HAL
open science

Sampling the feasible sets of SDPs and volume approximation

Tolis Chalkis, Vissarion Fisikopoulos, Panagiotis Repouskos, Elias Tsigaridas

To cite this version:

Tolis Chalkis, Vissarion Fisikopoulos, Panagiotis Repouskos, Elias Tsigaridas. Sampling the feasible sets of SDPs and volume approximation. ACM Communications in Computer Algebra, In press, 10.1145/3457341.3457349 . hal-02572792

HAL Id: hal-02572792

https://inria.hal.science/hal-02572792v1

Submitted on 13 May 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Sampling the feasible sets of SDPs and volume approximation

Tolis Chalkis
achalkis@di.uoa.gr
Department of Informatics & Telecommunications
National & Kapodistrian University of Athens, Greece

Vissarion Fisikopoulos
vfisikop@di.uoa.gr
Department of Informatics & Telecommunications
National & Kapodistrian University of Athens, Greece

Panagiotis Repouskos
cs1180004@di.uoa.gr
Department of Informatics & Telecommunications
National & Kapodistrian University of Athens, Greece

Elias Tsigaridas
elias.tsigaridas@inria.fr
Inria Paris, IMJ-PRG,
Sorbonne Université and Paris Université

ABSTRACT

We present algorithmic, complexity, and implementation results on the problem of sampling points in the interior and the boundary of a spectrahedron, that is the feasible region of a semidefinite program.

Our main tool is random walks. We define and analyze a set of primitive geometric operations that exploits the algebraic properties of spectrahedra and the polynomial eigenvalue problem, and leads to the realization of a broad collection of efficient random walks. We demonstrate random walks that experimentally show faster mixing time than the ones used previously for sampling from spectrahedra in theory or applications, for example Hit and Run. Consecutively, the variety of random walks allows us to sample from general probability distributions, for example the family of log-concave distributions which arise frequently in numerous applications.

We apply our tools to compute (i) the volume of a spectrahedron and (ii) the expectation of functions coming from robust optimal control. We provide a C++ open source implementation of our methods that scales efficiently up to to dimension 200. We illustrate its efficiency on various data sets.

ACM Reference Format:
Tolis Chalkis, Vissarion Fisikopoulos, Panagiotis Repouskos, and Elias Tsigaridas. 2020. Sampling the feasible sets of SDPs and volume approximation. In *Proceedings of* . ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Spectrahedra are probably the most well studied shapes after polyhedra. We can represent polyhedra as the intersection of the positive orthant with an affine subspace. Spectrahedra generalize polyhedra, in the sense that they are the intersection of the cone of positive semidefinite matrices —*i.e.,* symmetric matrices with non-negative eigenvalues— with an affine space. In other words, a spectrahedron $S \subset \mathbb{R}^n$ is the feasible set of a linear matrix inequality. That is, if

$$F(x) = A_0 + x_1 A_1 + \cdots + x_n A_n, \tag{1}$$

A_i are symmetric matrices in $\mathbb{R}^{m\times m}$, then $S = \{x \in \mathbb{R}^n \mid F(x) \succeq 0\}$, where $\succeq$ denotes positive semidefiniteness. We assume that S is bounded of dimension n. Spectrahedra are convex sets (Fig. 1 & 3) and every polytope is a spectrahedron, but not the opposite. They are the feasible regions of semidefinite programs [45] in the way that polyhedra are feasible regions of linear programs.

Efficient methods for sampling points in spectrahedra are crucial for many applications, such as volume approximation [14], integration [37], semidefinite optimization [25, 37] and applications in robust control analysis [9, 10, 50]. To sample in the interior or on the boundary of S, we employ geometric random walks [54]. A geometric random walk on S starts at some interior point and at each step moves to a "neighboring" point that we choose according to some distribution depending only on the current point. For example, in the so-called ball walk, we move to a point p that we choose uniformly at random in a ball of fixed radius δ, if $p \in S$. The complexity of a random walk depends on its mixing time —the number of steps required to bound the distance between the current and the stationary distribution— and the complexity of the basic geometric operations that we perform at each step of the walk; we also call the latter per-step complexity.

The majority of geometric random walks are defined for general convex bodies and are based on an oracle; usually the membership oracle. There are also a few, e.g., Vaidya walk [12], sub-linear ball walk [39], specialized for polytopes. Most results on their analysis focus on convergence and mixing time, while the operations they perform at each step are defined abstractly and are enclosed in the corresponding oracle. That is why the complexity bounds involve the number of oracle calls.

To specialize a random walk for a family or representation of convex bodies one has to come up with efficient algorithms for the basic geometric operations to realize the (various) oracles. These operations should exploit the underlying geometric and algebraic properties and are of independent interest. Even more, they dominate the per-step complexity and they are crucial both for the overall complexity to sample a point from the target distribution and for an implementation.

The study of basic geometric operations to sample from non-linear convex objects finds its roots in non-linear computational geometry. During the last two decades, there are combined efforts [7] to develop efficient algorithms for the basic operations (predicates) that are behind classical geometric algorithms, like convex hull,

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
,
© 2020 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

arrangements, Voronoi diagrams, to go beyond points and lines and handle curved objects.

To our knowledge, only the Hit and Run (HnR) random walk [48] has been studied for spectrahedra [9]. To exploit the various other walks, like Ball walk [54], Billiard walk [22], Hamiltonian Monte Carlo (HMC) [1] we need to provide geometric operations, such as the reflection of a curve at the boundary and computing the intersection point of a curve at the boundary.

We should mention that there is a gap [5, 15] between the theoretical worst case bounds for the mixing times and the practical performance of the random walk algorithms. Thus, it is not accurate to claim (for all the random walks) that the speed of convergence to the target distribution is the same for different families of convex bodies. Furthermore, there are random walks without known theoretical mixing times, such as Coordinate Directions HnR, billiard walk or HMC with reflections. To study them experimentally, the efficient realization of the corresponding oracles is crucial.

Previous Work. Sampling convex sets via random walks has attained a lot of interest in the last decades. Most of the works assume either convex sets or polytopes; [39] provides an overview of the state-of-the-art. Random walks on spectrahedra are studied in [16, 44], where it exploits the Hit and Run walk and the computation of the intersection reduces to a generalized eigenvalue problem.

The Billiard walk [22] is a general way of sampling in convex or non-convex shapes from the uniform distribution. A mathematical billiard consists of a domain $\mathcal{D}$ and a point-mass, moving freely in $\mathcal{D}$ [49]. When this point-mass hits the boundary, it performs a specular reflection, albeit without losing velocity. An application of billiards is the study of optical properties of conics [49, Sec. 4].

If the trajectory is not a line, but rather a parametric curve, then the intersection operation reduces to the polynomial eigenvalue problem (PEP). HMC with reflections requires this operation. PEP is a well-studied problem in computational mathematics, e.g., [51, 52], and it appears in many applications. There are important results both for the perturbation analysis of PEP [4, 17, 51], as well as for the condition-based analysis of algorithms for the real and complex versions of PEP, if we assume random inputs [2, 3].

For the closely related problem of volume computation, there is an extensive bibliography [8]. The bulk of the theoretical studies are either for general convex bodies [14, 18] or polytopes [33]. Practical algorithms and implementations exist only for polytopes [15, 20]. Nevertheless, there are notable exceptions that consider algorithms for computing the volume of compact (basic) semi-algebraic sets. For example, [29] that exploits the periods of rational integrals. In the same setting, [24, 28] introduce numerical approximation schemes for volume computations, which rely on the moment-based algorithms and semi-definite programming.

Finally, sampling from a multivariate distribution is a central problem in many applications. For example, it is useful in robust control analysis [9, 10, 50] to overcome the worst case hardness as well as in integration [37] and convex optimization [16, 25].

Our contribution. We develop a framework of basic geometric operations for computations with spectrahedra. Based on this framework we employ a rich class of geometric random walks, which in turn we use to build efficient methods for sampling points from spectarhedra, under various probability distributions. We apply these tools to approximate the volume of spectrahedra and to integrate over spectrahedral domains. This extends the limits of the state-of-the-art methods that sample from spectrahedra and entail only the Hit and Run algorithm [9, 16]. We offer an efficient C++ implementation of our algorithms as a development branch of the package `volesti`[1], an open source library for high dimensional sampling and volume computation. While the implementation is in C++, there is also an R interface, for easier access to its functionality.

We demonstrate the efficiency of our approach and implementation on problems from robust control and optimization. First, as a special case of integration, we approximate the volume of spectrahedra up to dimension 100; this is, to the best of our knowledge, the first time such computations for non-linear objects are performed in high dimensions. Then, we approximate the expected value of a function $f : \mathbb{R}^n \to [0, 1]$, whose argument is a random variable having uniform distribution over a spectrahedron of dimension 200.

Finally, we sample from the Boltzmann distribution using HMC; this exploits a random walk in a spectrahedron S that employs a polynomial trajectory of degree two. Sampling using HMC from truncated distributions is a classical problem in computational statistics [1, 13], alas existing approaches handle either special distributions or special cases of constraints [30, 40]. We equip HMC with geometric operations to handle log-concave densities truncated by linear matrix inequalities (LMI) constraints. A combination of Boltzmann distribution with a simulated annealing technique [25] could lead to a practical solver for semidefinite programs (SDP).

Notation. We denote by O, resp. O_B, the arithmetic, resp. bit, complexity and we use $\widetilde{O}$, respectively $\widetilde{O}_B$, to ignore (poly-)logarithmic factors. The bitsize of a univariate polynomial $A \in \mathbb{Z}[x]$ is the maximum bitsize of its coefficients. We use bold letters for matrices, $\boldsymbol{A}$, and vectors, $\boldsymbol{v}$; we denote by $A_{i,j}$, resp. v_i, their elements; $\boldsymbol{A}^\top$ is the transpose and $\boldsymbol{A}^*$ the adjoint of $\boldsymbol{A}$. If $\boldsymbol{x} = (x_1, \ldots, x_n)$, then $\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{A}_0 + \sum_{i=1}^n x_i \boldsymbol{A}_i$, see (1). For two points $\boldsymbol{x}$ and $\boldsymbol{y}$, we denote the line through them by $\ell(\boldsymbol{x}, \boldsymbol{y})$ and their segment as $[\boldsymbol{x}, \boldsymbol{y}]$. For a spectrahedron S, lets its interior be S° and its boundary ∂S. We represent a probability distribution π with a probability density function $\pi(x)$. When π is truncated to S the support of $\pi(x)$ is S. If π is log-concave, then $\pi(\boldsymbol{x}) \propto e^{-\alpha f(\boldsymbol{x})}$, where $f : \mathbb{R}^d \to \mathbb{R}$ a convex function. Finally, let $\mathcal{B}_n$ be the n-dimensional unit ball and denote by $\partial\mathcal{B}_n$ its boundary.

