [12] to obtain a working implementation even in very small dimension.

In this paper, we present a more practical algorithm for computing volume and Gaussian volume of a polyhedron, which can handle 100 dimensional bodies in as little as 10 minutes. We also extend this to polyhedra intersected with ellipsoids. The algorithm builds on [9–11], but crucially needs a few more ideas. We present extensive experimental results. The MATLAB implementation is publicly available on MATLAB's File Exchange [13]. To the best of our knowledge, there is no benchmark for high-dimensional volume computation. So we propose a family of test bodies that could serve this purpose for future improvements and algorithms. Following our work, [14] gave a C++ implementation for volume computation of polytopes and reported even faster results on some bodies in our benchmark.

In the next section, we describe the main ideas of the algorithm, noting clearly where we extend previous work to obtain a practical algorithm. Following that, we describe the key aspects of our implementation in details, with rigorous justifications to the extent possible. In Sect. 4, we present computational results of our algorithm. We conclude this section with our proposed benchmark for evaluating volume algorithms.

1.1 Benchmark

We propose the following families of test bodies. The first 5 families have volumes that can be computed efficiently by simple formulas, thus serve as testable instances in any dimension. Within these 5, families 2(b), 4 and 5 will typically require that a volume algorithm perform some type of "rounding" step to maintain efficiency. The last two families of bodies have volumes that can be computed exactly, but the best known algorithms to compute the volume take exponential time. Therefore, approximating the volume is necessary for efficiency. We discuss these test families in more detail when we present the results of our evaluation.

1. *Cube:* a standard $[-1, 1]^n$ cube, which has volume 2^n.
2. (a) *Isotropic simplex:* a regular n-simplex, which has volume $\sqrt{n+1}/(n!\sqrt{2^n})$. (b) *Standard simplex:* defined as $\{x \in \mathbb{R}^n : \sum_{i=1}^n x_i \leq 1, x_i \geq 0\}$, that is all coordinates are nonnegative and sum to at most 1. The volume of the standard simplex is $1/n!$.
3. *Half ball:* the n-dimensional unit ball, with the restriction that $x_1 \geq 0$. The volume of this body is $1/2 \cdot \pi^{n/2}/\Gamma(n/2+1)$.
4. *Transformed cube:* starting from the *Cube*, we apply a random linear transformation T–each entry of the n by n matrix T was chosen from $\mathcal{N}(0, 1)$. The volume is then $|T|2^n$.
5. *Ellipsoid:* an axis-aligned ellipsoid with radius 1 along $n-1$ axes and radius 100 along 1 axis. The volume of the shape is then $100\pi^{n/2}/\Gamma(n/2+1)$.
6. *Zonotope:* the Minkowski sum of m line segments where each line segment is in $\mathbb{R}^n$. The volume of a zonotope can be computed exactly with a direct method, but the algorithm will take exponential time for general m, n (roughly $\binom{m}{n}$). It is, in fact, #P-hard to compute the volume of a zonotope.
7. *Birkhoff polytope:* the polytope of all $n \times n$ doubly stochastic matrices of dimension $n^2 - 2n + 1$. The volume has been computed exactly for values of $n \leq 10$ using specialized algorithms, but is unknown for $n > 10$.

2 Algorithm

At a high level, our algorithm is based on that of [9] (henceforth referred to as the LV algorithm). We give an overview here, and a detailed discussion of each component in our implementation in Sect. 3. First, note that computing volume is a special case of integration. That is, the volume of K can be expressed as

$$\mathsf{vol}(K) = \int_K \mathbb{1}\, dx.$$

The above quantity is hard to compute, or even estimate, directly, but there is an insight which makes the problem tractable. Consider the following representation of $\mathsf{vol}(K)$, for any function $f : \mathbb{R}^n \to \mathbb{R}$:

$$\mathsf{vol}(K) = \frac{\int_K f(x)\, dx}{\int_K f(x)\, dx} \cdot \int_K \mathbb{1}\, dx = \int_K f(x)\, dx \cdot \frac{\int_K \mathbb{1}\, dx}{\int_K f(x)\, dx}.$$

Now the difficult task of computing the volume has been reduced to two, perhaps easier, tasks: (1) compute the ratio of two integrals and (2) integrating the function f over K. Note that the above representation can be extended to any sequence of functions $\{f_0, \ldots, f_{m-1}\}$ where each $f_i : \mathbb{R}^n \to \mathbb{R}$. Given this sequence of functions, the volume of K can be rewritten as

$$\mathsf{vol}(K) = \int_K f_0(x)\, dx \cdot \frac{\int_K f_1(x)\, dx}{\int_K f_0(x)\, dx} \cdot \frac{\int_K f_2(x)\, dx}{\int_K f_1(x)\, dx} \cdots \frac{\int_K \mathbb{1}\, dx}{\int_K f_{m-1}(x)\, dx}$$

The functions f_i can be chosen to be anything we like, but should be selected so that each term in the above equation is efficiently computable. First, we select f_0 such that $\int_K f_0(x)dx$ is directly computable. For instance, f_0 can be the indicator function for a ball B where $B \subseteq K$, and then $\int_K f_0(x)dx = \mathsf{vol}(B)$, which has a nice, direct formula. Another option is to select f_0 to be a low variance Gaussian centered inside K; that is, the weight of the Gaussian will be highly concentrated around a single point inside K. Then, if the Gaussian is sufficiently "sharp", then $\int_K f_0(x)dx \approx \int_{\mathbb{R}^n} f_0(x)dx$, and integrating a Gaussian over the full space $\mathbb{R}^n$ again has a nice, direct formula. For the implementation, we select f_0 as a low variance Gaussian; further details on how we select f_0 are given in Sect. 3.2.1.

Next, we want to efficiently compute each integral ratio:

$$\frac{\int_K f_i(x)\,dx}{\int_K f_{i-1}(x)\,dx}.$$

At first glance, the above problem seems just as intractable as volume. However, we do not need an exact answer, and instead want to estimate the above integral ratio within some target relative error. Estimating this ratio of integrals is where sampling comes into the picture. Suppose we had a random sample from a distribution proportional to f_{i-1} restricted to K. Denote the measure of this random sample as μ_{i-1}. Then, for a random sample X drawn from μ_{i-1}, define a corresponding variable Y as

$$Y = \frac{f_i(X)}{f_{i-1}(X)}.$$

Consider the expected value of this random variable Y, which is equal to the desired quantity we want to estimate:

$$E(Y) = \int_K \frac{f_i(x)}{f_{i-1}(x)}\,d\mu_{i-1}(x) = \int_K \frac{f_i(x)}{f_{i-1}(x)} \cdot \frac{f_{i-1}(x)}{\int_K f_{i-1}(x)}\,dx = \frac{\int_K f_i(x)\,dx}{\int_K f_{i-1}(x)\,dx}.$$

Suppose we have k samples $\{X_1, \ldots, X_k\}$. Then, we can estimate the integral ratio as

$$\frac{1}{k}\sum_{j=1}^{k} \frac{f_i(X_j)}{f_{i-1}(X_j)},$$

which converges to $E(Y)$. For the estimation to be efficient, we need that the number of samples k to get within a target relative error is not too large. For instance, if we went immediately from f_0 to the uniform distribution, an exponential number of samples k would be required. Instead, we construct a cooling schedule from f_0 to the uniform distribution while controlling the variance of Y, which results in a small number of "phases" (i.e. integral ratios) and not too many samples per phase. We select each f_i as a Gaussian, where we slowly increase the variance until the Gaussian is essentially the uniform distribution (i.e. the volume). Further details on how to construct such a sequence are given in Sect. 3.2.2.

Hit-and-run(K, f, x): Convex body K, target distribution f, current point x.

- Pick a uniform random line ℓ through the current point x.
- Return a random point on the chord $\ell \cap K$ according to the target distribution f.

Fig. 1 Hit-and-run sampler

There is still the question of how to obtain samples from the target distribution. We use the *hit-and-run* random walk (Fig. 1) to generate *approximate* samples from the target distribution. The implementation details of hit-and-run are given in Sect. 3.4. Given enough steps, hit-and-run will converge to the target distribution. So, hit-and-run can provide sample points we can use to estimate each integral ratio.

But how many steps of hit-and-run are required before the point is from the target distribution? Subsequent points in the random walk may be highly correlated. However, if enough steps of hit-and-run are used, then the current point will "forget" where it started and be an approximately random point from the target distribution. Assuming the current point is somewhat close to the target distribution, it is known $O^*(n^2R^2/r^2)$ hit-and-run steps are required before we are close to target distribution, where $rB_n \subseteq K \subseteq RB_n$; that is, a ball of radius r is contained in K and K is contained in a ball of radius R. The term R/r could unfortunately be very high (e.g. n^{50}) for a general convex body and have a drastic effect on the mixing time. For instance, a long, thin cylinder will have higher mixing time than that of the unit ball. Intuitively, this is because it takes a long time to move from one end of the long cylinder to the other, while it is comparatively easier to move between any two regions in the unit ball.

We can get around this issue by applying a linear transformation T to the convex body K to get a new body $K' = TK$ that is round. Since T is a linear transformation, we have that $\mathsf{vol}(K') = |T| \cdot \mathsf{vol}(K)$ where $|T|$ is the determinant of the matrix corresponding to the transformation. In the case of the long cylinder, while the cylinder could have essentially an arbitrarily high value of R/r, if we instead work with the rounded body, where we shrink the body along the stretched axis and compute $\mathsf{vol}(K')$, then we can efficiently compute the volume. More details of how this transformation is computed are given in Sect. 3.1.

