# Hamiltonian Monte Carlo with boundary reflections, and application to polytope volume calculations

Augustin Chevallier, Sylvain Pion, Frédéric Cazals


# Hamiltonian Monte Carlo with boundary reflections, and application to polytope volume calculations

**Augustin Chevallier and Sylvain Pion and Frédéric Cazals**

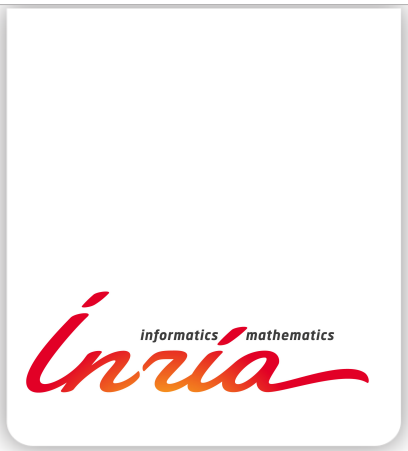
**RESEARCH REPORT**

**N° 9222**

November 2018

Project-Team Algorithms-Biology-Structure

ISSN 0249-6399 ISRN INRIA/RR--9222--FR+ENG




# Hamiltonian Monte Carlo with boundary reflections, and application to polytope volume calculations

Augustin Chevallier and Sylvain Pion and Frédéric Cazals

Project-Team Algorithms-Biology-Structure

Research Report n° 9222 — version 2 — initial version November 2018 — revised version May 2020 — 32 pages

**Abstract:** This paper studies HMC with reflections on the boundary of a domain, providing an enhanced alternative to Hit-and-run (HAR) to sample a target distribution in a bounded domain. We make three contributions. First, we provide a convergence bound, paving the way to more precise mixing time analysis. Second, we present a robust implementation based on multi-precision arithmetic – a mandatory ingredient to guarantee exact predicates and robust constructions. Third, we use our HMC random walk to perform polytope volume calculations, using it as an alternative to HAR within the volume algorithm by Cousins and Vempala. The tests, conducted up to dimension 50, show that the HMC RW outperforms HAR.

**Key-words:** HMC, polytope, random walk, importance sampling, billiard walk

RESEARCH CENTRE
SOPHIA ANTIPOLIS – MÉDITERRANÉE

2004 route des Lucioles - BP 93
06902 Sophia Antipolis Cedex

# Hamiltonian Monte Carlo avec réflexions, et application au calcul du volume de polytopes

**Résumé :** Ce papier étudie HMC avec réflexions au bord du domaine, donnant une meilleure alternative a Hit-and-Run (HAR) pour échantillonner une distribution cible dans un domaine borné. Nous apportons trois contributions. Premièrement, nous prouvons une borne de convergence, préparant le terrain pour une analyse plus précise du mixing time. Deuxièmement, nous produisons une implémentation robuste basée sur l'arithmétique multi-precision. Troisièmement, nous utilisons HMC avec réflexions comme une alternative à HAR pour calculer le volume de polytopes pour l'algorithme de Cousins et Vempala. Les tests, conduits jusqu'en dimension 50 montrent que HMC avec réflexions est plus performant que HAR.

**Mots-clés :** HMC, polytope, marche aléatoire, importance sampling, billiard

# Contents

# 1 Introduction

## 1.1 Sampling in high dimensional space: a pervasive challenge

**Sampling with MCMC algorithms.** Broadly speaking, Monte Carlo algorithms provide means to obtain numerical values from simulations resorting to randomness. Such algorithms have countless applications, as illustrated by the following examples. In statistical physics, macroscopic properties also called observables may be obtained from computer simulations generating ensembles of conformations over which averages are computed [1]. In numerical mathematics, a frequent goal is to generate point following a given target distribution [2]. In (Bayesian) statistics, a classical goal is the calculation of maximum a posteriori (MAP). Two classes of sampling techniques deserve a mention in this work. The first one is importance sampling [3], a generic strategy aiming at sampling rare events and reducing the variance of estimators by biasing the target distribution. The second one is Markov Chains Monte Carlo, where an ergodic Markov chain is used to target a given distribution [3]. A generic method in this realm, both conceptually elegant and remarkably generic is the Metropolis-Hastings method, where the Markov chain used consists in iteratively generating samples, and accepting/discarding them. Interestingly, this simple description suffices to capture the major difficulties of the method in high-dimensional spaces. When the target distribution allocates its mass on a *typical set* of small dimension, the samples generated should clearly remain in close vicinity of this set so as to avoid large rejection rates.