2 BASIC GEOMETRIC OPERATIONS

Our toolbox for computations with spectrahedra and implementing random walks, consists of three basic geometric operations: MEMBERSHIP, INTERSECTION, and REFLECTION. For a spectrahedron S, MEMBERSHIP decides if a point is inside S, INTERSECTION computes the intersection of an algebraic curved trajectory C with the ∂S, and REFLECTION computes the reflection of an algebraic curved trajectory when it hits ∂S. We need the last two operations because random walks can move along non-linear trajectories inside convex bodies. For the ones that we consider, the trajectories are parametric polynomial curves, of various degrees. To compute with these curves we need to solve a polynomial eigenvalue problem (PEP).

[1] https://github.com/GeomScale/volume_approximation

2.1 An algorithm for PEP

To estimate the complexity of INTERSECTION we need the complexity of PEP. The Polynomial Eigenvalue Problem (PEP) consists in computing $\lambda \in \mathbb{R}$ and $\boldsymbol{x} \in \mathbb{R}^m$ that satisfy the (matrix) equation

$$P(\lambda)\,\boldsymbol{x} = 0 \Leftrightarrow (\boldsymbol{B}_d\lambda^d + \cdots + \boldsymbol{B}_1\lambda + \boldsymbol{B}_0)\boldsymbol{x} = 0 \ , \qquad (2)$$

where $P(\lambda)$ is a univariate matrix polynomial whose coefficients are the matrices $\boldsymbol{B}_i \in \mathbb{R}^{m\times m}$. We further assume that $\boldsymbol{B}_d$ and $\boldsymbol{B}_0$ are invertible. In general, there are $\delta = m\,d$ values of λ. We refer the reader to [51, 52] for a thorough exposition of PEP.

One approach for solving PEP is to linearalize the problem and to express λ's as the eigenvalues of a bigger matrix. For this we transform Eq. (2) into a linear pencil of dimension δ. Following closely [4], the *Companion Linearization* consists in solving the generalized eigenvalue problem $\boldsymbol{C}_0 - \lambda\boldsymbol{C}_1$, where

$$\boldsymbol{C}_0 = \begin{bmatrix} \boldsymbol{B}_d & 0 & \cdots & 0 \\ 0 & \boldsymbol{I}_m & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \boldsymbol{I}_m \end{bmatrix} \text{ and } \boldsymbol{C}_1 = \begin{bmatrix} \boldsymbol{B}_{d-1} & \boldsymbol{B}_{d-2} & \cdots & \boldsymbol{B}_0 \\ -\boldsymbol{I}_m & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & -\boldsymbol{I}_m & 0 \end{bmatrix},$$

and $\boldsymbol{I}_m$ denotes the $m \times m$ identity matrix. The eigenvectors $\boldsymbol{x}$ and $\boldsymbol{z}$ are related $\boldsymbol{z} = [1, \lambda, \ldots, \lambda^{d-2}, \lambda^{d-1}]^\top \otimes \boldsymbol{x}$.

To obtain an exact algorithm for PEP we exploit the assumption that $\boldsymbol{B}_d$ is invertible to transform the problem to the following classical eigenvalue problem $(\lambda\boldsymbol{I}_d - \boldsymbol{C}_2)\boldsymbol{z} = 0$, where

$$\boldsymbol{C}_2 = \begin{bmatrix} \boldsymbol{B}_{d-1}\boldsymbol{B}_d^{-1} & \boldsymbol{B}_{d-2}\boldsymbol{B}_d^{-1} & \cdots & \boldsymbol{B}_0\boldsymbol{B}_d^{-1} \\ -\boldsymbol{I}_m & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & -\boldsymbol{I}_m & 0 \end{bmatrix}.$$

The eigenvectors are roots of the characteristic polynomial of $\boldsymbol{C}_2$. Therefore, the problem is to compute the eigenvalues of $\boldsymbol{C} \in \mathbb{R}^{\delta\times\delta}$. From a complexity point of view the best algorithm to compute the eigenvalues relies on computing the roots of the characteristic polynomial [47]. We also follow this approach. However, in practise other methods are more efficient and stable.

LEMMA 2.1. *Consider a PEP of degree d, involving matrices of dimension $m \times m$, with integer elements of bitsize at most τ, see (2). There is a randomized algorithm for computing the eigenvalues and the eigenvectors of PEP up to precision $\epsilon = 2^{-L}$, in $\widetilde{O}_B(\delta^{\omega+3}L)$, where $\delta = md$ and $L = \Omega(\delta^3\tau)$. The arithmetic complexity is $\widetilde{O}(\delta^{2.697} + \delta\lg(1/\epsilon))$.*

2.2 MEMBERSHIP

The operation MEMBERSHIP$(\boldsymbol{F}, \boldsymbol{p})$ decides if a point $\boldsymbol{p}$ lies in the interior of a spectrahedron $S = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{F}(\boldsymbol{x}) \succeq 0\}$. For this, first, we construct the matrix $\boldsymbol{F}(\boldsymbol{p})$. Next, if the matrix is positive definite, then $\boldsymbol{p} \in S^\circ$, if it is positive semidefinite, then $\boldsymbol{p} \in \partial S$, and otherwise $\boldsymbol{p} \in \mathbb{R}^n \setminus S$. The pseudo-code appears in Alg. 1.

LEMMA 2.2. *Alg. 1, MEMBERSHIP$(\boldsymbol{F}, \boldsymbol{p})$, requires $\widetilde{O}(nm^2 + m^{2.697})$ arithmetic operations. If $\boldsymbol{F}$ and $\boldsymbol{p}$ have integers elements of bitsize at most τ, resp. σ, then the bit complexity is $\widetilde{O}_B((nm^2 + m^{3.697})(\tau + \sigma))$.*

2.3 INTERSECTION

Consider a parametric polynomial curve C such that it has a non-empty intersection with a spectrahedron S. Assume that the value

Algorithm 1: MEMBERSHIP$(\boldsymbol{F}, \boldsymbol{p})$

Input : An LMI $\boldsymbol{F}(\boldsymbol{x}) \succeq 0$ representing a spectrahedron S and a point $\boldsymbol{p} \in \mathbb{R}^n$.
Output : TRUE if $\boldsymbol{p} \in S$, FALSE otherwise.

```
λ_min ← smallest eigenvalue of F(p);
if λ_min ≥ 0 then return TRUE ;
return FALSE ;
```

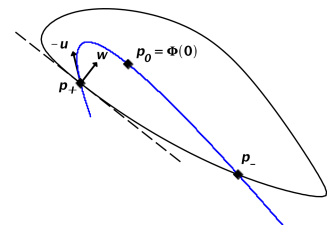

Figure 1: *A spectrahedron S described by $\boldsymbol{F}(\boldsymbol{x})$ and a parameterized curve Φ. The point $\boldsymbol{p}_0 = \Phi(0)$ lies in the interior of S, and the points $\boldsymbol{p}_+ = \Phi(t_+)$ and $\boldsymbol{p}_- = \Phi(t_-)$ on the boundary. The vector $\boldsymbol{w}$ is the surface normal of ∂S at $\boldsymbol{p}_+$ and $\boldsymbol{u}$ the direction of Φ at time $t = t_+$.*

of the parameter $t = 0$ corresponds to a point, $\boldsymbol{p}_0$, that lies in $C \cap S^\circ$. Further assume that the part of C that $\boldsymbol{p}_0$ lies on, intersects the boundary of S transversally at two points, say $\boldsymbol{p}_-$ and $\boldsymbol{p}_+$. The operation INTERSECTION computes the parameters, t_- and t_+, corresponding to these two points. Fig. 1 illustrates this discussion and the pseudo-code of INTERSECTION appears in Alg. 2.

To prove correctness and estimate its complexity we proceed as follows: As before, S is the feasible region of an LMI $\boldsymbol{F}(\boldsymbol{x}) \succeq 0$. Consider the real trace of a polynomial curve C, with parametrization

$$\begin{array}{rcccl} \Phi & : & \mathbb{R} & \to & \mathbb{R}^n \\ & & t & \mapsto & \Phi(t) := (p_1(t), \ldots, p_n(t)), \end{array} \qquad (3)$$

where $p_i(t) = \sum_{j=0}^{d_i} p_{i,j}t^j$ are univariate polynomials in t of degree d_i, for $i \in [m]$. Also let $d = \max_{i\in[m]}\{d_i\}$. If the coefficients of the polynomials are integers, then we further assume that the maximum coefficient's bitsize is bounded by τ.

As t varies over the real line, there may be several disjoint intervals, for which the corresponding part of C lies in S°. We aim to compute the endpoints, t_- and t_+, of a maximum interval containing $t = 0$. Let $\boldsymbol{p}_0 = \Phi(0)$; by assumption it holds $\boldsymbol{F}(\Phi(0)) = \boldsymbol{F}(\boldsymbol{p}_0) \succ 0$.

The input of INTERSECTION (Alg. 2) is $\boldsymbol{F}$, the LMI representation of S, and $\Phi(t)$, the polynomial parametrization of C. Its crux is a routine, PEP, that solves a polynomial eigenvalue problem. The following lemma exploits this relation.