An outline of the algorithm is given in Fig. 2. Throughout this paper, we assume that the origin lies in K. We start by rounding the convex body K into approximate isotropic position, with respect to the uniform distribution over K (Sect. 3.1). We then compute an annealing schedule $\{a_0, \ldots, a_m\}$ such that almost all of the volume of $e^{-a_0\|x\|^2}$ is contained inside K', and $a_m = 0$ (i.e. the m-th phase is the uniform distribution) (Sect. 3.2). Once we have the cooling schedule, we compute the volume by estimating each of the ratios

$$\frac{\int_{K'} f_i(x)\,dx}{\int_{K'} f_{i-1}(x)\,dx},$$

where $f_i(x) = e^{-a_i\|x\|^2}$. The ratio is estimated by approximately sampling from $f_{i-1} \cap K'$ (Sect. 3.4) and then averaging the function ratio over the sample points. To test for convergence of this ratio, we use a sliding window over the last W ratios,

Volume(K, ε): Convex body K, error parameter ε.

- $T =$ **Round**(body: K, steps: $8n^3$), set $K' = T \cdot K$.
- $\{a_0, \ldots, a_m\} =$ **GetAnnealingSchedule**(body: K').
- Set x to be random point from $f_0 \cap K'$, $\varepsilon' = \varepsilon/\sqrt{m}$.
- For $i = 1, \ldots, m$,
  - Set $k = 0, x_0 = x, converged = false, W = 4n^2 + 500$.
  - While $converged = false$,
    - $k = k + 1$.
    - $x_k =$**HitAndRun**(body: K, target distribution: f_{i-1}, current point: x_{k-1}).
    - Set
$$r_k = \frac{1}{k} \sum_{j=1}^{k} \frac{f_i(x_j)}{f_{i-1}(x_j)}.$$
    - Set $W_{max} = \max\{r_{k-W+1}, \ldots, r_k\}$ and $W_{min} = \min\{r_{k-W+1}, \ldots, r_k\}$.
    - If $W_{max} - W_{min} \leq \varepsilon'/2 \cdot W_{max} \rightarrow converged = true$.
  - Set $R_i = r_k, x = x_k$.
- Return $volume = |T| \cdot (\pi/a_0)^{n/2} \cdot R_1 \ldots R_m$.

Fig. 2 Volume algorithm

where if the last W ratios are all within some $C(\varepsilon, m)$ relative error of each other, then we declare convergence (Sect. 3.3). The estimated volume is then the product of the intial integral f_0 over K', the determinant of the linear transformation T that rounded the body, and the ratio estimate R_i for each phase. That is,

$$\begin{aligned}\mathrm{vol}(K) &= \int_K \mathbb{1}\, dx = |T| \cdot \int_{K'} \mathbb{1}\, dx = |T| \cdot \int_{K'} f_0(x)\, dx \cdot \frac{\int_{K'} f_1(x)\, dx}{\int_{K'} f_0(x)\, dx} \cdots \frac{\int_{K'} \mathbb{1}\, dx}{\int_{K'} f_{m-1}(x)\, dx} \\ &= |T| \cdot R_1 R_2 \cdots R_m \cdot \int_{K'} f_0(x)\, dx.\end{aligned}$$

We will now summarize the key optimizations that were made in our implementation to make volume computation practical.

1. For convex bodies that are not sufficiently round, we may need to perform a rounding preprocessing step before estimating the volume. The LV algorithm uses an $O^*(n^4)$ algorithm for rounding, whereas we use a new algorithm which experimentally runs in $O^*(n^3)$ membership calls (see Sect. 3.1).
2. Instead of using a fixed rate of cooling to the uniform distribution, we *adaptively* compute a cooling schedule according to some constraints. This significantly reduces the number of volume phases required (see Sect. 3.2.2).
3. Finally, we only sample from spherical Gaussians, which gives a computationally efficient hit-and-run sampler and experimentally improves the mixing time over an arbitrary Gaussian or logconcave function (see Sect. 3.1).
4. First, we use the empirical distribution of hit-and-run to estimate the volume. In the LV algorithm, hit-and-run is used for some large number of steps, and only a very small fraction of the total steps as sample points, and we experimentally find that using every point from hit-and-run provides a better estimate.

3 Implementation and techniques

In this section we will give a mathematical description of the components of our implementation, and give proofs and/or motivation behind the components. The primary motivation behind our implementation decisions was using as few "hard-coded" constants as possible, and instead try to optimize our runtime for a particular problem instance. For instance, in Sect. 3.2.1, instead of using $a = 2n$ as in [9], we instead use concentration inequalities to binary search for a value of a that is close to optimal for that particular body.

An important question about hit-and-run is how fast it converges to the target distribution. The LV algorithm was designed to minimize the asymptotic worst case. Thus, constants of the type 10^{10} and many log factors are present in the runtime analysis. For instance, it is proven in [15] that $T = 10^{10} n^3 \log 1/\varepsilon$ steps of hit-and-run suffice before we are within distance ε of the target distribution. Combined with the fact that each step of hit-and-run takes $\Omega(n)$ arithmetic operations, this number of steps T is far too large for an actual algorithm, even for very small dimensions. We observe that in practice, one can do much better, but it seems to be difficult to obtain tight bounds on the number of required steps. We instead employ heuristic techniques that try to detect convergence based on the stream of points observed; these techniques experimentally seem to provide a reasonable estimate, but do not give a guarantee of accuracy. For a further discussion, please refer to Sects. 3.3 and 3.5.

In most of the theoretical volume algorithms, there is only an assumption that we have a membership oracle for the convex body. While our experimental evaluations are on explicit polytopes and ellipsoids, our algorithm only needs to compute the intersection of a ray with the body, i.e., the halfspace that first intersects it, and an outer approximation to this would suffice.

3.1 Rounding the body

Problem: Given a convex body K, we would like to find a linear transformation T such that $T \cdot K$ is in approximately isotropic position.

Solution: See Fig. 3. We assume that K is contained a ball of radius R, because we observe that if enough sample points are taken, each rounding should shrink the maximum singular value by a constant factor. If we do not converge within $\log R$ iterations, we know the number of steps t was not sufficient to accurately estimate the singular values, so we restart with $2t$ steps.

The need for such a transformation is exhibited in the following theorem of [15].

Theorem 1 [15] *Let K be a convex body that contains a ball of radius r and is contained in a ball of radius R. Let σ be a starting distribution and let σ^m be the distribution of the current point after m steps of hit-and-run in K. Let $\varepsilon > 0$, and suppose that the density function $d\sigma/d\pi_K$ is bounded by M except on a set S with $\sigma(S) \leq \varepsilon/2$. Then for*

Round(K, t): Convex body: K, rounding steps: t. (**Note**: assume that $B_n \subseteq K \subseteq RB_n$)

- Set $x_0 = 0$, $T = I$, $tries = 0$.
- **Repeat:**
  - $tries = tries + 1$.
  - For $i = 1, \ldots, t$,
    - $x_i =$**HitAndRun**(body: TK, target distribution: $f = 1$, point: x_{i-1}).
  - $(U, S, V^T) = SVD(\{x_1, \ldots, x_r\})$.
  - Set $T = VS^{-1}T$.
- **Until:** $\max(S) \leq 2$ OR $tries > \log R$.
- If $\max(S) \leq 2$:
  - Return T.
- Else:
  - Return **Round**$(K, 2 \cdot t)$.

Fig. 3 Rounding algorithm

$$m > 10^{10} \frac{n^2 R^2}{r^2} \ln \frac{M}{\varepsilon},$$

the total variation distance of σ^m *and* π_K *is less than* ε.

The above mixing time is also shown to be best possible in terms of the quantity R/r, where $rB_n \subseteq K \subseteq RB_n$. This is one measure of how "round" the body K is: if K is a long, thin cylinder, then the ratio R/r can be very high. We can control the ratio R/r by putting the body in approximately isotropic position. We say that a density function is isotropic if its centroid is 0 and its covariance matrix is the identity. That is, for a random variable X drawn from f,

$$\mathsf{E}(X) = 0 \quad \text{and} \quad \mathsf{E}(XX^T) = I.$$

The above condition is equivalent to saying that for every unit vector $v \in \mathbb{R}^n$,

$$\int_{\mathbb{R}^n} (v^T x)^2 f(x)\, dx = 1.$$

We can now consider a notion of approximately isotropic, and say that f is C-isotropic if

$$\frac{1}{C} \leq \int_{\mathbb{R}^n} (v^T x)^2 f(x)\, dx \leq C.$$

We tested two approaches for rounding the body, one of which performed significantly better. One way is to round the body once beforehand with respect to the uniform distribution. The second method is to round the body in each volume phase, as in [9]. That is, put the body in approximate isotropic position with respect to the current distribution. Based on experimental results, while both methods were comparable in terms of runtime for the actual rounding, the first method of rounding the body once beforehand made the volume computation much more efficient. The two primary benefits of rounding the body once beforehand are that fewer volume phases are required

to keep $\mathsf{Var}(Y^2)/\mathsf{E}(Y)^2 \leq 1$ (as in Sect. 3.2), and we can use spherical Gaussians for every phase, which experimentally appear to mix faster. The second method will apply a linear transformation T to both the target distribution f and the body K, which can make the distribution f very skew. For a brief numerical justification, consider a 10-dimensional randomly transformed hypercube. If the cube is rounded once beforehand, then ∼5 phases with ∼10k steps/phase will give ≤20 % accuracy. If the cube is rounded in each phase, then ∼12 phases with ∼250k steps/phase will give ≤20 % accuracy, with the later (i.e. most "skew") phases requiring ∼500k–1000k steps. So, even for small dimensions, we notice rounding the body once beforehand is orders of magnitude faster.