**Hamiltonian Monte Carlo.** An efficient solution to deal with such cases is Hamiltonian Monte Carlo (HMC) [4, 5]. HMC aims at sampling a distribution $\pi$ in a high dimensional space, by defining Markov chain in phase space $R^{2n}$, with $n$ coordinates for the position $q$, and $n$ coordinates for the velocity $p$. As the name suggests, HMC is governed by an ODE system defined by Hamilton's equations. In a nutshell, a HMC step involves three steps which are (i) picking a random velocity $p$, (ii) traveling deterministically the level set surface of the Hamiltonian, and (iii) projecting down in configuration space.

As seen from Hamilton's equation, the fact that the gradient of the target density is used to twist the momentum $p$ rather than the position $q$ helps forcing the dynamical system to be diffusive near the typical set [4]. A key difficulty though is the calculation of exact orbits, which we shall fudge around analytically in our case.

**Volume calculations and density of states.** Computing the volume of a polytope – a bounded region of $\mathbb{R}^n$ defined by the intersection of a fixed set of half spaces, is a classical problem in science and engineering. Unfortunately, exact volume calculation algorithms take an exponential time in the worst case [6]. A related problem to polytope volume calculations is the calculation of density of states (DoS) in statistical physics. To see why, recall that a partition function may be written as a sum or integral over energy levels of the DoS achieving a particular energy [1]. Therefore, a classical strategy to compute a partition function consists of estimating the volume in phase space of the pre-image of an energy level.

The case of polytopes is especially interesting since considerable efforts were devoted to complexity bounds, and the hardness results obtained calibrate the intrinsic difficulty of such problems in high dimensional spaces. Complexity-wise, it can be proved that the problem is #-P hard both for V-polytopes and H-polytopes [7]. Intuitively, any deterministic algorithm using a polynomial number of points and computing the corresponding convex hull omits an exponentially large fraction of the volume. This observation naturally calls for approximation algorithms [8, 9].

A major breakthrough was the development of the polynomial-time randomized algorithm by Dyer et al. [10]. More recently, it has been shown that $(\varepsilon, \delta)$ approximations of the volume could be computed in $O^\star(n^4)$ [11].

**Remark 1.** *To conform with previous work, the following notations are used in this paper:*

- $\varepsilon$*: criterion used to assess the quality of the volume approximation [12].*
- $\epsilon$*: notation used for the total variation bound on the mixing time. See theorems 3 and 5.*

The volume calculation boils down to estimating ratios in a telescoping product, each ratio being the integral over the convex of exponential functions carefully chosen according to a *cooling* schedule. The first function is sharply concentrated within the convex, while the last one is a flat distribution. A total of $O^\star(\sqrt{n})$ such functions are used. Each ratio in the telescoping product is estimated (with guarantees) using $O^\star(\sqrt{n})$ samples. The complexity of generating a given sample being $O^\star(n^3)$, the overall algorithm has complexity $O^\star(n^4)$.

The complexity was recently improved to $O^\star(n^3)$ [13, 14], using Gaussian rather than exponential functions. The improvement come from an enhanced cooling schedule, and a decreasing mixing time – $O^\star(\sigma^2 n^2)$ rather $O^\star(n^3)$ with $\sigma^2$ the variance of the sampled isotropic Gaussian.

Interestingly, these randomized algorithms implement a multi-phase MC strategy, and the estimation of individual terms in the telescoping product resort to importance sampling.

**Random walks and mixing properties.** The aforementioned randomized algorithms embark several key ingredients, the most prominent one being the strategy to generate random samples. Except in simple cases, the most generic approach is to use a random walk defining a Markov chain. Several such walks have been used, including ball walk [15], hit-and-run (HAR) [16, 17, 18, 19], billiard walk [20].