LEMMA 2.3 (PEP AND $S \cap C$). *Consider the spectrahedron $S = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{F}(\boldsymbol{x}) \succeq 0\}$. Let $\Phi : \mathbb{R} \to \mathbb{R}^n$ be a parametrization of a polynomial curve C, such that $\Phi(0) \in S^\circ$. Let $[t_-, t_+]$ be the maximum interval containing 0, such that the corresponding part of C lies in S. Then, t_-, resp. t_+, is the maximum negative, resp. minimum positive, polynomial eigenvalue of $\boldsymbol{F}(\Phi(t))\boldsymbol{x} = 0$, where $\boldsymbol{F}(\Phi(t)) = \boldsymbol{B}_0 + t\boldsymbol{B}_1 + \cdots + t^d\boldsymbol{B}_d$.*

Algorithm 2: INTERSECTION$(F, \Phi(t))$

Input : An LMI $F(\boldsymbol{x}) \succeq 0$ for a spectrahedron S and a parametrization $\Phi(t)$ of a polynomial curve C

Require : $\Phi(0) \in S^\circ$

Output : t_-, t_+ s.t. $\Phi(t_-), \Phi(t_+) \in \partial S$

$T := \{t_1 \leq t_2 \leq \cdots \leq t_\ell\} \leftarrow$ PEP$(F(\Phi(t)))$;

$t_- \leftarrow \max\{t \in T \mid t < 0\}$; // *max neg polynomial eigenvalue*

$t_+ \leftarrow \min\{t \in T \mid t > 0\}$; // *min pos polynomial eigenvalue*

return t_-, t_+;

2.3.1 Complexity of INTERSECTION. We have to construct PEP and solve it. If $\Phi(t)$ has degree d, then $F(\Phi(t)) = B_0 + tB_1 + \cdots + t^d B_d$. This construction costs $O(dnm^2)$ operations. The solving phase, from Lemma 2.1 requires $\widetilde{O}((md)^{2.697} + md \lg L)$ arithmetic operations and dominates the complexity bound of the operation.

2.4 REFLECTION

The REFLECTION operation (Alg. 3) takes as input an LMI F representation of a spectrahedron S and a polynomial curve C, given by a parametrization Φ. Assume that $t = 0$ corresponds to a point $\Phi(0) \in S^\circ \cap C$. Starting from $t = 0$, we increase t along the positive real semi-axis. As t changes, we move along the curve C through $\Phi(t)$, until we hit the boundary of S at the point $\boldsymbol{p}_+ := \Phi(t_+) \in \partial S$, for some $t_+ > 0$. Then, a specular reflection occurs at this point with direction $\boldsymbol{s}_+$; this is the reflected direction. We output t_+ and $\boldsymbol{s}_+$. Fig. 1 depicts the procedure.

The boundary of S, ∂S, with respect to the Euclidean topology, is a subset of the real algebraic set $\{\boldsymbol{x} \in \mathbb{R}^n \mid \det(F(\boldsymbol{x})) = 0\}$. The latter is a real hypersurface defined by one (determinantal) equation. For any $\boldsymbol{x} \in \partial S$ we have $\operatorname{rank}(F(\boldsymbol{x})) \leq m-1$. We assume that $\boldsymbol{p}_+ = \Phi(t_+)$ is such that $\operatorname{rank}(F(\boldsymbol{p}_+)) = m - 1$. The normal direction at a point $\boldsymbol{p} \in \partial S$, is the gradient of $\det F(\boldsymbol{p})$.

We compute the reflected direction using the following formula

$$\boldsymbol{s}_+ = \boldsymbol{u} - \frac{2}{|\boldsymbol{w}|^2} \langle \boldsymbol{u}, \boldsymbol{w} \rangle \, \boldsymbol{w}, \tag{4}$$

where $\boldsymbol{w}$ is the normalized gradient vector at the point $\Phi(t_+)$ and $\boldsymbol{u} = \frac{d\Phi}{dt}(t_+)$ is the direction of the trajectory at this point. We illustrate the various vectors in Fig. 1.

Algorithm 3: REFLECTION $(F, \Phi(t))$

Input : An LMI $F(\boldsymbol{x}) \succeq 0$ for a spectrahedron S and a parametrization $\Phi(t)$ of a polynomial curve C.

Require : (i) $\Phi(0) \in S^\circ$
(ii) C intersects ∂S transversally at a smooth point.

Output : t_+ such that $\Phi(t_+) \in \partial S$ and the direction of the reflection, $\boldsymbol{s}_+$, at this point.

$t_-, t_+ \leftarrow$ INTERSECTION $(F, \Phi(t))$;

$\boldsymbol{w} \leftarrow \nabla \det F(\Phi(t_+))$;

$\boldsymbol{w} \leftarrow \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$; // *Normalize* $\boldsymbol{w}$

$\boldsymbol{s}_+ \leftarrow \frac{d\Phi}{dt}(t_+) - 2 \langle \nabla \frac{d\Phi}{dt}(t_+), \boldsymbol{w} \rangle \, \boldsymbol{w}$;

return $t_+, \boldsymbol{s}_+$;

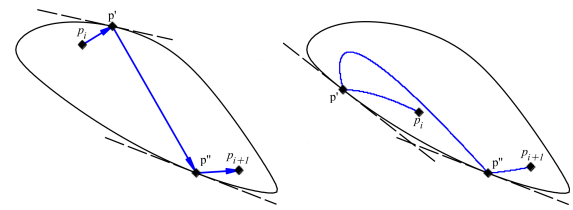

Figure 2: *The i-th step of the W-*BILLARD *[5] (left) and of the W-HMC-*R *[6] (right) random walks.*

LEMMA 2.4 (GRADIENT OF $\det F(\boldsymbol{x})$). *Assume that $\boldsymbol{x} \in \partial S$ and the rank of the $m \times m$ matrix $F(\boldsymbol{x})$ is $m - 1$. Then*

$$\nabla \det(F(\boldsymbol{x})) = c \cdot (\boldsymbol{v}^\top A_1 \boldsymbol{v}, \cdots, \boldsymbol{v}^\top A_n \boldsymbol{v}), \tag{5}$$

where $c = \frac{\mu(F(\boldsymbol{x}))}{|\boldsymbol{v}|^2}$, $\mu(F(\boldsymbol{x}))$ is the product of the nonzero eigenvalues of $F(\boldsymbol{x})$, and $\boldsymbol{v}$ is a non-trivial vector in the kernel of $F(\boldsymbol{x})$. If $\operatorname{rank}(F(\boldsymbol{x})) \leq m - 2$, then the gradient is the zero.

REFLECTION exploits Lemma 2.4. Nevertheless, it is not necessary to perform all computations indicated by the lemma. Since, we will normalize the resulting vector and we do not need its actual direction (internal or external), we can omit the computation of c. Moreover, the nonzero vector $\boldsymbol{v}$ s.t. $F(\boldsymbol{p})\boldsymbol{v} = 0$, corresponds to the eigenvector w.r.t. the eigenvalue t_+ from the PEP (Lem. 2.3). This holds because $\boldsymbol{p} = \Phi(t_+) \in \partial S$ and thus $\det F(\Phi(t_+)) = 0$.

We compute the eigenvalues of PEP up to some precision. Since matrix-vector multiplication is backward stable, a small perturbation on $\boldsymbol{v}$ does not affect the computation of each coordinate of $\nabla \det(F(\boldsymbol{x}))$ [53, p. 104]. We quantify the accuracy of the computed $\nabla \det(F(\boldsymbol{x}))$ using floating point arithmetic as follows:

LEMMA 2.5. *The relative error of each coordinate of the gradient given in Lemma 2.4 when we compute it using floating point arithmetic with machine epsilon ϵ_M is $O(\frac{\epsilon_M}{\sigma_{\max}(A_i)})$, for $i \in [n]$, where $\sigma_{\max}$ is the largest singular value of A_i.*

As mentioned, for $\nabla \det(F(\boldsymbol{x}))$ we just need to compute $(\boldsymbol{v}^\top A_1 \boldsymbol{v}, \cdots, \boldsymbol{v}^\top A_n \boldsymbol{v})$. If we have already computed $\boldsymbol{v}$, then we need $O(nm^2)$ operations. Computing the derivative of $\Phi(t)$ is straightforward, since Φ is a univariate polynomial. Taking into account the complexity of INTERSECTION, the total complexity for REFLECTION is $\widetilde{O}((md)^{2.697} + md \lg L + dnm^2 + nm^2) = \widetilde{O}((md)^{2.697} + md \lg L + dnm^2)$.

2.5 An example in 2D

Consider a spectrahedron S in the plane (Fig. 2), given by an LMI $F(\boldsymbol{x}) = A_0 + x_1 A_1 + x_2 A_2$. The matrices $\boldsymbol{A}$ are in the appendix.

Starting from point $\boldsymbol{p}_0 = (-1, 1)^\top$, we walk along the line L with parametrization: $\Phi(t) = \boldsymbol{p}_0 + t\boldsymbol{u}$, where $\boldsymbol{u} = (1.3, 0.8)^\top$. Then, INTERSECTION finds the intersection of S with L, by solving the degree one PEP, $(B_0 + tB_1)\boldsymbol{x} = 0$, where $B_0 = F(\boldsymbol{p}_0)$ and $B_1 = u_1 A_1 + u_2 A_2$. Acquiring $t_- = -0.8$ and $t_+ = 0.5$, we get the intersection point $\boldsymbol{p}_1$, which corresponds to $\boldsymbol{p}_0 + t_+ \boldsymbol{u} = (-0.3, 1.4)^\top$.

	per-step Complexity
W-HnR	$O(m^{2.697} + m \lg L + nm^2)$
W-CHnR	$O(m^{2.697} + m \lg L + m^2)$
W-Billard	$\widetilde{O}(\rho(m^{2.697} + m \lg L + nm^2))$
W-HMC-r	$\widetilde{O}(\rho((dm)^{2.697} + md \lg L + dnm^2))$

Table 1: *The per-step complexity of the random walks in Sec. 3.*

To compute the direction of the trajectory, immediately after we reflect on the boundary of S at $\boldsymbol{p_1}$, reflection computes

$$\boldsymbol{w} = \frac{\nabla \det \boldsymbol{F}(\Phi(t_+))}{|\nabla \det \boldsymbol{F}(\Phi(t_+))|} = (\boldsymbol{v}^\top \boldsymbol{A}_1 \boldsymbol{v}, \boldsymbol{v}^\top \boldsymbol{A}_2 \boldsymbol{v})^\top = (-0.2, -1)^\top, \quad (6)$$

where $\boldsymbol{v}$ is the eigenvector of $(\boldsymbol{B}_0 + t\boldsymbol{B}_1)\boldsymbol{x} = 0$, with eigenvalue t_+. The reflected direction is $\boldsymbol{u'} = \boldsymbol{u} - 2\langle \boldsymbol{u}, \boldsymbol{w}\rangle\, \boldsymbol{w} = (0.8, -1.3)^\top$.