Our goal is to find a linear transformation T such that the TK is 2-isotropic. We do this by obtaining a sequence of points $\{X_1, X_2, \ldots\}$ using hit-and-run with uniform target distribution over K. We then compute the transformation that will put the points from hit-and-run into isotropic position. If enough steps of hit-and-run are taken, then applying that transformation to the body will put the body in approximately isotropic position. Building upon the work of Bourgain [16] and Rudelson [17], Adamczak et al. [18] showed that $O(n)$ random samples suffice to achieve 2-isotropic position. However, the trajectory of points from hit-and-run are not random samples because subsequent points are highly dependent on each other, and to get an adequate "picture" of the distribution, we will need more than $O(n)$ steps of hit-and-run. We observe that taking $8n^3$ steps of hit-and-run seems to suffice for $n \leq 100$ to achieve 2-isotropic position. However, if we note that we are not converging to a sufficiently round body, we restart the rounding process with twice as many steps per rounding. So, this $8n^3$ samples for rounding is not a fixed parameter, but rather a first guess at how many hit-and-run samples are required.

There is one final complexity to the rounding algorithm, which can be seen in the case of a very long box, say a $10^9 \times 1 \times \cdots \times 1$ box. It would require an impractical number of steps of hit-and-run to round this body to 2-isotropic position, but we can do it in multiple phases. We let hit-and-run mix for $8n^3$ steps on this $10^9 \times 1 \times \cdots \times 1$ box, and (very roughly speaking) we may observe a distribution similar to a $20 \times 1 \times \cdots \times 1$ box, so when we compute the transformation T that puts the sequence of points $\{X_1, X_2, \ldots, \ldots X_r\}$ into isotropic position, and apply that transformation to the body K, we are left with, approximately, a $5 \cdot 10^7 \times 1 \times \cdots \times 1$ box. We then repeat until the sequence of points $\{X_1, X_2, \ldots, X_r\}$ is in 2-isotropic position.

To compute the transformation that points the sequence of points $\{X_1, X_2, \ldots, X_r\}$ into isotropic position, we compute the singular value decomposition of the points. That is, for the matrix of points M, we find matrices such that $M = USV'$ such that S is a diagonal matrix that contains the n singular values. Assume the minimum singular value is 1 by rescaling. Then, if any singular value is $s > 2$, we scale the body K along that axis by s. We ignore the smaller singular values for numerical stability.

3.2 Annealing schedule

In this section, we describe how to compute an appropriate sequence of functions $\{f_0, f_1, \ldots, f_m\}$ that are used in the volume algorithm. We first show how to compute

f_0 in Sect. 3.2.1, and then give a recursive approach that computes f_i from f_{i-1} in Sect. 3.2.2.

3.2.1 Selection of starting Gaussian

Problem: Given a convex body K and error parameter $\varepsilon > 0$, select $a_0 \in \mathbb{R}^n$ such that

$$\int_K e^{-a_0\|x\|^2}\,dx \geq (1-\varepsilon)\int_{\mathbb{R}^n} e^{-a_0\|x\|^2}\,dx.$$

Solution outline:

- Consider a random point X from $e^{-a\|x\|^2}$ over $\mathbb{R}^n$, and bound the probability, as a function of a, that $X \notin K$ using Gaussian tail inequalities. Denote this probability as $p(a)$.
- Binary search of the value of a that gives $p(a) = \varepsilon$, and return $a_0 = a$.

As noted above, we will assume $K = P \cap E$, where P is a polyhedron and E is an ellipsoid. Note that we could select a sufficient value of a_0 without much work, say by assuming that K contains the unit ball and then deriving that $a_0 = (n + \sqrt{8n\ln(1/\varepsilon)})/2$ using Lemma 1. However, this could signficantly increase the time to anneal to Gaussian to the uniform distribution (i.e. the volume). We can use our explicit description of the body K to select a value of a_0 so that K contains close to a $(1-\varepsilon)$ fraction of the volume over $\mathbb{R}^n$. First, note the following two Gaussian concentration inequalities.

Lemma 1 *Let X be drawn from a spherical Gaussian in $\mathbb{R}^n$ with mean μ and variance σ^2 along any direction. Then for any $t > 1$,*

$$\mathsf{P}\left(\|X-\mu\|^2 - \sigma^2 n > t\sigma^2\sqrt{n}\right) \leq e^{-t^2/8}.$$

Lemma 2 *Let X be drawn from a one-dimensional Gaussian with variance $\sigma^2 = 1/(2a)$. Then,*

$$\mathsf{P}(X > t) \leq \frac{e^{-at^2}}{2t\sqrt{a\pi}}.$$

To use these bounds, we will first compute the minimum distance from $0 \in K$ to each hyperplane describing P and to the boundary of E. We then apply Lemma 1 to the ellipsoid and Lemma 2 to the polyhedron, and union bound over these probabilities to get a lower bound on the fraction of the Gaussian that lies within the convex body K. That is, for a given a and d the minimum distance from 0 to the surface of the ellipsoid,

$$\begin{aligned}\mathsf{P}(x \notin K) &\le \mathsf{P}(x \notin P) + \mathsf{P}(x \notin E) \le \sum_H \mathsf{P}(x \text{ violates } H) + \mathsf{P}(\|x\| > d) \\ &\le \sum_H \mathsf{P}\left(\|x_H\| > d(0,H)\right) \\ &\quad + e^{-d^2/8} \text{ where } x_H \text{ is the projection of } x \text{ onto the normal of } H \\ &\le \sum_H \frac{e^{-a\cdot d(0,H)^2}}{2d(0,H)\sqrt{a\pi}} + e^{-d^2/8} =: p(a).\end{aligned}$$

3.2.2 Annealing step

Problem: Given a starting function $f_0(x) = e^{-a_0\|x\|^2}$ and a convex body K, construct a sequence of functions that converge to the uniform distribution over K, such that we can efficiently estimate

$$R_i = \frac{\int_K f_i(x)\,dx}{\int_K f_{i-1}(x)\,dx} = \frac{\int_K e^{-a_i\|x\|^2}\,dx}{\int_K e^{-a_{i-1}\|x\|^2}\,dx}.$$

Solution outline: They key idea, which was used in [9,10], is that we would like to control the quantity

$$\frac{\mathsf{Var}(Y)}{\mathsf{E}(Y)^2},$$

where $Y = e^{(a_{i-1}-a_i)X}$ and X is drawn from distribution proportional to $f_{i-1} \cap K$. In [9], it is proven that $\mathsf{E}(Y) = R_i$. Therefore, applying Chebyshev's inequality to Y, having $\mathsf{Var}(Y)/\mathsf{E}(Y)^2 \le 1$ guarantees that only a polynomial number of points are needed to accurately estimate R_i. We estimate $\mathsf{Var}(Y)/\mathsf{E}(Y)^2$ by taking a small number of samples, and stepping as far as we can while keeping $\mathsf{Var}(Y)/\mathsf{E}(Y)^2 \le 1$. We take $f_m = f_i$ once f_i appears to be sufficiently close to the uniform distribution.

The following lemma is essentially shown in [9].

Lemma 3 *Let X be a random point in K with density proportional to $e^{-a_i\|x\|^2}$, $a_{i+1} = a_i(1-1/n)$, $n \ge 4$, and*

$$Y = e^{(a_{i+1}-a_i)\|X\|^2}.$$

Then,

$$\frac{\mathsf{Var}(Y)}{\mathsf{E}(Y)^2} \le \left(\frac{a_i^2}{a_{i+1}(2a_i - a_{i+1})}\right)^{n+1} = \left(1 + \frac{1}{n^2-2n}\right)^{n+1} < 1.$$

Our approach seeks to select the sequence of functions $\{f_i\}$ to approach the uniform distribution as quickly as possible while keeping the variance bounded by a constant. We do this by trying to maximize r, where $a_{i+1} = a_i(1 - 1/n)^r$ so that

$$\frac{\mathsf{Var}(Y^2)}{\mathsf{E}(Y)^2} < 1. \tag{1}$$

By Lemma 3, we know that $r = 1$ now suffices. We will then binary search to get, within a factor of 2, the maximum value of r that satisfies (1), by taking a small number of sample points and observing their variance.

In [19], for the problem of approximating the partition function of a discrete system, they prove that using a constant number of samples suffice to accurately estimate $\mathsf{Var}(Y)/\mathsf{E}(Y)^2$ for their cooling schedule. We empirically observe a similar behavior for computing the cooling schedule for convex bodies, where a small number of hit-and-run samples suffice, roughly $O(n^2)$, to reasonably estimate $\mathsf{Var}(Y)/\mathsf{E}(Y)^2$.

3.3 Convergence of ratio

Problem: Given an error parameter ε' and a stream of dependent random variables $\{X_1, X_2, \ldots\}$, where $Y_k = 1/k \cdot \sum_{i=1}^{k} X_i$ and

$$R = \lim_{k\to\infty} Y_k,$$

determine a point k such that $Y_k \in [(1 - \varepsilon')R, (1 + \varepsilon')R]$.

Solution outline: We use a sliding window of size W and declare convergence once the last W points are all within $\varepsilon'/2$ of each other. That is,

$$\frac{\max_{i:k-W\le i\le k} Y_i - \min_{i:k-W\le i\le k} Y_i}{\max_{i:k-W\le i\le k} Y_i} \le \varepsilon'/2.$$

The goal of our algorithm is to compute a quantity V' such that $V' \in [(1-\varepsilon)V, (1+\varepsilon)V]$ where V is the true volume of our convex body K. Our algorithm is composed of m phases, each of which approximate the ratio of two integrals over K. By a standard argument, if each of m terms have a relative error $\varepsilon' = \varepsilon/\sqrt{m}$ and have unbiased expectation, then the product of the m terms will have relative error ε. Therefore, we assign relative error $\varepsilon' = \varepsilon/\sqrt{m}$ to each of our integral estimators (the R_i's in Fig. 2).