As discussed above for the case of MCMC algorithms, the goal of such a walk is to generate samples according to a target distribution. Of particular interest is HAR, since the method is amenable to several optimizations [21, 22], including the choice of the random line used, and the calculation of the facet of the polytope intersected by a line. In any case, the convergence is assessed by mixing properties, e.g. based on the total variation distance which measures how far the random walk is from its stationary distribution [23]. For the particular case of polytope volume calculations, both HAR and ball walk were particularly studied. HAR mixes in $O^\star(n^3)$ from any starting point, with constants depending on the geometry of the polytope. Ball-walk mixes in $O^\star(n^3)$, and requires a *warm-start* – certain hypotheses on the starting point are required. Finally, we also mention billiard walk, a random walk based on reflections on boundaries. While we are unaware of mixing time analysis, convincing experiments have been reported for the generation of uniform samples [24].

**Robustness issues.** The simplest computer model to use when designing numerical / geometric algorithms is the real RAM model which assumes that exact operations on real numbers are available at constant time per operation [25]. In practice, such assumptions are not valid, in particular due to rounding operations inherent to representations of floating point numbers in computers, and arithmetic operations are specified by the IEEE 754-2008 standard [26].

The case of geometric algorithms is particularly critical, since erroneous evaluation of expressions conditioning the branching of the algorithm typically yield situations where the calculation is no more coherent (the algorithm loops or crashes), or possibly terminates with an erroneous answer.

Such situations occur near degenerate situations, and manifest systematically even on the simplest expressions in 2D space [27]. The design of robust geometric algorithms can be done in

a general way using the Exact Geometric Computation paradigm [28], as it is done for example in the CGAL [29] software library. In order to do so, the CGAL kernel distinguishes between predicates and constructions. A predicate is a function whose output belongs to a finite set – e.g. boolean, or checks whether an expression is positive / negative / null. A construction exhibits a new numeric or geometric object (e.g. distance, point, vector) from pre-existing ones.

To ensure robustness, we develop our algorithms in this framework, using multi-precision number types whose accuracy can be increased on demand to ensure the correctness of predicates.

### 1.2 Contributions

This paper makes contributions touching upon random walks for MCMC algorithms, polytope volume calculations, and Hamiltonian Monte Carlo. More precisely:

1. In section 2, we analyse a novel random walk combining billiard walk and HMC. The random walk is designed to sample a target distribution, and piecewise smooth analytical trajectories can be obtained. For the particular case of a polytope $K$, intersection of the trajectory yields reflections inside $K$. Convergence properties are established.

2. In section 3, we instantiate our random walk to sample distributions used for polytope volume calculations. Analytical expressions for HMC trajectories are obtained. We also provide a robust implementation of the random walk based on multi-precision interval arithmetic.

3. Finally, section 4 reports experiments comparing HAR and our random walk. The first test samples a target distribution. The second one embeds our random walk into the practical polytope volume calculation of Cousins and Vempala [12]. In both cases, we show superior performances over HAR, for dimension up to $n = 50$.

## 2 Hamiltonian Monte Carlo method with boundary reflections

In this section we consider a bounded open set $Q \subset \mathbb{R}^n$ with piecewise smooth boundary and a probability measure with density $\pi : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ such that $\pi(x) = 0$ for all $x$ not in the closure $\overline{Q}$ of $Q$. We would like an algorithm sampling points according to $\pi$. The traditional HMC algorithm does not handle boundaries, hence we present a modification of HMC with reflections which is drawn from [30] and [20, 24].