3 RANDOM WALKS

Using the basic geometric operations of Sec. 2, we implement and analyze three random walks for spectrahedra: Hit and Run (W-HnR) , Billiard Walk (W-Billard), and Hamiltonian Monte Carlo with reflections (W-HMC-r). In Table 1, we present the per-step arithmetic complexity for each random walk.

3.1 Hit and Run

W-HnR (Alg. 4) is a random walk that samples from any probability distribution π, truncated to a convex body K; in our case a spectrahedron S. However, for its mixing time, there exist bounds only when π is log-concave distribution (e.g. the uniform distribution), which is $\widetilde{O}(n^3)$. At the i-th step, W-HnR chooses uniformly at random a (direction of a) line ℓ, passing from its current position $\boldsymbol{p}_i$. Let $\boldsymbol{p}_1, \boldsymbol{p}_2$ be the intersection points of ℓ with S. Let π_ℓ be the restriction of π on the segment $[\boldsymbol{p}_2, \boldsymbol{p}_2]$. Then, we choose $\boldsymbol{p}_{i+1}$ from $[\boldsymbol{p}_1, \boldsymbol{p}_2]$ w.r.t. the distribution π_ℓ.

Algorithm 4: Hit-and-Run_Walk (W-HnR)

Input : LMI $\boldsymbol{F}(\boldsymbol{x}) \geq 0$ for a spectrahedron S & a point $\boldsymbol{p}_i$.
Require : $\boldsymbol{p}_i \in S$
Output : The point $\boldsymbol{p}_{i+1}$ of the $(i+1)$-th step of the walk.

$\boldsymbol{v} \leftarrow_R \mathcal{U}(\partial\mathcal{B}_n)$; // *choose direction*
$\Phi(t) := \boldsymbol{p}_i + t\boldsymbol{v}$; // *define trajectory*
$t_-, t_+ \leftarrow$ intersection $(\boldsymbol{F}, \Phi(t))$;
$\boldsymbol{p}_{i+1} \leftarrow_R [\boldsymbol{p}_i + t_-\boldsymbol{v},\ \boldsymbol{p}_i + t_+\boldsymbol{v}]$ w.r.t. π_ℓ;
return $\boldsymbol{p}_{i+1}$;

The per-step complexity of W-HnR is dominated by the intersection, which requires $O(nm^2)$ operations for the construction of the pep and $\widetilde{O}(m^{2.697} + m \lg 1/\epsilon)$ for solving it, where we want to approximate the intersection point up to a factor or 2^{-L}.

There is a variation of W-HnR, the *coordinate directions Hit and Run* (W-CHnR) [48], in which the direction vector is chosen randomly and uniformly from the vector basis $\{\boldsymbol{e}_i,\ i \in [n]\}$. In W-CHnR, for every step aside the first, the construction of the pep takes $O(m^2)$ operations, and the complexity does not depend on the dimension n. The reason is, that to build the pep $\boldsymbol{F}(\boldsymbol{p}_i) + t\boldsymbol{e}_j) = \boldsymbol{F}(\boldsymbol{p}_i) + t\boldsymbol{A}_j$, the value of $\boldsymbol{F}(\boldsymbol{p}_i)$ can be obtained via $\boldsymbol{F}(\boldsymbol{p}_i) = \boldsymbol{F}(\boldsymbol{p}_{i-1}) + \hat{t}\boldsymbol{A}_k$, assuming in the previous step $\boldsymbol{e}_k$ was chosen as direction. There is not a theoretical mixing time for W-CHnR.

3.2 Billiard walk

W-Billard [43], Alg. 5, is used to sample a convex body K under the uniform distribution; no theoretical results for its mixing time exist. At i-th step, being on position $\boldsymbol{p}_i$, it chooses uniformly a direction vector $\boldsymbol{v}$ and a number L, where $L = -\tau \ln \eta$, $\eta \sim U(0,1)$. Then, it moves at the direction of $\boldsymbol{v}$ for distance L. If during the movement, it hits the boundary without having covered the required distance L, then it continues on a reflected trajectory. If the number of reflections exceeds a bound ρ, it stays at $\boldsymbol{p}_i$. In [43] they experimentally conclude that W-Billard mixes faster when $\tau \approx diam(K)$.

Algorithm 5: Billiard_Walk (W-Billard)

Input : An LMI $\boldsymbol{F}(\boldsymbol{x}) \geq 0$ for a spectrahedron S, a point $\boldsymbol{p_i}$, the diameter τ of S and a bound ρ on the number of reflections.
Require : $\boldsymbol{p_i} \in S$
Output : The point $\boldsymbol{p_{i+1}}$ of the $(i+1)$-th step of the walk.

$L \leftarrow -\tau \ln \eta$; $\eta \leftarrow_R \mathcal{U}((0,1))$; // *choose length*
$\boldsymbol{v} \leftarrow_R \mathcal{U}(\partial\mathcal{B}_n)$; // *choose direction*
$\boldsymbol{p} \leftarrow \boldsymbol{p}_i$;
do
  $\Phi(t) := \boldsymbol{p} + t\boldsymbol{v}$; // *define trajectory*
  $t_+, \boldsymbol{s}_+ \leftarrow$ reflection $(\boldsymbol{F}, \Phi(t))$;
  $\hat{t} \leftarrow \min\{t_+, L\}$; $\boldsymbol{p} \leftarrow \Phi(\hat{t})$; $\boldsymbol{v} \leftarrow \boldsymbol{s}_+$; $L \leftarrow L - \hat{t}$;
while $L > 0$;
if $\#\{reflections\} > \rho$ **then return** $\boldsymbol{p_{i+1}} = \boldsymbol{p_i}$;
else return $\boldsymbol{p_{i+1}} = \boldsymbol{p}$;

The per-step complexity of W-Billard is dominated by the reflection, which requires $\widetilde{O}(m^{2.697} + m \lg L + nm^2)$ arithmetic operations, when we want to approximate the intersection point up to a factor of 2^{-L}. Since we allow at most ρ reflections per step, the total complexity become $\widetilde{O}(\rho(m^{2.697} + m \lg L + nm^2))$.

3.3 Hamiltonian Monte Carlo with Reflections

Hamiltonian Monte Carlo (HMC) samples from any probability distribution π. Our focus lies again on the log-concave distributions ($\pi(\boldsymbol{x}) \propto e^{-\alpha f(\boldsymbol{x})}$). In this case, if we assume that f is a strongly convex function, then the mixing time of HMC is $O(k^{1.5} \log(n/\epsilon))$, where κ is the condition number of $\nabla^2 f$ [32]. If we truncate π in a convex body, then we can use boundary reflections (W-HMC-r) to ensure that the random walk converges to the target distribution [13]; however, in this case the mixing time is unclear.

The Hamiltonian dynamics behind HMC operate on a n-dimensional position vector $\boldsymbol{p}$ and a n-dimensional momenta $\boldsymbol{v}$, so the full state space has $2n$ dimensions. The system is described by a function of $\boldsymbol{p}$ and $\boldsymbol{v}$ known as the Hamiltonian, $H(\boldsymbol{p}, \boldsymbol{v}) = U(\boldsymbol{p}) + K(\boldsymbol{v}) = f(\boldsymbol{p}) + \frac{1}{2}|\boldsymbol{v}|^2$. To sample from π one has to solve the

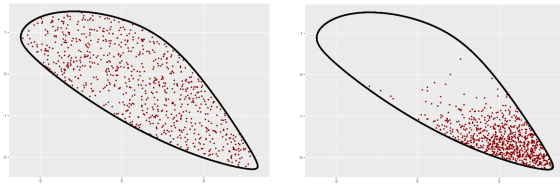

Figure 3: *Samples from the uniform distribution with W-Billard (left) and from the Boltzmann distribution $\pi(x) \propto e^{-cx/T}$, where $T = 1$, $c = [-0.09, 1]^T$, with W-HMC-R (right). The volume of this spectrahedron is* 10.23.

following system of Ordinary Differential Equations (ODE):

$$\begin{aligned} \frac{d\boldsymbol{p}}{dt} &= \frac{\partial H(\boldsymbol{p},\boldsymbol{v})}{\partial \boldsymbol{v}} \\ \frac{d\boldsymbol{v}}{dt} &= -\frac{\partial H(\boldsymbol{p},\boldsymbol{v})}{\partial \boldsymbol{p}} \end{aligned} \quad \begin{cases} \frac{d\boldsymbol{v}(t)}{dt} = -\alpha \nabla f(\boldsymbol{p}) \\ \frac{d\boldsymbol{p}(t)}{dt} = \boldsymbol{v}(t) \end{cases} \tag{7}$$

If $\pi(\boldsymbol{x})$ is a log-concave density, then we can approximate the solution of the ODE with a low degree polynomial trajectory [32], using the collocation method. A degree $d = O(1/\log(\epsilon))$ suffices to obtain a polynomial trajectory with error $O(\epsilon)$, for a fixed time interval, while we perform just $\widetilde{O}(1)$ evaluations of $\nabla f(\boldsymbol{x})$.

HMC at the i-th step uniformly samples a step ℓ from a proper interval to move on the trajectory implied by ODE (7), choses $\boldsymbol{v}$ randomly from $\mathcal{N}(0, I)$, and updates $\boldsymbol{p}$ using the ODE in (7), for $t \in [0, \ell]$. When π is truncated in a convex body, then W-HMC-R fixes an upper bound ρ on the number of reflections and reflects a polynomial trajectory as we describe in Sec. 2.4.

Algorithm 6: HMC_W_REFLECTION (W-HMC-R)

Input : An LMI $\boldsymbol{F}(\boldsymbol{x}) \succeq 0$ representing a spectrahedron S, a point $\boldsymbol{p_i}$, the diameter τ of S and a bound ρ to the number of reflections.
Require : $\boldsymbol{p_i} \in S$
Output : The point $\boldsymbol{p_{i+1}}$ of the $(i+1)$-th step of the walk.

```
ℓ ← τη;  η ←_R U((0,1)); // choose length
v ←_R N(0, I_d); // choose direction
do
    Compute trajectory Φ(t) from ODE (7);
    t_+, s_+ ← REFLECTION (F, Φ(t));
    t̂ ← min{t_+, ℓ} ; p ← Φ(t̂) ; v ← s ; ℓ ← ℓ − t̂ ;
while L > 0;
if # {reflections} > ρ then return p_{i+1} = p_i ;
return p_{i+1} = p ;
```

Each step of W-HMC-R, when $\pi(x)$ is a log-concave density truncated by S, costs $\widetilde{O}(\rho((dm)^{2.697} + md \lg L + dnm^2))$, if we approximate the intersection points up to a factor 2^{-L}, where d is the degree of the polynomial that approximates the solution of the ODE (7).