We are given a stream of random points in K $\{X_1, X_2, \ldots\}$ drawn approximately from $f_{i-1} \cap K$, which we then relate to $\{Y_1, Y_2, \ldots\}$ by

$$Y_k = \frac{1}{k}\sum_{j=1}^{k} e^{(a_{i-1}-a_i)X_k}. \tag{2}$$

Denoting the ratio of integrals in a single phases as R, from [9] we know that

$$R = \lim_{k\to\infty} Y_k. \tag{3}$$

Our problem is now: compute a quantity R' with relative error ε' from R. Formally, we want to estimate the following quantity within a ε'-fraction:

$$\frac{\int_K f_i(x)\,dx}{\int_K f_{i-1}(x)\,dx} = \frac{\int_K e^{-a_i\|x\|^2}\,dx}{\int_k e^{-a_{i-1}\|x\|^2}\,dx}.$$

From hit-and-run, we generate random variables $\{X_1, X_2, \ldots\}$, which are points in K from a distribution approximately proportional to f_{i-1}. Then, applying Eqs. (2) and (3), we get a sequence of random variables $\{Y_1, Y_2, \ldots\}$ such that

$$R = \lim_{i\to\infty} Y_i.$$

That is, if we take enough steps of hit-and-run, we will converge to the actual ratio R. Computationally, we would like to determine a point where we are within our target accuracy ε'. If the points generated by hit-and-run were independent, then it would be quite easy to determine how many points are necessary to obtain an accurate estimate. However, these points will be highly dependent, and the best known bounds for the number of steps required are far too high for practical computations. Therefore, we use a sliding window approach that stores the last few values of Y_i, and declares convergence once these last few values are all within, say, $\varepsilon'/2$ relative distance of each other.

The benefit of this approach is that it will quickly detect convergence of $\{y_i\}$ to an ε'-fraction of R. However, the drawback is that we can have false positives; that is, we can declare the sequence has converged too soon and we are not within a ε'-fraction of R. There is a clear relationship between the size of this window and how accurately we will estimate R: the more values we store, the longer we will take before declaring convergence. It is unclear how to obtain a good bound on the probability of failure with relation to the window size, but we choose the size of the window based on experimental results (Sect. 4.2).

3.4 Hit-and-run steps

Throughout the implementation, we use the random walk *hit-and-run* (Fig. 1) for generating approximately random points from spherical Gaussians restricted to convex bodies. A single hit-and-run sample requires computing the intersection of a line with our convex body K and then sampling along that chord according to our target distribution. This section can be viewed as the "inner-most" section of our algorithm, so the efficiency of a hit-and-run step significantly affects the total runtime of our volume algorithm. For polytopes and ellipsoids, we describe here how to efficiently compute the chord. For other convex bodies, such as zonotopes in Sect. 4.4, the chord computation is less efficient and has a drastic effect on the overall runtime. Once the chord is computed, we then need to efficiently sample from the chord according to a Gaussian distribution. We now discuss how each of the two steps are implemented in our algorithm.

3.4.1 Chord computation

Problem: Given a random direction $u \in \mathbb{R}^n$ and a point $x \in K$, determine the two endpoints (x^-, x^+) of the line $x + \alpha u, \alpha \in \mathbb{R}$ intersected with the convex body K.

Solution outline: One simple approach is to binary search for the positive α that intersects K (likewise for the negative α value), which only relies on having a membership oracle. However, if we know

$$K = P \cap E = \{x : Ax \leq b\} \cap \{x : (x - v)^T Q^{-1}(x - v) \leq 1\},$$

we can use this description to get a more efficient algorithm by explicitly computing the intersection points of $x + \alpha u$ with both P and E, and taking the closest intersection points in each direction as (x^-, x^+).

For the polyhedron $P = \{x : Ax \leq b\}$, we can compute the distance to each hyperplane of P (row,entry pair of A, b) in the $+u$ and $-u$ direction, and take the minimum distance for each direction. Letting the rows of A and entries of b be labeled $1, \ldots, r$, the values (α_P^-, α_P^+) are given by the following:

$$\alpha_P^- = \max_{1 \leq i \leq r : A_i \cdot u \leq 0} \frac{b_i - A_i \cdot x}{A_i \cdot u} \qquad \alpha_P^+ = \min_{1 \leq i \leq r : A_i \cdot u > 0} \frac{b_i - A_i \cdot x}{A_i \cdot u}. \tag{4}$$

For the ellipsoid $E = \{x : (x - v)^T Q^{-1}(x - v) \leq 1\}$, we would again like to determine the two points that intersect E along the line $x + \alpha_E u$. To simplify notation, we will assume $v = 0$ and shift x accordingly. We then would like to solve the equation $(x + \alpha_E)^T Q^{-1}(x + \alpha_E u) = 1$, which is quadratic in α_E and yields a solution pair (α_E^-, α_E^+):

$$u^T Q^{-1} u \cdot \alpha_E^2 + \left(u^T Q^{-1} x + x^T Q^{-1} u\right) \cdot \alpha_E + x^T Q^{-1} x - 1 = 0. \tag{5}$$

Combining the values from 4 and 5, the chord for the convex body is then $(x^-, x^+) = (x + \max\{\alpha_P^-, \alpha_E^-\}u, x + \min\{\alpha_P^+, \alpha_E^+\}u)$. Note that all of the values may not exist, for instance if the polyhedron is unbounded, but if a particular $\alpha_{P/E}^{+/-}$ does not exist, we simply ignore it.

3.4.2 Sampling from chord

Problem: Given a description of a chord as $\ell = (u, v) \subset \mathbb{R}^n$, generate a sample according to a density proportional to $f(x) = e^{-a\|x\|^2}$ restricted to ℓ.

Solution outline:

- If $\|u - v\| \geq 2/\sqrt{2a} \rightarrow$ return *GaussianSample*(chord: (u, v), target distribution: f).
- Else $\rightarrow$ return *UniformSample*(chord: (u, v), target distribution: f).

GaussianSample: Since this density is spherical Gaussian, its restriction to the 1-dimensional chord will be a 1-D Gaussian with variance $1/(2a)$. The projection of 0 onto the line extending ℓ will be the mean of this Gaussian, call it μ_Z. Note we can map the points u and v to this 1-dimensional Gaussian in terms of their distance and direction from μ_Z. We can then sample $Z \sim \mathcal{N}(0, 1/(2a))$ until the point $\mu_Z + Z \cdot \overrightarrow{uv}$ lies on $\ell = (u, v)$, where $\overrightarrow{uv}$ is a unit vector in the direction from u to v. Then, return $\mu_Z + Z \cdot \overrightarrow{uv}$.

UniformSample: Another way to sample from this chord is by simple rejection sampling. We enclose the distribution by a rectangle whose width is $\|u - v\|$ and height is $\max_{x \in (u,v)} e^{-a\|x\|^2}$. We then sample uniformly from the box and reject the point if it lies outside the region enclosed by $e^{-a\|x\|^2}$.

To sample efficiently from the chord, we use a combination of these two methods. Note that the two methods perform well in different cases. The first method will perform better when the Gaussian is sharp and mostly contained inside the chord, whereas the second method will perform better when the Gaussian is flat. In the first method, the success probability is the measure of the Gaussian inside (u, v), and for the second method, the success probability is the ratio of the average value to the maximum value of $e^{-a\|x\|^2}$ in (u, v). We can compute these values, and select the approach that has the highest acceptance probability. However, computing the exact values is somewhat expensive compared to actually generating the random samples. We achieve better performance by a rough approximation which performs well in practice: if the chord length is more than 2 standard deviations long, then we use the Gaussian method. In practice, using this rough approximation for the Gaussian and uniform sampler, we observe an average failure probability of at most 20 % over all the test bodies in Sect. 4.

3.5 Sampling and multiple threads

An unavoidable problem with hit-and-run is that subsequent points of a trajectory are very dependent. In recent volume computation algorithms [9,10,20], this dependence was handled by allowing the trajectory to mix for some $c(\varepsilon)$ steps before collecting a new "sample point" to obtain ε-independence between subsequent points. The drawback of this approach is that $c(\varepsilon)$ could be very large. Another approach is to use every point along the chain, with the hope that the greater number of sample points outweighs the dependency. There is theoretical evidence in favor of this approach for different applications of Markov chains [21], and we observe it to be true experimentally for estimating volume with hit-and-run. We then test for convergence by the heuristic approach outlined in Sect. 3.3.

Another natural question is if we can obtain a better estimate of the volume ratio in a single phase by using multiple trajectories of our Markov chain. That is, concurrently run t independent threads of hit-and-run in a circular queue–step thread 1, step thread 2, …, step thread t, step thread 1, etc. The following intuition gives some insight into how for a fixed number of total steps s (i.e. s/t steps per thread), the number of threads could affect our answer. Running t threads can be thought of as running $2t$

threads, where thread $2i$ starts from the last point of thread $2i-1$. The last point of this thread will be fairly mixed as opposed to whatever point thread $2i$ would start with if we were actually using $2t$ threads. Therefore, using a smaller number of threads decreases the total mixing time, but a smaller number of threads increases the total dependence between the sample points. So increasing the number of threads gives a trade-off between dependence and mixing time. In Sect. 4.6, we see that mixing times seems to have a more significant effect than the dependence, and we should therefore use a very small number of threads. Based on the experiments, we use 5 threads, a constant number independent of dimension, throughout our implementation.

4 Computational results

In this section, we will present numerical results of our algorithm over a test set of convex bodies, which are described in Sect. 1.1.

4.1 Complexity

The plots in Figs. 4 and 5 show how our program scales with dimension for the n-dimensional *Cube*. We give results over two different hardware configurations. We plot the number of hit-and-run steps and total computation time, for $n = 5, 10, \ldots, 95, 100$. For each value of n, 5 trials were performed. The figures suggest that the volume of the cube can be computed in $O^*(n^2)$ membership oracle calls. For the time complexity, note that each step of hit-and-run requires a chord computation and sampling along that chord. The chord computation requires $\Theta(n^2)$ arithmetic operations, while the sampling steps requires $O(1)$ time. So, we would expect that the time grows as $n^2 \times n^2 = O(n^4)$ based on the number of oracle calls, but the hidden constant in the sampler is quite high. For these small values of n, we instead observe the time to grow roughly like $n^{2.5}$. We note that the current best theoretical complexity is $O^*(n^3)$ oracle calls and $O^*(n^5)$ arithmetic operations [11].

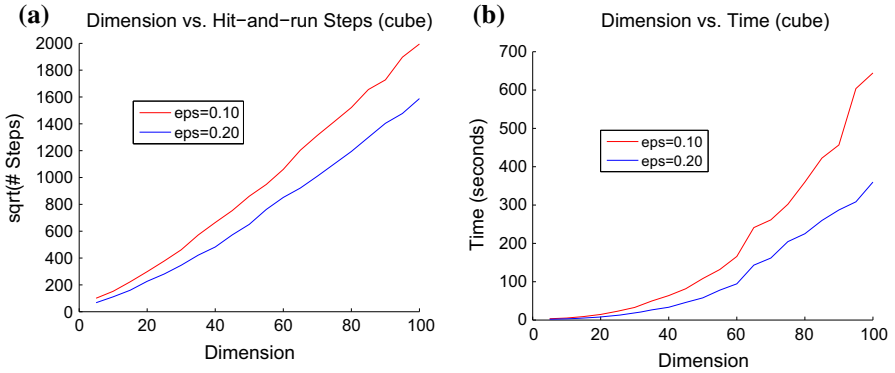

Fig. 4 The above data was computed on a 64-bit Windows 8 machine with a i7-3630QM (8 threads, 2.40 GHz) processor and 8GB RAM using MATLAB R2013a. **a** In the above graph, the correlation coefficient (R-value) is 0.997 for both lines. **b** The above runtimes grow at roughly $n^{2.5}$

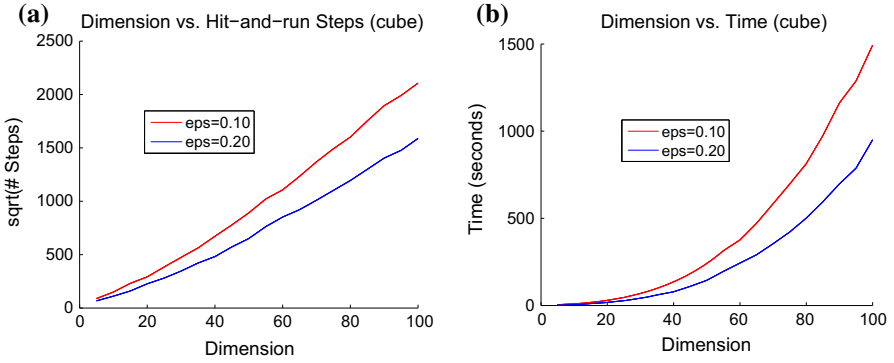

Fig. 5 The above data was computed on a 64-bit CentOs 5.6 machine with a X5570 (8 threads, 2.93 GHz) processor and 48 GB RAM using MATLAB R2014b. **a** As expected, this plot is nearly identical to Fig. 4a and the number of steps taken is consistent across a different hardware configuration. **b** The computation time appears to be roughly double that of Fig. 4b on a different machine, but grows at approximately the same rate

4.2 Accuracy and times

The Tables 1, 2, 3 and 4 in this section show the numerical results of our implementation on our set of 5 bodies, in 10, 50, and 100 dimensions. The tables show the mean and standard deviation over a certain number of trials, where the standard deviation is reported as a percentage of the mean. Also included are the average time for one run of the algorithm and the average number of hit-and-run steps that were taken. The total number of hit-and-run steps is equivalent, up to a logarithmic factor, to the number of membership oracle calls, and the best known theoretical algorithm accurately estimates

Table 1 10-Dimensional results

Body	Actual vol	Mean	$\frac{\text{Std dev}}{\text{Mean}}$	Time (s)	# Steps
Cube	1.02×10^{3}	9.91×10^{2}	0.165	6.48×10^{0}	1.22×10^{4}
Isotropic Simplex	1.47×10^{-6}	1.34×10^{-6}	0.249	9.10×10^{0}	1.92×10^{4}
Half-Ball	1.28×10^{0}	1.22×10^{0}	0.164	1.09×10^{1}	1.73×10^{4}

Numerical results for 10-dimensional bodies using 1000 trials with error parameter $\varepsilon = 0.20$

Table 2 50-Dimensional results

Body	Actual vol	Mean	$\frac{\text{Std dev}}{\text{Mean}}$	Time (s)	# Steps
Cube	1.12×10^{15}	1.09×10^{15}	0.189	1.11×10^{2}	4.59×10^{5}
Isotropic Simplex	3.85×10^{-64}	3.21×10^{-64}	0.324	2.02×10^{2}	6.62×10^{5}
Half-Ball	8.65×10^{-14}	8.36×10^{-14}	0.159	1.20×10^{2}	3.95×10^{5}

Numerical results for 50-dimensional bodies using 200 trials with error parameter $\varepsilon = 0.20$

Table 3 100-Dimensional results

Body	Actual vol	Mean	$\frac{\text{Std dev}}{\text{Mean}}$	Time (s)	# Steps
Cube	1.26×10^{30}	1.23×10^{30}	0.172	4.68×10^{2}	2.12×10^{6}
Isotropic Simplex	1.77×10^{-157}	1.41×10^{-157}	0.478	8.30×10^{2}	3.81×10^{6}
Half-Ball	1.18×10^{-40}	1.16×10^{-40}	0.202	4.56×10^{2}	1.72×10^{6}

Numerical results for 100-dimensional bodies using 100 trials with error parameter $\varepsilon = 0.20$

Table 4 10-Dimensional results

Body	Actual vol	Mean	$\frac{\text{Std dev}}{\text{Mean}}$	Time (s)	# Steps
Transformed Cube	5.20×10^{0}	4.94×10^{0}	0.192	9.33×10^{0}	1.73×10^{4} 2.72×10^{4}
Ellipsoid	2.55×10^{2}	2.46×10^{2}	0.130	2.85×10^{1}	1.01×10^{4} 2.40×10^{4}
Standard Simplex	2.76×10^{-07}	2.46×10^{-07}	0.312	8.60×10^{0}	2.40×10^{4} 3.13×10^{4}

Numerical results for 10-dimensional bodies using 1000 trials with error parameter $\varepsilon = 0.20$

volume in $O^*(n^4)$ membership oracle calls [9]. The majority of the experimental data here was computed on Georgia Tech's Jinx computing cluster, and the computation times given were computed with a i7-3630QM (quad-core, 2.40 GHz) processor and 8GB 1600 MHz RAM using the MATLAB R2013a profiler. Each time given was averaged over 5 trials.

4.2.1 Volume, no rounding

In this section, we will give experimental results for convex bodies without the rounding step. The rounding step is an expensive preprocessing step that ensures the efficiency of our volume algorithm, so these bodies will run much faster than bodies that need to be rounded. We observe that the *Cube* and *Half-Ball* perform noticeably better than the *Isotropic Simplex*. There is theoretical motivation for a simplex being a "bad" case for sampling and volume computation. For one, the value of R for an isotropic body K that contains the unit ball, where R is the minimum radius ball that contains K, is maximized when K is a simplex [22], where R is essentially n. Most of the mass of the *Isotropic Simplex* will lie near its $n + 1$ corners, which are sharper than the vertices of the *Cube*. In a rough sense, sharp corners perform poorly with hit-and-run; it will take longer to visit a sharp corner than, say, a region in the middle of the body. But once a sharp corner is visited, hit-and-run will take small steps that stay near the corner before eventually drifiting away. Since most of the volume of a simplex is contained near its corners, it may take more steps of hit-and-run to get an accurate estimate of its volume.

Table 5 50-Dimensional results

Body	Actual vol	Mean	$\frac{\text{Std dev}}{\text{Mean}}$	Time (s)	# Steps
Transformed Cube	5.57×10^{-17}	5.24×10^{-17}	0.286	1.12×10^{3}	4.93×10^{5} 6.19×10^{6}
Ellipsoid	1.73×10^{-11}	1.69×10^{-11}	0.096	4.75×10^{2}	1.52×10^{5} 2.30×10^{6}
Standard Simplex	3.29×10^{-65}	2.84×10^{-65}	0.691	8.21×10^{2}	8.16×10^{5} 6.39×10^{6}

Numerical results for 50-dimensional bodies using 200 trials with error parameter $\varepsilon = 0.20$

4.2.2 Volume with rounding

The following bodies go through a rounding preprocessing step to put the body in approximate isotropic position. To illustrate the rounding complexity, the "# Steps" column will now have two lines: the first line will be "# Volume Steps", and the second line will be "# Rounding Steps". Note the substantial decrease in efficiency for these bodies, as compared to the round bodies in Sect. 4.2.1, where this increase in both runtime and membership oracle calls is due to the rounding preprocessing step, which begins to take a prohibitively long time for bodies much higher than 50 dimensions.