4 APPLICATIONS AND EXPERIMENTS

This section demonstrates and compares the algorithms of Sec. 3 and the efficiency of our software, on three applications that rely on sampling from spectrahedra. In the sequel, we call *walk length* the number of the intermediate points a random walk visits before "burning", in the process of producing a single sample. The greater the walk length of a random walk, the smaller the distance of the current distribution to the stationary (target) distribution.

Regarding the implementation, the code is parametrized by the floating point precision of the computations. We use `Eigen` [23] for basic linear algebra operations, such as Cholesky decomposition and matrix multiplication. For eigenvalue computations, we employ `Spectra` [19], which is based on Eigen and offers crucial optimizations, like the computation of the largest eigenvalue. Our software can sample spectrahedra up to dimension $n = 100$ and it is accessible from github[2]. The spectrahedra for the tests were generated randomly [16]. All experiments were performed on a PC with `Intel Core i7-6700 3.40GHz × 8 CPU` and `32GB RAM`.

4.1 Volume

We use the geometric operations (Sec. 2) and the random walks (Sec. 3) to compute the volume, vol(S), of a spectrahedron S. Our implementation approximates vol(S) within error 0.1 with high probability in a few minutes, for dimension $n = 100$.

A typical randomized algorithm for volume approximation exploits a Multiphase Monte Carlo (MMC) technique, which reduces volume approximation of convex body S to computing a telescoping product of ratios of integrals over S. In particular, for any sequence of functions $\{f_0, \ldots, f_k\}$, where $f_i : \mathbb{R}^n \to \mathbb{R}$, we have:

$$\mathrm{vol}(S) = \int_S 1dx = \int_S f_k(x)dx \frac{\int_S f_{k-1}(x)dx}{\int_S f_k(x)dx} \cdots \frac{\int_S 1dx}{\int_S f_0(x)dx}. \tag{8}$$

Notice that $\frac{\int_P f_{i-1}(x)dx}{\int_P f_i(x)dx} = \int_P \frac{f_{i-1}(x)}{f_i(x)} \frac{f_i(x)}{\int_P f_i(x)dx} dx$. To estimate each ratio of integrals, we sample N points from a distribution proportional to f_i and, we use the unbiased estimator $\frac{1}{N}\sum_{j=1}^{N} \frac{f_{i-1}(x_j)}{f_i(x_j)}$. To exploit Eq. (8) we have to (i) fix the sequence such that k is as small as possible, (ii) select f_i's such that we can compute efficiently each integral ratio, and (iii) compute $\int_P f_k(x)dx$. The best theoretical result of [14] fixes a sequence of spherical Gaussians $\{f_0, \ldots, f_k\}$ with the mode being in S, parameterized by the variance. The overall complexity is $\widetilde{O}(n^3)$ MEMBERSHIP calls. The implementation in [15] is based on this algorithm but handles only convex polytopes in H-representation as it requires the facets of the polytope and an inscribed ball to fix the sequence of Gaussians. Both the radius of the inscribed ball and the number of facets strongly influence the performance of the algorithm. So, it cannot handle efficiently the case of convex bodies without a facet description, e.g., zonotopes [15], as it results a big sequence of ratios that spoil practical efficiency.

Our approach is to consider the f_i's as a sequence of indicator functions of concentric balls centered in S, as in [20]. In particular, let f_k and f_0 be the indicator functions of $r\mathcal{B}_n$ and $R\mathcal{B}_n$ respectively, while $r\mathcal{B}_n \subseteq S \subseteq R\mathcal{B}_n$ and $S_i = (2^{(k-i)/n} r\mathcal{B}_n) \cap S$ for $i = 0, \ldots, k$. Thus, it suffices to compute vol($r\mathcal{B}_n$) and apply the following:

$$\mathrm{vol}(S) = \mathrm{vol}(S_k)\frac{\mathrm{vol}(S_{k-1})}{\mathrm{vol}(S_k)} \cdots \frac{\mathrm{vol}(S_0)}{\mathrm{vol}(S_1)}, \quad k = \lceil n \lg(R/r) \rceil. \tag{9}$$

We employ the annealing schedule from [11] to minimize k, without computing neither an enclosed $r\mathcal{B}_n$ nor an enclosing ball $R\mathcal{B}_n$ of S. We do so by probabilistically bounding each ratio of Eq. (9) in

[2] https://github.com/GeomScale/volume_approximation/tree/issac20

S-n-m	$\mu \pm t_{\alpha,\nu-1}\frac{s}{\sqrt{\nu}}$	*Points*	*Time (sec)*
S-40-40	(1.34 ± 0.12)e-06	9975.2	6.7
S-60-60	(1.23 ± 0.11)e-20	20370.9	28.5
S-80-80	(4.24 ± 0.26)e-33	31539.1	124.4
S-100-100	(1.21 ± 0.10)e-51	52962.7	362.3

Table 2: *(Volume of spectrahedra) For each S-n-m we run $\nu = 10$ experiments. μ stands for the average volume, s for the standard deviation. The second column gives a confidence interval with level of confidence $\alpha = 0.05$; $t_{\alpha,\nu-1}$ is the critical value of student's distribution with $\nu - 1$ degrees of freedom. Points for the average number of points generated and Time for the average runtime in seconds; m is the dimension of the matrix in LMI and n the ambient dimension. For all the above we set the error parameter $e = 0.1$.*

an interval $[r, r + \delta]$, which is given as input. To approximate each ratio of volumes, we sample uniformly distributed points from S_i and count points in S_{i-1}. We follow the experimental results of Sec. 4.2 and use W-Billard which mixes faster than W-HnR.

Table 2 reports the average volume, runtime, number of points generated for each S-n-m over 10 trials. We also compute a 95% confidence interval for the volume. Notice that for all cases the extreme values of each interval imply an error ≤ 0.1, which was the requested error. For $n = 40$ just a few seconds suffice to approximate the volume and for $n = 100$ our implementation takes a few minutes.

4.2 Expected value of a function

Randomized algorithms are commonly used for problems in robust control analysis to overcome the (worst case) hardness, especially in probabilistic robustness [6, 10, 27, 50]. central problem is to approximate the integral of a function over a spectrahedron, e.g. [9, 46] and thus uniform sampling is of particular interest. To put our experiments into perspective, we present experiment up to $n = 100$, while in [9] and [10] they use only W-HnR for experiments in $n \leq 10$.

Our goal is to compute the expected value of a function $f : \mathbb{R}^n \to [0, 1]$, with respect to the measure given by the uniform distribution π over S, i.e., $I = \int_S f(x)\pi(x)dx$. A standard approach is the Monte Carlo method, which suggests to sample N independent samples from π. Then, the $\hat{\mathbb{E}}_N[f] = \frac{1}{N}\sum_{i=1}^{N} f(x_i)$ is an unbiased estimator for I. We employ the random walks of Sec. 3 to sample uniformly distributed points from S (i.e., W-HnR, W-CHnR, and W-Billard) and we experimentally compare their efficiency. It turns out that W-Billard mixes much faster and results to better accuracy.

The variance of an estimator is a crucial as it bounds the approximation error. Using Chebyshev's inequality and [35], we have

$$Prob[|\hat{\mathbb{E}}_N[f] - \mathbb{E}[f]| \leq \epsilon] \leq \frac{var(\hat{\mathbb{E}}_N[f])}{\epsilon^2} \leq \frac{4M_\epsilon}{N\epsilon^2}, \qquad (10)$$

where M_ϵ is the mixing time of the random walk one uses to sample "ϵ close" to the uniform distribution from S. Thus, for fixed N and ϵ, the smaller the mixing time of the random walk is, the smaller the variance of estimator $\hat{\mathbb{E}}_N[f]$ and hence, the better the approximation. We estimate I when $f := 1(S \cap H)$, where H is the union of two half-spaces $H := \{x \mid cx \leq b_1 \text{ or } cx \geq b_2\}$, where $b_2 > b_1$ and $1(\cdot)$ is the indicator function. To estimate $I = \text{vol}(S \cap H)/\text{vol}(S)$, we sample points from S and count points in $S \cap H$.

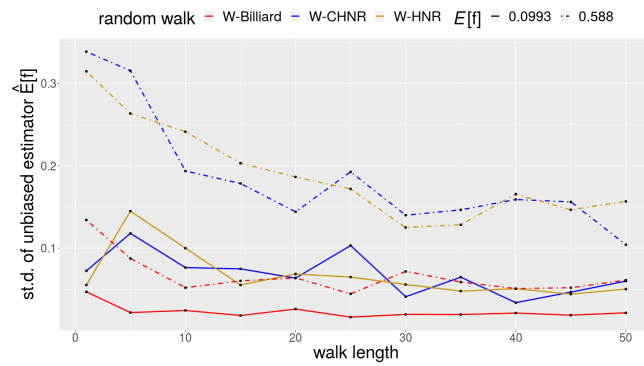

Figure 4: *The standard deviation of $\hat{\mathbb{E}}_N[f]$ over $M = 20$ trials, estimating 2 functions with $\mathbb{E}[f_1] = 0.0993$ and $\mathbb{E}[f_2] = 0.588$. For each walk length we sample $N = 200$ points and we repeat $M = 20$ times.*

walk length	1	5	10	20	30	40	50
S-100-100	1.4	3.2	7.7	9.5	16.1	21.4	28.2
S-200-200	16.2	75.7	148	303	443	584	722

Table 3: *Average time in sec to sample 200 points with W-Billard from 10 random spectrahedra S-n-m; n for the dimension that S-n-m lies; m for the dimension of the matrix in LMI.*

We estimate two functions with $\mathbb{E}[f_1] = 0.0993$ and $\mathbb{E}[f_2] = 0.5880$ in $n = 50$ and for various walk lengths. For each walk length we sample $N = 200$ points and we repeat $M = 20$ times. Then, for each N-set we compute $\frac{1}{N}\sum_{i=1}^{N} f(x_i)$ and we take the average and the standard deviation (st.d.) over M. Figure 4 (and Fig. 5 in Appendix) illustrates these values, while the walk length increases. Notice that the st.d. is much smaller and the approximation more stable when W-Billard is used compare to both W-HnR and W-CHnR. As W-Billard mixes faster, we report in Table 3 the average time our software needs to sample $N = 200$ points for various walk lengths for W-Billard in $n = 100,\ 200$. The average time to generate a point is ≈ 0.3 and ≈ 7.2 milliseconds respectively.