After the rounding phase, the efficiency of the volume computation is comparable, but slightly worse, than the corresponding bodies in the previous section. For instance, the 50-dimensional *Transformed Cube* required around 10 % more hit-and-run steps than the 50-dimensional *Cube* and had a higher standard deviation in its computed volume; we observe a similar relationship between the 50-dimensional simplices. This behavior is in line with what we expect, because the *Cube* and *Isotropic Simplex* are in isotropic position, whereas the rounding phase will only achieve *approximate* isotropic position for the *Transformed Cube* and *Standard Simplex*.

4.2.3 Improving accuracy

In the above tables, the simplices had the highest standard deviation relative to the mean volume. The standard deviation divided by the mean will give a rough idea of how accurate the reported volume is. The worst case was the 50-dimensional standard simplex, where the observed standard deviation divided by the mean is 0.691, much higher than the target relative error of 0.20. For the results in Table 5, only 70 of 200 trials were within 20 % of the actual volume. We can improve this accuracy by averaging the result over multiple trials. The average volume over 200 trials was within 0.20 relative error of the actual volume for the 50-dimensional standard simplex; this observation also holds true for all other test bodies reported in this section. The simplex is conjectured to be the body with the smallest isoperimetric coefficient, i.e., where the random walk mixes slowest.

There are two general ways to do increase the probability of an accurate answer. The first is to average over multiple trials. For the 50-dimensional standard simplex, if

we group the trials into pairs and average the volume of the two trials, the probability that we are within our target relative error is now 0.49, as opposed to 0.35 when just considering a single trial. If we group the trials into groups of 10, then this probability improves to 0.65. Another heuristic to obtain a more accurate volume estimate is to lower the error parameter provided to the volume algorithm, until the observed standard deviation divided by mean is approximately the desired relative error.

The main reason we did not incorporate either of these methods in our implementation is that for all other test bodies it seems superfluous and results in a constant factor increase in the runtimes.

4.3 Birkhoff polytope

An interesting application of our volume algorithm was for computing the volume of the Birkhoff polytope. The nth Birkhoff polytope B_n is the polytope of all $n \times n$ doubly stochastic matrices; equivalently, it is the perfect matching polytope of the complete bipartite graph $K_{n,n}$. This polytope has a number of nice combinatorial properties, with one important question being its volume [23–27]. There has been previous work on computing the volume of B_n for small values of n, where specialized algorithms were developed for this specific polytope [28,29]. The most recent work of [28] computed B_n for $n = 10$ through a distributed algorithm, with a total computation time of 17 years at 1GHz. To our knowledge, the values for $n > 10$ have not yet been obtained, as the current approaches are too computationally expensive for any value of n higher than 10. We show here that if you relax the requirement for an exact answer, our volume algorithm can obtain a reasonable estimate for the value for $n = 15$ within a few hours.

We now give a more complete description of B_n using its formulation as all $n \times n$ doubly stochastic matrices. Define n^2 variables X_{ij} for $i, j \in \{1, \ldots n\}$ as the values assigned to the corresponding entries of a doubly stochastic matrix. The following equations then define the polytope B_n:

$$\sum_{i=1}^{n} X_{ij} = 1, \quad j \in \{1, \ldots, n\} \tag{6}$$

$$\sum_{j=1}^{n} X_{ij} = 1, \quad i \in \{1, \ldots, n\} \tag{7}$$

$$X_{ij} \geq 0, \quad i, j \in \{1, \ldots, n\} \tag{8}$$

Note that while $B_n \subset \mathbb{R}^{n^2}$, the dimension of the polytope is lower: $\dim(B_n) = (n-1)^2$. We have $2n$ equality constraints above on n^2 variables, but one is redundant. Therefore, to compute the volume of B_n, we compute the volume of the $n^2 - (2n-1) = (n-1)^2$ dimensional subspace spanned by Eqs. (6)–(7), restricted to the positive orthant by Eq. (8).

One aspect of the Birkhoff polytope that benefits our approach is that the polytope is already round. In Sect. 4.2.2, we see that the rounding preprocessing step for our convex body is quite expensive and dominates computing its volume. By avoiding this

Table 6 Birkhoff polytope results

n	Actual vol	Mean	$\frac{\text{Std dev}}{\text{Mean}}$	Time (s)	# Steps
2	$2.00 \times 10^{+00}$	$2.01 \times 10^{+00}$	0.044	4.80×10^{-1}	8.87×10^{2}
3	$1.12 \times 10^{+00}$	$1.14 \times 10^{+00}$	0.146	1.45×10^{0}	4.41×10^{3}
4	6.21×10^{-02}	5.79×10^{-02}	0.209	3.68×10^{0}	1.33×10^{4}
5	1.41×10^{-04}	1.32×10^{-04}	0.233	8.00×10^{0}	4.09×10^{4}
6	7.35×10^{-09}	6.67×10^{-09}	0.211	2.08×10^{1}	1.12×10^{5}
7	5.64×10^{-15}	5.34×10^{-15}	0.240	4.66×10^{1}	2.82×10^{5}
8	4.42×10^{-23}	3.92×10^{-23}	0.290	1.18×10^{2}	6.27×10^{5}
9	2.60×10^{-33}	2.36×10^{-33}	0.289	2.33×10^{2}	1.26×10^{6}
10	8.78×10^{-46}	8.04×10^{-46}	0.261	5.39×10^{2}	2.35×10^{6}
11	???	1.26×10^{-60}	0.238	8.62×10^{2}	4.07×10^{6}
12	???	7.17×10^{-78}	0.254	1.37×10^{3}	6.49×10^{6}
13	???	1.21×10^{-97}	0.281	2.15×10^{3}	1.04×10^{7}
14	???	5.52×10^{-120}	0.292	3.51×10^{3}	1.59×10^{7}
15	???	4.97×10^{-145}	0.277	5.67×10^{3}	2.35×10^{7}

Numerical results for Birkhoff polytopes using 100 trials with error parameter $\varepsilon = 0.20$. Computational times were on a 64-bit Windows 8 machine with a i7-3630QM (8 threads, 2.40 GHz) processor and 8GB RAM using MATLAB R2013a

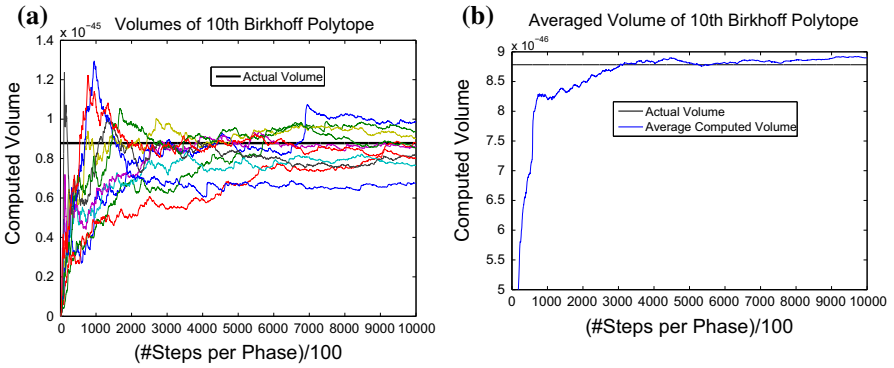

Fig. 6 **a** A plot of 10 independents trials for B_{10} and the computed volume of each trial as a function of the number of steps per volume phase. **b** A plot of the average computed volume over 100 independent trials for B_{10}. We see that taking the average over trials provides a more accurate estimate

rounding step for the Birkhoff polytope, we can more easily go to high dimensions. Here we go up to $n = 15$, i.e. a 196-dimensional polytope, and each run completes within 5 h. If desired, it should be computationally feasible to get accurate estimates for $n = 20$ or $n = 25$ within days or weeks, respectively. In Table 6, we show the results of 100 independent runs of our volume algorithm for $B_2, \ldots, B_{15}$ with $\varepsilon = 0.20$. In Fig. 6, we show how our computed volume converges as the number of steps per phase, over multiple trials. In Fig. 7, we show a similar plot for $n = 15$, where the volume is unknown. The plot indicates that the volume of B_{15} should be close to $5.5 \cdot 10^{-145}$.