4.3 Sampling from non-uniform distributions

The random walks of Sec. 3 open a promising avenue for approximating the optimal solution of a semidefinite program, that is

$$\min\langle \boldsymbol{c}, \boldsymbol{x}\rangle, \text{ subject to } \boldsymbol{x} \in S. \qquad (11)$$

We parametrize the optimization algorithm in [25] with the choice of random walk and demonstrate that its efficiency relies heavily on the sampling method. We perform experiments with W-HMC-R and W-HnR, as both can sample from the distribution the algorithm requires. Deterministic approximations to the optimal solutions of these tests, were acquired via the SDPA library [55].

The strategy to approximate the optimal solution $\boldsymbol{x}^*$ of Eq. (11), is based on sampling from the Boltzmann distribution, i.e., $\pi(\boldsymbol{x}) \propto e^{-\boldsymbol{c}\boldsymbol{x}/T}$, truncated to S. The scalar T, is called *temperature*. As

S-n-m	W-HMC-R	W-HnR W_1	W-HnR W_2
S-30-30	20.1 / 2.9/ 0	184.3 / 3.4 / 1	52.1 / 5.2 / 0
S-40-40	24.6 / 7.9 / 0	223.3 / 9.9 / 2	61.9 / 17.1 / 0
S-50-50	29.2 / 12.7 / 0	251.2 / 22.3 / 3	72.3 / 44.6 / 0
S-60-60	32.8 / 24.32 / 0	272.7 / 41.1 / 3	81.5 / 98.9 / 0

Table 4: *The average #iteration / runtime / failures over* 10 *generated S-n-m, to achieve error $\epsilon \leq 0.05$. The walk length is one for W-HMC-R and $W_1 = 4\sqrt{n}$ and $W_2 = 4n$ for W-HnR. With "failures" we count the number of times the method fails to converge. Also m is the dimension of the matrix in LMI and n is the dimension that S-n-m lies.*

the temperature T diminishes, the mass of π tends to concentrate around its mode, which is $\boldsymbol{x}^*$. Thus, one could obtain a uniform point using the algorithm in [36], and then use it as a starting point to sample from $\pi_0 \propto e^{-cx/T_0}$, where $T_0 = R$ and $S \subseteq R\mathcal{B}_n$. Then, the cooling schedule $T_{i+1} = T_i(1 - 1/\sqrt{n})$ guarantees that a sample from π_i yields a good starting point for π_{i+1}. After $\widetilde{O}(\sqrt{n})$ steps the temperature will be low enough, to sample a point within distance ϵ from $\boldsymbol{x}^*$ with high probability.

In [25], they use only W-HnR. We also employ W-HMC-R. To sample from Boltzmann distributions with W-HMC-R, at each step, starting from $\boldsymbol{p_i}$ and with momenta $\boldsymbol{v_i}$, the ODE of Eq. (7) becomes

$$\frac{d^2}{dt^2}\boldsymbol{p}(t) = -\frac{\boldsymbol{c}}{T},\ \frac{d}{dt}\boldsymbol{p}(0) = \boldsymbol{v_i},\ \boldsymbol{p}(0) = \boldsymbol{p_i}. \tag{12}$$

Its solution is the polynomial $\boldsymbol{p}(t) = -\frac{\boldsymbol{c}}{2T}t^2 + \boldsymbol{v_i}t + \boldsymbol{p_i}$, which is a parametric representation of a polynomial curve, see Eq. (3).

In Table 4 we follow the cooling schedule described, after setting $T_0 \approx R$ and sampling the first uniform point with W-Billard. We give the optimal solution as input and we stop dropping T when an error $\epsilon \leq 0.05$ is achieved.

Even in the case when the walk length is set equal to one, W-HMC-R still converges to to the optimal solution. To the best of our knowledge, this is the first time that a randomized algorithm, which is based on random walks, is functional even when the walk length is set to one. On the other hand, we set the walk length of W-HnR $O(\sqrt{n})$ or $O(n)$ in our experiments. Notice that for the smaller walk length, its runtime decreases, but the method becomes unstable, as it sometimes fails to converge. For both cases its runtime in is worse than W-HMC-R. Furthermore, in Fig. 6 (Appendix) we illustrate a 50-dimensional example of Table 4, through 70 iterations, by setting a $O(n^2)$ walk length for W-HnR. W-HMC-R still converges faster than W-HnR, which implies much smaller mixing time.

REFERENCES

[1] H. M. Afshar and J. Domke. Reflection, Refraction, and Hamiltonian Monte Carlo. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, pages 3007–3015, Cambridge, MA, USA, 2015. MIT Press.

[2] D. Armentano and C. Beltrán. The polynomial eigenvalue problem is well conditioned for random inputs. *SIMAX*, 40(1):175–193, 2019.

[3] C. Beltrán and K. Kozhasov. The real polynomial eigenvalue problem is well conditioned on the average. *FoCM*, pages 1–19, 2019.

[4] M. Berhanu. *The polynomial eigenvalue problem.* PhD thesis, University of Manchester, 2005.

[5] M. Betancourt. A conceptual introduction to hamiltonian monte carlo, 2017.

[6] V. D. Blondel and J. N. Tsitsiklis. A survey of computational complexity results in systems and control. *Automatica*, 36(9):1249 – 1274, 2000.

[7] J.-D. Boissonnat and M. Teillaud. *Effective computational geometry for curves and surfaces.* Springer, 2006.

[8] B. Büeler, A. Enge, and K. Fukuda. Exact volume computation for polytopes: A practical study, 1998.

[9] G. Calafiore. Random walks for probabilistic robustness. In *Proc. CDC*, volume 5, pages 5316–5321. IEEE, 2004.

[10] G. Calafiore and M. Campi. Robust convex programs: Randomized solutions and applications in control. In *Proc. CDC*, volume 3, pages 2423–2428. IEEE, 2003.

[11] A. Chalkis, I. Z. Emiris, and V. Fisikopoulos. Practical volume estimation by a new annealing schedule for cooling convex bodies, 2019.

[12] Y. Chen, R. Dwivedi, M. J. Wainwright, and B. Yu. Vaidya walk: A sampling algorithm based on the volumetric barrier. In *2017 55th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1220–1227, Oct 2017.

[13] A. Chevallier, S. Pion, and F. Cazals. Hamiltonian Monte Carlo with boundary reflections, and application to polytope volume calculations. Research Report RR-9222, INRIA Sophia Antipolis, France, Nov. 2018.

[14] B. Cousins and S. Vempala. Bypassing KLS: Gaussian cooling and an $O^*(n^3)$ volume algorithm. In *Proc. ACM STOC*, pages 539–548, 2015.

[15] B. Cousins and S. Vempala. A practical volume algorithm. *Mathematical Programming Computation*, 8, June 2016.

[16] F. Dabbene, P. Shcherbakov, and B. T. Polyak. A randomized cutting plane method with probabilistic geometric convergence. *SIOPT*, 20:3185–3207, 2010.

[17] J.-P. Dedieu and F. Tisseur. Perturbation theory for homogeneous polynomial eigenvalue problems. *Linear algebra and its applications*, 358(1-3):71–94, 2003.

[18] M. Dyer, A. Frieze, and R. Kannan. A random polynomial-time algorithm for approximating the volume of convex bodies. *J. ACM*, 38(1):1–17, 1991.

[19] Eigen. Spectra. https://github.com/yixuan/spectra, 2015.

[20] I. Emiris and V. Fisikopoulos. Practical polytope volume approximation. *ACM Trans. Math. Soft.*, 44(4):38:1–38:21, 2018. Prelim. version: SoCG, 2014.

[21] I. Emiris, B. Mourrain, and E. Tsigaridas. Separation bounds for polynomial systems. *Journal of Symbolic Computation*, 2019.

[22] E. Gryazina and B. Polyak. Random sampling: Billiard walk algorithm. *European Journal of Operational Research*, 238, 11 2012.

[23] G. Guennebaud, B. Jacob, et al. Eigen v3, 2010. http://eigen.tuxfamily.org.

[24] D. Henrion, J. B. Lasserre, and C. Savorgnan. Approximate volume and integration for basic semialgebraic sets. *SIAM review*, 51(4):722–743, 2009.

[25] A. T. Kalai and S. Vempala. Simulated annealing for convex optimization. *Math. Oper. Res.*, 31(2):253–266, Feb. 2006.

[26] E. Kaltofen and G. Villard. On the complexity of computing determinants. *Computational complexity*, 13(3-4):91–130, 2005.

[27] P. Khargonekar and A. Tikku. Randomized algorithms for robust control analysis and synthesis have polynomial complexity. In *Proceedings of 35th IEEE Conference on Decision and Control*, volume 3, pages 3470–3475 vol.3, Dec 1996.

[28] M. Korda and D. Henrion. Convergence rates of moment-sum-of-squares hierarchies for volume approximation of semialgebraic sets. *Optimization Letters*, 12(3):435–442, 2018.

[29] P. Lairez, M. Mezzarobba, and M. Safey El Din. Computing the volume of compact semi-algebraic sets. In *Proc. ISSAC*. ACM Press, 2019.

[30] S. Lan, B. Zhou, and B. Shahbaba. Spherical Hamiltonian Monte Carlo for Constrained Target Distributions. *JMLR workshop and conf procs.*, 32:629–637, 2014.

[31] G. Lecerf. On the complexity of the Lickteig–Roy subresultant algorithm. *Journal of Symbolic Computation*, 92:243–268, May 2019.