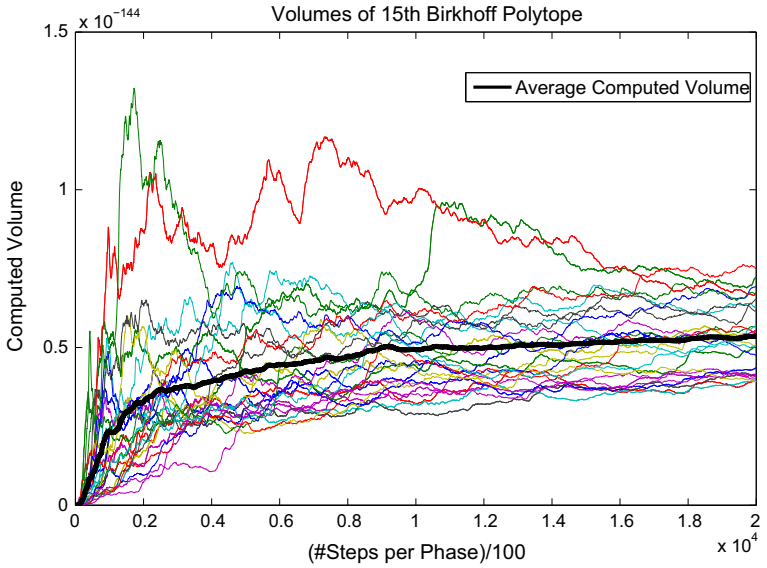

Fig. 7 A plot of 25 independents trials for B_{15} and the computed volume of each trial as a function of the number of steps per volume phase. The average of the trials is included

4.4 Zonotopes

Another application where our algorithm can efficiently provide an accurate estimate, where the exact volume is difficult to compute, is with zonotopes. A zonotope is defined as the Minkowski sum of a set of line segments. Let $\{v_i | 1 \leq i \leq m\}$ be a set of line segments, where each v_i lives in $\mathbb{R}^n$. The zonotope Z is then defined as

$$Z = \left\{ \sum_{i=1}^{m} \lambda_i v_i | 0 \leq \lambda_i \leq 1 \right\}.$$

To compute its exact n-dimensional volume deterministically, consider a set of n line segments and compute the volume of the parallelopiped formed by these n line segments. Then, sum over all $\binom{m}{n}$ sets of n segments to obtain the volume for the zonotope Z. The volume of each parallelopiped is the absolute value of the determinant of the matrix of the n line segments. However, for a fixed dimension n, the runtime of this algorithm will scale as $O(m^n)$. Even for a somewhat small dimension $n = 10$, when the number of line segments m grows to, say, 50 or 100, this approach quickly becomes inefficient. For a further discussion of zonotope volume, we refer the reader to [30], where it is shown that exactly computing the volume of a zonotope is #P-Hard.

The description of a zonotope is different than a polytope or ellipsoid. To use our volume algorithm, we need to compute the chord in one step of hit-and-run. That is, given a current point $x \in Z$ and a direction $u \in \mathbb{R}^n$, determine the points at which the line $x + \alpha u$, $\alpha \in \mathbb{R}$, intersects the boundary of Z. We can formulate this as a linear program:

Table 7 Zonotope results

m	Exact volume	Exact time (s)	Approx volume	Approx time	# Steps
20	4.93×10^3	1.04	4.10×10^3	3.01×10^3	9.58×10^4
50	1.33×10^8	5.78×10^4	1.17×10^8	3.73×10^3	1.04×10^5
100	???	9.74×10^8	1.62×10^{11}	4.90×10^3	1.25×10^5

Numerical results for zonotopes in $\mathbb{R}^{10}$ with 20, 50, and 100 line segments. For each value of m, the data is averaged over 10 independent trials. Computational times were on a 64-bit Windows 8 machine with a i7-3630QM (8 threads, 2.40 GHz) processor and 8GB RAM using MATLAB R2013a

$$\min \quad \alpha \tag{9}$$

$$\text{s.t.}$$

$$x = \sum_{i=1}^{m} \lambda_i v_i - \alpha u \tag{10}$$

$$0 \leq \lambda_i \leq 1 \qquad 1 \leq i \leq m \tag{11}$$

Solving (9)–(11) will produce one point at which the line intersects the boundary (i.e. one endpoint of the hit-and-run chord). To find the other, we simply minimize $-\alpha$ in the objective function (9). The chord computation therefore requires solving two linear programs. We note that this is much less efficient than the corresponding algorithms for polytopes or ellipsoids, and in the below computational results, the time to solve these linear programs heavily dominates the runtime.

To generate a zonotope with m line segments, we start with the generators $e_1, \ldots, e_n$ for the unit cube and then add $m - n$ random unit vectors. The zonotope constructed in this fashion may not necessarily be round, so we perform a rounding preprocessing step before computing our estimate for the volume. In Table 7, we give computational results for computing volumes of zonotopes. The dimension $n = 10$ is fixed, and we increase the number of line segments m. For $m = 20$, we see that the exact algorithm is much more efficient than our algorithm. However, for $m = 50$, the exact algorithm's runtime drastically increases, while our approximation takes roughly the same amount of time to compute. The exact answer for $m = 50$ took approximately 16 computation hours. The data for $m = 100$ was also included for our algorithm, along with an estimated time for the exact answer, which is approximately 31 years.

4.5 Convergence

Here we will give experimental results for the rate at which the relative error decreases as a function of the number of hit-and-run steps per volume phase. For a fixed body K and dimension n, we expect that $O(\varepsilon^{-2})$ hit-and-run steps are required to obtain relative error ε [9]. We observe this to be true in expectation in Fig. 8 for the 20-dimensional *Cube*, where we plot the following quantity as a function of the number of steps per volume phase t:

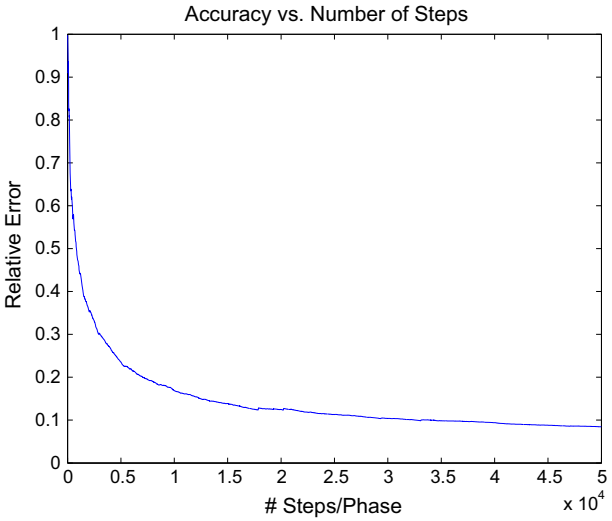

Fig. 8 In the above plot, the relative error ε decreases as $O(\varepsilon^{-2})$ as a function of the number of steps per phase. The data was averaged over 1000 trials for the 20 dimensional cube

$$\varepsilon(t) = \mathsf{E}\left(\frac{|V(t) - 2^{20}|}{2^{20}}\right),$$

where $V(t)$ is the observed volume after t hit-and-run steps per phase. That is, $\varepsilon(t)$ is the expected error after taking t steps per phase, and we estimate $\varepsilon(t)$ by averaging over 1000 independent trials.

We also observe an additional property of how our estimate converges to the true value, where the estimated value is monotonically increasing, in expectation, as a function of the number of hit-and-run steps. This behavior can be best viewed in Sect. 4.6, Fig. 9 where a large number of threads estimates the behavior of our estimated volume in expectation, and we see that a large number of threads clearly exhibits the monotonicity property for the 20-dimensional *Cube*. Also, all of the reported volumes in Sect. 4.2 are lower than the actual volume, which suggests we approach the true volume from below.

4.6 Number of threads

In this section, we give experimental results on the 20-dimensional *Cube* for how the number of threads, as discussed in Sect. 3.5, affects the accuracy of the volume estimate. Figure 9 illustrates the effect of mixing time on the volume estimate. We see that for a greater number of threads, it takes longer to approach the true volume, but it does so more "smoothly" because nearby samples are less dependent. Figure 9 also suggests that the computed ratio approaches the true ratio monotonically, in expectation, but this fact remains to be proven. In Fig. 10, we see the computed volume after 500,000 steps per volume phase. It suggests that using a smaller number

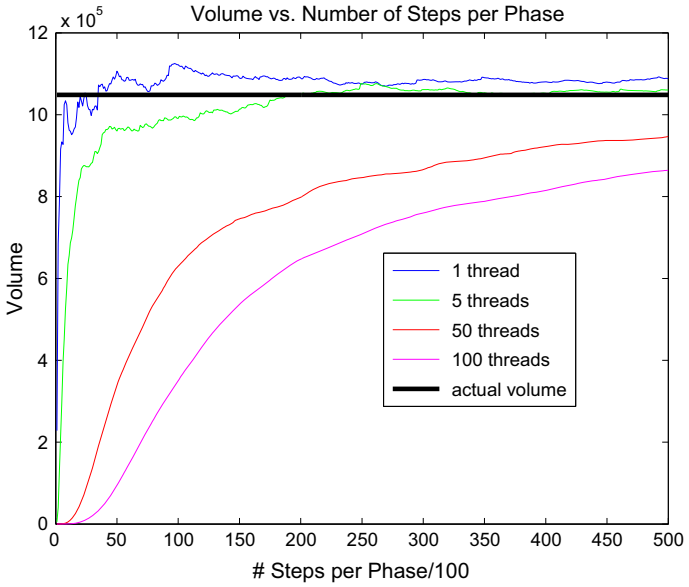

Fig. 9 In the above graph, we see how the volume estimate approaches the true volume, for varying numbers of threads. The volumes are averaged over 50 trials

of threads will provide the most accurate estimate for a fixed number of steps. Using 1 thread will provide the fastest mixing time, but at the cost of high dependence between subsequent sample points; 5 to 10 threads seems to provide an appropriate balance between mixing time and dependence.

5 Concluding remarks

1. Our implementation is available from the MATLAB File Exchange [13] and also from our webpage [31].
2. In the implementation, we use an adaptive cooling schedule; experimentally it performs significantly faster than a fixed cooling schedule. In subsequent work [11], we obtained an $O^*(n^3)$ volume algorithm with an accelerated schedule. Roughly speaking, we start with a Gaussian with $\sigma_0^2 = 1/n$ and cool according to the schedule:

$$\sigma_i^2 = \begin{cases} \sigma_{i-1}^2 \cdot \left(1 + \frac{1}{\sqrt{n}}\right) & \text{if } \sigma_{i-1}^2 = O(1) \\ \sigma_{i-1}^2 \cdot \left(1 + \frac{\sigma_{i-1}}{\sqrt{n}}\right) & \text{otherwise} \end{cases}$$

Note that this schedule does not depend on the body. In practice, we observe an even faster convergence of $O^*(n^2)$ steps for the convex bodies in our benchmark (for example, in Figs. 4, 5).

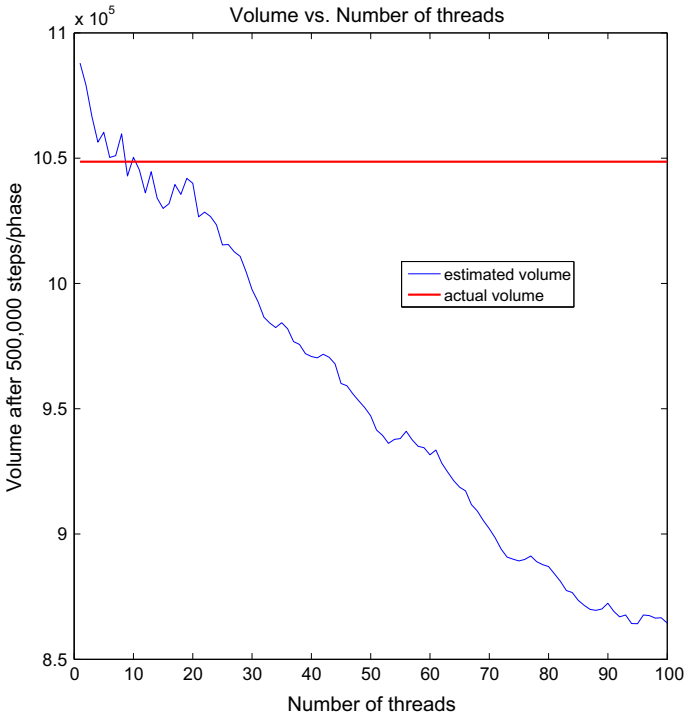

Fig. 10 This is a snapshot of the computed volumes after 500,000 steps per phase, for # threads= $1, \ldots, 100$. The volumes are averaged over 50 trials

3. Hit-and-run was one choice for the random walk to sample from the body. We could also use the well-studied ball walk [10,20]; in each step of the ball walk, we pick a random point y in a ball centered at the current point, and move to y if $y \in K$. In brief implementation tests, it seemed to perform comparable to hit-and-run if the average local conductance (i.e. the probability that $y \in K$) is kept around $1/2$.
4. For our implementation, the volume computation experimentally runs in $O^*(n^2)$ oracle calls, and the rounding algorithm runs in $O^*(n^3)$ oracle calls. The current best algorithm for volume computation is $O^*(n^3)$ [11] and for rounding is $O^*(n^4)$ [9]. So our empirical results suggest algorithms with better worst-case bounds are possible.
5. One area for improvement in this algorithm is the sliding window convergence test (Sect. 3.3). The size of the window is based on experimental results to perform well for $n \leq 100$ and $0.10 \leq \varepsilon \leq 0.20$. It would be preferable to have a more robust way to determine convergence of the volume estimator in a single phase that will perform accurately and efficiently for all values of n and ε.
6. We note that we can naively parallelize both the volume and rounding algorithms by assigning each thread of hit-and-run to a different processor. Additionally, for the volume algorithm, we could assign each volume phase (Sects. 3.2.2, 3.3) to a separate processor, provided we first obtain a warm start for each phase.

Several intriguing theoretical questions arise from our implementation:

1. *Convergence tests:* Our implementation uses empirical convergence tests to test that a sample estimates a ratio within a desired error. How can such methods be made rigorous?
2. *Sampling:* Instead of using a random walk to generate approximately independent points and then using these points to estimate the volume as in the theoretical algorithms, is it asymptotically more efficient, in terms of the total number of hit-and-run steps, to use the entire sequence of points visited during the walk to estimate the volume?
3. *Monotonic convergence:* When estimating the integral ratio of two functions f, g over a convex body K (i.e. $\int_K f(x)dx / \int_K g(x)dx$) with a sequence of points from the empirical distribution of hit-and-run, under what conditions on f, g and the starting distribution, does the estimated value approach the true value monotonically from below in expectation? We observed such monotonicity when f, g are spherical Gaussians and f has higher variance than g (Fig. 9).

Acknowledgments B. Cousins was supported in part by a National Science Foundation Graduate Research Fellowship. Both authors were supported in part by National Science Foundation award CCF-1217793.

References

1. Genz, A., Bretz, F.: Computation of Multivariate Normal and t Probabilities. Springer, New York (2009)
2. Iyengar, S.: Evaluation of normal probabilities of symmetric regions. SIAM J. Sci. Stat. Comput. **9**, 812–837 (1988)
3. Kannan, R., Li, G.: Sampling according to the multivariate normal density. In: FOCS '96: Proceedings of the 37th Annual Symposium on Foundations of Computer Science, Washington, DC, USA, p. 204. IEEE Computer Society (1996)
4. Martynov, G.: Evaluation of the normal distribution function. J. Soviet Math **77**, 1857–1875 (1980)
5. Schellenberger, J., Palsson, B.: Use of randomized sampling for analysis of metabolic networks. J. Biol. Chem. **284**(9), 5457–5461 (2009)
6. Somerville, P.N.: Numerical computation of multivariate normal and multivariate-t probabilities over convex regions. J. Comput. Graph. Stat. **7**, 529–545 (1998)
7. Dyer, M.E., Frieze, A.M., Kannan, R.: A random polynomial time algorithm for approximating the volume of convex bodies. In: STOC, pp. 375–381 (1989)
8. Dyer, M.E., Frieze, A.M., Kannan, R.: A random polynomial-time algorithm for approximating the volume of convex bodies. J. ACM **38**(1), 1–17 (1991)
9. Lovász, L., Vempala, S.: Simulated annealing in convex bodies and an $O^*(n^4)$ volume algorithm. J. Comput. Syst. Sci. **72**(2), 392–417 (2006)
10. Cousins, B., Vempala, S.: A cubic algorithm for computing Gaussian volume. In: SODA, pp. 1215–1228 (2014)
11. Cousins, B., Vempala, S.: Bypassing KLS: Gaussian cooling and an $O^*(n^3)$ volume algorithm. In: STOC, pp. 539–548 (2015)
12. Lovász, L., Deák, I.: Computational results of an $o^*(n^4)$ volume algorithm. Eur. J. Oper. Res. **216** (2012)
13. Cousins, B., Vempala, S.: Volume computation of convex bodies. MATLAB File Exchange. http://www.mathworks.com/matlabcentral/fileexchange/43596-volume-computation-of-convex-bodies (2013)
14. Emiris, I., Fisikopoulos, V.: Efficient random-walk methods for approximating polytope volume. In: Proceedings of the 30th Annual Symposium on Computational Geometry, p. 318. ACM (2014)
15. Lovász, L., Vempala, S.: Hit-and-run from a corner. SIAM J. Comput. **35**, 985–1005 (2006)
16. Bourgain, J.: Random points in isotropic convex sets. Convex Geom. Anal. **34**, 53–58 (1996)

17. Rudelson, M.: Random vectors in the isotropic position. J. Funct. Anal. **164**, 60–72 (1999)
18. Adamczak, R., Litvak, A., Pajor, A., Tomczak-Jaegermann, N.: Quantitative estimates of the convergence of the empirical covariance matrix in log-concave ensembles. J. Am. Math. Soc. **23**, 535–561 (2010)
19. Štefankovič, D., Vempala, S., Vigoda, E.: Adaptive simulated annealing: a near-optimal connection between sampling and counting. J. ACM **56**(3), 18–36 (2009)
20. Kannan, R., Lovász, L., Simonovits, M.: Random walks and an $O^*(n^5)$ volume algorithm for convex bodies. Random Struct. Algorithms **11**, 1–50 (1997)
21. Gillman, D.: A Chernoff bound for random walks on expander graphs. In: FOCS, pp. 680–691. IEEE Comput. Soc. Press, Los Alamitos (1993)
22. Kannan, R., Lovász, L., Simonovits, M.: Isoperimetric problems for convex bodies and a localization lemama. Discrete Comput. Geom. **13**, 541–559 (1995)
23. Canfield, E.R., McKay, B.: The asymptotic volume of the Birkhoff polytope. Online J. Anal. Comb. **4**, 4 (2009)
24. Chan, C., Robbins, D., Yuen, D.: On the volume of a certain polytope. Exp. Math. **9**(1), 91–99 (2000)
25. De Loera, J.A., Liu, F., Yoshida, R.: A generating function for all semi-magic squares and the volume of the Birkhoff polytope. J. Algebraic Comb. **30**(1), 113–139 (2009)
26. Pak, I.: Four questions on Birkhoff polytope. Ann. Comb. **4**(1), 83–90 (2000)
27. Zeilberger, D.: Proof of a conjecture of Chan, Robbins, and Yuen. Electron. Trans. Numer. Anal. **9**, 147–148 (electronic) (1999) [Orthogonal polynomials: numerical and symbolic algorithms (Leganés, 1998)]
28. Beck, M., Pixton, D.: The Ehrhart polynomial of the Birkhoff polytope. Discrete Comput. Geom. **30**(4), 623–637 (2003)
29. Chan, C., Robbins, D.: On the volume of the polytope of doubly stochastic matrices. Exp. Math. **8**(3), 291–300 (1999)
30. Dyer, M., Gritzmann, P., Hufnagel, A.: On the complexity of computing mixed volumes. SIAM J. Comput. **27**(2), 356–400 (1998)
31. Cousins, B., Vempala, S.: Volume computation and sampling. http://www.cc.gatech.edu/~bcousins/volume.html (2013)