[32] Y. T. Lee, Z. Song, and S. S. Vempala. Algorithmic theory of odes and sampling from well-conditioned logconcave densities, 2018.

[33] Y. T. Lee and S. Vempala. Convergence Rate of Riemannian Hamiltonian Monte Carlo and Faster Polytope Volume Computation. In *Proc. STOC*, 2018.

[34] T. Lickteig and M.-F. Roy. Sylvester–habicht sequences and fast cauchy index computation. *Journal of Symbolic Computation*, 31(3):315–341, Mar. 2001.

[35] L. Lovász, R. Kannan, and M. Simonovits. Random walks and an $O^*(n^5)$ volume algorithm for convex bodies. *Random Structures and Algorithms*, 11:1–50, 1997.

[36] L. Lovász and S. Vempala. Simulated annealing in convex bodies and an $O^*(n^4)$ volume algorithm. In *In Proc. FOCS*, volume 2003, pages 650–659, 2003.

[37] L. Lovasz and S. Vempala. Fast algorithms for logconcave functions: Sampling, rounding, integration and optimization. In *Proc. FOCS*, pages 57–68, 2006.

[38] J. Magnus and H. Neudecker. *Matrix Differential Calculus with Applications in Statistics and Econometrics (Revised Edition).* John Wiley & Sons Ltd, 1999.

[39] O. Mangoubi and N. K. Vishnoi. Faster polytope rounding, sampling, and volume computation via a sub-linear ball walk. In *Proc. FOCS*, pages 1338–1357, 2019.

[40] A. Pakman and L. Paninski. Exact Hamiltonian Monte Carlo for Truncated Multivariate Gaussians. *Journal of Computational and Graphical Statistics*, 23(2):518–542, Apr. 2014. Publisher: Taylor & Francis.

[41] V. Y. Pan. Univariate polynomials: nearly optimal algorithms for numerical factorization and root-finding. *J. of Symbolic Computation*, 33(5):701–733, 2002.

[42] V. Y. Pan and E. P. Tsigaridas. Nearly optimal refinement of real roots of a univariate polynomial. *Journal of Symbolic Computation*, 74:181–204, 2016.

[43] B. Polyak and E. Gryazina. Billiard walk - a new sampling algorithm for control and optimization. *IFAC Proceedings Volumes*, 47(3):6123 – 6128, 2014.

[44] B. Polyak and P. Shcherbakov. The d-decomposition technique for linear matrix inequalities. *Automation and Remote Control*, 67:1847–1861, 11 2006.
[45] M. Ramana and A. Goldman. Some geometric results in semidefinite programming. *Journal of Global Optimization*, 7, 02 1999.
[46] L. R. Ray and R. F. Stengel. A monte carlo approach to the analysis of control system robustness. *Automatica*, 29(1):229 – 236, 1993.
[47] A. Schönhage. The fundamental theorem of algebra in terms of computational complexity. *Manuscript. Univ. of Tübingen, Germany*, 1982.
[48] R. L. Smith. Efficient monte carlo procedures for generating points uniformly distributed over bounded regions. *Operations Research*, 32(6):1296–1308, 1984.
[49] S. Tabachnikov. *Geometry and billiards.* Student mathematical library. American Mathematical Society, Providence, RI, 2005.
[50] R. Tempo, G. Calafiore, and F. Dabbene. *Randomized algorithms for analysis and control of uncertain systems: with applications.* Springer Science, 2012.
[51] F. Tisseur. Backward error and condition of polynomial eigenvalue problems. *Linear Algebra and its Applications*, 309(1):339–361, 2000.
[52] F. Tisseur and K. Meerbergen. The quadratic eigenvalue problem. *SIAM review*, 43(2):235–286, 2001.
[53] L. N. Trefethen and D. Bau. *Numerical linear algebra.* SIAM, 1997.
[54] S. Vempala. Geometric random walks: A survey. *Combinatorial and Computational Geometry MSRI Publications Volume*, 52, 01 2005.
[55] M. Yamashita, K. Fujisawa, and M. Kojima. Implementation and evaluation of sdpa 6.0. *Optimization Methods and Software*, 18(4):491–505, 2003.

A ADDITIONAL PROOFS

To prove lemma 2.4 we will need the following lemmas.

Lemma A.1 (Partial Derivative of Determinant). *Let $\boldsymbol{A}$ be a symmetric $m \times m$ matrix. Then*

$$\frac{\partial \det \boldsymbol{A}}{\partial A_{ij}} = c_{ij}$$

where c_{ij} the cofactor of A_{ij}.

Proof. From Laplace expansion:

$$\det \boldsymbol{A} = \sum_{j=1}^{m} A_{ij} c_{ij}$$

Notice that $c_{1j}, \cdots, c_{mj}$ are independent of A_{ij}, so we have

$$\frac{\partial \det \boldsymbol{A}}{\partial A_{ij}} = c_{ij}$$

□

Lemma A.2. *Let $\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{A}_0 + x_1 \boldsymbol{A}_1 + \cdots + x_n \boldsymbol{A}_n$. Then*

$$\frac{\partial \det \boldsymbol{F}(\boldsymbol{x})}{\partial x_k} = Tr\left(\boldsymbol{F}(\boldsymbol{x})^* \boldsymbol{A}_k)\right)$$

Proof. The function $\det \boldsymbol{F}(\boldsymbol{x})$ is the composition of $\det \boldsymbol{A}$ and $\boldsymbol{A} = \boldsymbol{F}(\boldsymbol{x})$, so from Lemma A.1 and the chain rule:

$$\frac{\partial \det \boldsymbol{F}(\boldsymbol{x})}{\partial x_k} = \sum_{i=1}^{m}\sum_{j=1}^{m} \frac{\partial \det \boldsymbol{F}}{\partial \boldsymbol{F}_{ij}} \cdot \frac{\partial \boldsymbol{F}_{ij}}{\partial x_k} = \sum_{i=1}^{m}\sum_{j=1}^{m} c_{ij} A_{ij}^k = Tr\left(\boldsymbol{F}(\boldsymbol{x})^* \boldsymbol{A}_k\right)$$

where A_{ij}^k the ij-th element of matrix $\boldsymbol{A}_k$ □

Lemma A.3 (Adjoint Matrix of $\boldsymbol{A}$). *Let $\boldsymbol{A}$ be a $m \times m$ matrix of rank $r(\boldsymbol{A}) = m - 1$. Then*

$$\boldsymbol{A}^* = \mu(\boldsymbol{A}) \frac{\boldsymbol{v}\boldsymbol{u}^\top}{\boldsymbol{u}^\top \boldsymbol{v}}$$

where $\mu(\boldsymbol{A})$ is the product of the $m - 1$ non-zero eigenvalues of $\boldsymbol{A}$, and $\boldsymbol{x}$ and $\boldsymbol{y}$ satisfy $\boldsymbol{A}\boldsymbol{v} = \boldsymbol{A}^\top \boldsymbol{u} = \boldsymbol{0}$ (see chapter 3.2 in [38]).

Proof of Lemma 2.4. From Lemma A.2:

$$\frac{\partial \det \boldsymbol{F}(\boldsymbol{x})}{\partial x_k} = Tr\left(\boldsymbol{F}(\boldsymbol{x})^* \boldsymbol{A}_k\right) \tag{13}$$

If $\text{rank}(\boldsymbol{F}(\boldsymbol{x})) \leq -2$, then $\boldsymbol{F}(\boldsymbol{x})^*$ is the zero matrix. Supposing $\text{rank}(\boldsymbol{F}(\boldsymbol{x})) = m - 1$, from Lemma A.3:

$$Tr\left(\boldsymbol{F}(\boldsymbol{x})^* \boldsymbol{A}_k\right) = Tr\left(\mu(\boldsymbol{F}(\boldsymbol{x})) \frac{\boldsymbol{v}\boldsymbol{u}^\top}{\boldsymbol{u}^\top \boldsymbol{v}} \boldsymbol{A}_k\right) =$$
$$\frac{\mu(\boldsymbol{F}(\boldsymbol{x}))}{\boldsymbol{u}^\top \boldsymbol{v}} \cdot Tr\left(\boldsymbol{v}\boldsymbol{u}^\top \boldsymbol{A}_k\right) = \frac{\mu(\boldsymbol{F}(\boldsymbol{x}))}{\boldsymbol{u}^\top \boldsymbol{v}} \cdot \boldsymbol{u}^\top \boldsymbol{A}_k \boldsymbol{v}$$

but since $\boldsymbol{F}(\boldsymbol{x})$ is symmetric, we can choose $\boldsymbol{v} = \boldsymbol{u}$, so:

$$\frac{\mu(\boldsymbol{F}(\boldsymbol{x}))}{\boldsymbol{u}^\top \boldsymbol{v}} \cdot \boldsymbol{u}^\top \boldsymbol{A}_k \boldsymbol{v} = \frac{\mu(\boldsymbol{F}(\boldsymbol{x}))}{|\boldsymbol{v}|^2} \cdot \boldsymbol{v}^\top \boldsymbol{A}_k \boldsymbol{v}$$

□

Proof of Lemma 2.2. We construct $\boldsymbol{F}(\boldsymbol{p})$ in $O(nm^2)$. Then, with $O(m^{2.697})$ operations we compute its characteristic polynomial [26] and in $\widetilde{O}(m)$ we decide if it has negative roots, for example by solving [41] or using fast sub-resultant algorithms [31, 34]. For the bit complexity bound, the construction costs $\widetilde{O}_B(nm^2(\tau+\sigma)$ and computation of the characteristic polynomial $\widetilde{O}_B(m^{2.697+1}(\tau+\sigma))$ using a randomized algorithm [26]. We test for negative roots, and thus eigenvalues, in $\widetilde{O}_B(m^2n(\tau+\sigma))$ [34]. □

Proof of Lemma 2.3. The composition of $\boldsymbol{F}(\boldsymbol{x})$ and $\Phi(t)$ gives

$$\boldsymbol{F}(\Phi(t)) = \boldsymbol{A}_0 + p_1(t)\boldsymbol{A}_1 + \cdots + p_n(t)\boldsymbol{A}_n. \tag{14}$$

We rewrite (14) by grouping the coefficients for each t^k, $i \in [d]$, then

$$\boldsymbol{F}(\Phi(t)) = \boldsymbol{B}_0 + t\boldsymbol{B}_1 + \cdots + t^d\boldsymbol{B}_d, \tag{15}$$

where $\boldsymbol{B}_k = \sum_{j=0}^{n} p_{j,k}\,\boldsymbol{A}_j$, for $0 \le k \le d$. We use the convention that $p_{j,k} = 0$, when $k > d_j$.

For $t = 0$, it holds, by assumption, that $\boldsymbol{F}(\Phi(0)) = \boldsymbol{B}_0 \succ 0$; that is the point $\Phi(0)$ is in the interior S. Actually, for any $\boldsymbol{x} \in S^\circ$ it holds $\boldsymbol{F}(\boldsymbol{x}) \succ 0$. On the other hand, if $\boldsymbol{x} \in \partial S$, then $\boldsymbol{F}(\boldsymbol{x}) \succeq 0$. Our goal is to compute the maximal interval $[t_-, t_+]$ that contains 0 and for every t in it, we have $\boldsymbol{F}(\Phi(t)) \succeq 0$.

Starting from the point $\Phi(0)$, by varying t, we move at the trajectory that C defines (in both directions) until we hit the boundary of S. When we hit ∂S, the matrix $\boldsymbol{F}(\Phi(t))$ is not strictly definite anymore. Thus, its determinant vanishes.

Consider the function $\theta : \mathbb{R} \to \mathbb{R}$, where $\theta(t) = \det \boldsymbol{F}(\Phi(t))$ is a univariate polynomial in t. If a point $\Phi(t)$ is on the boundary of the spectrahedron, then $\theta(t) = 0$. We opt to compute t_- and t_+, such that $t_- \le 0 \le t_+$ and $\theta(t_-) = \theta(t_+) = 0$. At $t = 0$, $\theta(0) > 0$ and the graph of θ is above the t-axis. So C intersects the boundary when the graph of θ touches the t-axis for the first time at $t_1 \le 0 \le t_2$. It follows that $t_- = t_1$ and $t_+ = t_2$ are the maximum negative and minimum positive roots of θ, or equivalently the corresponding polynomial eigenvalues of $\boldsymbol{F}(\Phi(t))$. □

Proof of Lemma 2.1. We can compute the characteristic polynomial of an $N \times N$ matrix M in $\widetilde{O}_B(N^{2.697+1}\lg\|M\|)$ using a randomized algorithm, see [26] and references therein. Here $\|M\|$ denotes the largest entry in absolute value. In our case, the elements of $\boldsymbol{C}_2$ have bitsize $\widetilde{O}(\delta\tau)$ and its characteristic polynomial is of degree d and coefficient bitsize $\widetilde{O}_B(\delta^2\tau)$. We compute it in $\widetilde{O}_B(\delta^{2.697+1}\delta\tau) = \widetilde{O}_B(\delta^{4.697}\tau)$. We isolate all its real roots in $\widetilde{O}_B(\delta^5 + \delta^4\tau)$ [41]; they correspond to the real eigenvalues of PEP. We can decrease the width of the isolating interval by a factor of $\epsilon = 2^{-L}$ for *all* the roots in $\widetilde{O}_B(\delta^3\tau + \delta L)$ [42]. Thus, the overall complexity is $\widetilde{O}_B(\delta^5 + \delta^{4.697}\tau + \delta L)$.

It remains to compute the corresponding eigenvectors. For each eigenvalue λ we can compute the corresponding eigenvector $\boldsymbol{z}$ by performing Gaussian elimination and back substitution to the (augmented) matrix $[\lambda\boldsymbol{I}_\delta - \boldsymbol{C}_2 \mid \boldsymbol{0}]$. We can do this with $\widetilde{O}(\delta^\omega)$ arithmetic operations. However, as λ is a root of the characteristic polynomial we have to perform operations with algebraic numbers, which a highly non-trivial task and it is not clear what is the number of bits that we need to compute the elements of $\boldsymbol{z}$ correctly and to recover $\boldsymbol{x}$. For this task we employ the separation bounds for polynomial system adopted to the problem of eigenvector computation [21]. We need, as in the case of eigenvalues, $\widetilde{O}_B(\delta^4 + \delta^3\tau)$ bits to isolate the coordinates of the eigenvectors. To decrease the width of the corresponding isolating intervals by a factor of $\epsilon = 2^{-L}$, then the number of bits becomes $\widetilde{O}_B(\delta^4 + \delta^3\tau + L)$. Thus, we compute the eigenvectors in $\widetilde{O}_B(\delta^\omega(\delta^4 + \delta^3\tau + L)) = \widetilde{O}_B(\delta^{\omega+4} + \delta^{\omega+3}\tau + \delta^\omega L)$.

For the arithmetic complexity we proceed as follows: We compute the characteristic polynomial in $\widetilde{O}(\delta^{2.697})$, we approximate its roots up to ϵ in $\widetilde{O}(\delta\lg(1/\epsilon))$. Finally, we compute the eigenvectors with $\widetilde{O}(\delta^\omega)$ arithmetic operations. So the overall cost is $\widetilde{O}(\delta^{2.697} + \delta\lg(1/\epsilon))$. □

Proof of Lemma 2.5. Let $A \in \mathbb{R}^{m\times m}$ be a symmetric matrix and consider the map $f : \boldsymbol{v} \mapsto \boldsymbol{v}^T A\boldsymbol{v}$. The relative condition number of f as defined in [53, p. 90] is

$$k(\boldsymbol{v}) = \frac{||J(\boldsymbol{v})||}{||f(\boldsymbol{v})||/||\boldsymbol{v}||} = 2\frac{||A\boldsymbol{v}||}{\boldsymbol{v}^T A_i\boldsymbol{v}} = 2\frac{\sigma_{\max}(A)}{\sigma^2_{\max}(A)} = \frac{2}{\sigma_{\max}(A)}$$

where $J(\cdot)$ is the Jacobian of f. According to Theorem 15.1 in [53, p. 111], since matrix-vector multiplication is backward stable, the relative error of each coordinate in the gradient computation of Lemma 2.4 is $O(\frac{\epsilon_M}{\sigma_{\max}(A_i)})$, $i = 1, \ldots, n$. □

B MATRICES OF THE EXAMPLE

The spectrahedron was randomly generated as in [16]. Due to space considerations, the entries of the matrices are rounded to the first decimal.

$$\boldsymbol{A_0} = \begin{bmatrix} 16.7 & 3.7 & 12.3 & 8.7 & 5.1 & 10.4 \\ 3.7 & 9.4 & 2.3 & 4 & -2.3 & -1 \\ 12.3 & 2.3 & 26.8 & 18.7 & 7.1 & 16.7 \\ 8.7 & 4 & 18.7 & 20 & 3.7 & 12.3 \\ 5.1 & -2.3 & 7.1 & 3.7 & 6.1 & 5.4 \\ 10.4 & -1 & 16.7 & 12.3 & 5.4 & 18.7 \end{bmatrix} \tag{16}$$

$$\boldsymbol{A_1} = \begin{bmatrix} 0.5 & -0.4 & 2.7 & 0 & 0 & \\ -0.4 & 1.4 & -0.2 & 0 & 0 & 0 \\ 2.7 & -0.2 & 1.7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & -0.4 & 2.7 \\ 0 & 0 & 0 & -0.4 & 1.4 & -0.2 \\ 0 & 0 & 0 & 2.7 & -0.2 & 1.7 \end{bmatrix} \tag{17}$$

$$\boldsymbol{A_2} = \begin{bmatrix} 2.6 & -0.1 & 3 & 0 & 0 & 0 \\ -0.1 & 1 & -0.1 & 0 & 0 & 0 \\ 3 & -0.1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2.6 & -0.1 & 3 \\ 0 & 0 & 0 & -0.1 & 1 & -0.1 \\ 0 & 0 & 0 & 3 & -0.1 & -1 \end{bmatrix} \tag{18}$$

C ADDITIONAL FIGURES

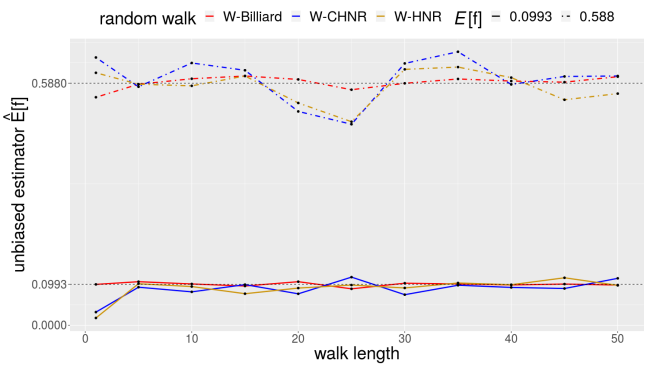

Figure 5: *The mean value of the estimator* $\hat{\mathbb{E}}_N[f]$ *over* $M = 20$ *trials, estimating two functions with* $\mathbb{E}[f_1] = 0.0993$ *and* $\mathbb{E}[f_2] = 0.588$. *For each walk length we sample* $N = 200$ *points and we repeat* $M = 20$ *times.*

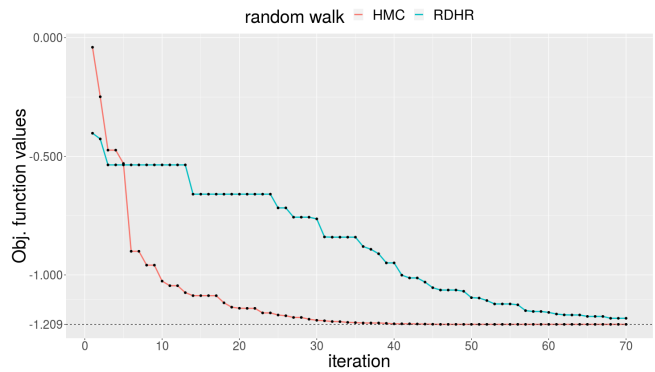

Figure 6: *Sample a point from* $\pi(x) \propto e^{-cx/T_i}$ *and update the objective current best in each iteration, with* $T_0 \approx diam(S)$ *and* $T_i = T_{i-1}(1 - 1/\sqrt{n})$, $i = 1, \ldots 70$. *The walk length equals to one for W-HMC-R and* $500 + 4n^2 = 10\,500$ *for W-HnR.*