### 2.1 HMC with reflections

Denoting $q^{(i)}$ and $p^{(i)}$ the i-th coordinates of position and momentum respectively, recall Hamilton's equations which state that the velocity field is orthogonal to the gradient of the Hamiltonian $H$:

$$\frac{dq^{(i)}}{dt} = \frac{\partial H}{\partial p^{(i)}}, \frac{dp^{(i)}}{dt} = -\frac{\partial H}{\partial q^{(i)}}. \tag{1}$$

As with HMC, we define a potential energy $U(q) = -\log(\pi(q))$ and the Hamiltonian $H(q,p) = U(q) + \frac{1}{2}\|p\|^2$ but this time restricted to $\Gamma = \overline{Q} \times \mathbb{R}^n$. We assume that $\pi$ is the restriction to $\overline{Q}$ of positive smooth function defined on $\mathbb{R}^n$ and we use $\Phi_t$ the Hamiltonian flow. However, the trajectories of this flow are not included in $\overline{Q}$ even if the initial point $(q,p)$ is in $Q \times \mathbb{R}^n$. For

every $q \in Q$ and every $p \in \mathbb{R}^n$, we define $T(q,p)$ as the largest $T$ such that for all $0 \leq t < T$, $\Phi_t(q,p) \in Q$. We also define $T(q,p) = 0$ when $q$ is in the boundary of $Q$.

Following [30] and [20, 24], we modify the flow by forcing reflections on the boundary of $Q$. This flow, illustrated on Fig. 2, is denoted as follow:

$$\begin{cases} \tilde{\Phi}_t : \overline{Q} \times \mathbb{R}^n \to Q \times \mathbb{R}^n \\ (q,p) \mapsto \tilde{\Phi}_t(q,p) \equiv (\tilde{\Phi}_t^{(q)}(q,p), \tilde{\Phi}_t^{(p)}(q,p)). \end{cases} \tag{2}$$

Note that the latter equation defines position and velocity upon applying the flow. However, as noted in [20, 24], this new flow might exhibit problematic trajectories: some of them might not be defined for all $t$ because of singularities on the boundary, others might have huge number of reflections or even an infinite number of reflections in finite time. Hence, following ideas of [20, 24], we cull problematic trajectories by not moving in those cases.

**Remark 2.** *For $q \in Q$ and any $p$, $T(q,p) > 0$ and $\tilde{\Phi}_{T(q,p)}(q,p)$ is in the boundary of $Q$.*

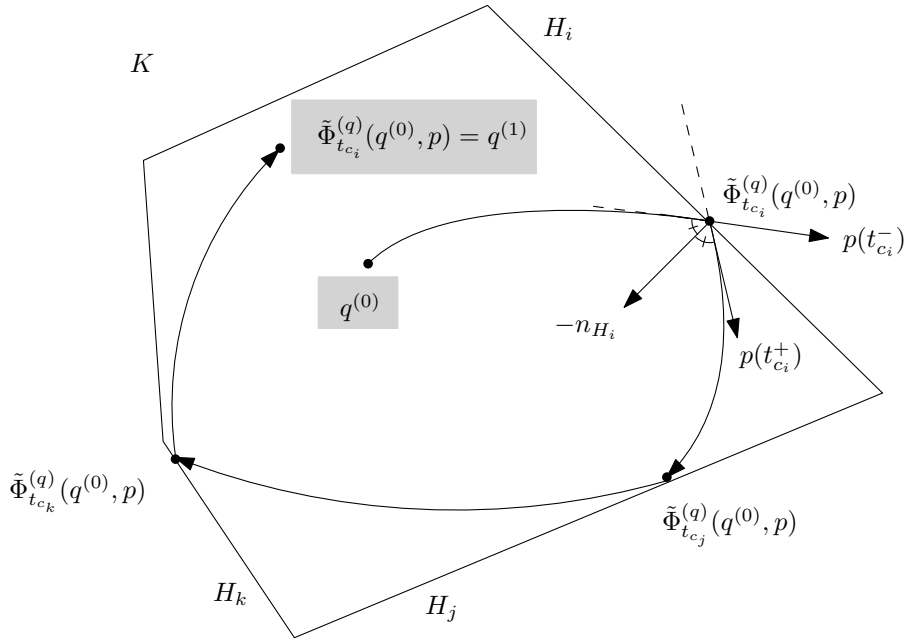
**Remark 3.** *When $q$ is in the boundary of $Q$ and $t > 0$, $\tilde{\Phi}_t(q,p)$ is only defined for the momenta $p$ such that the open half-line with origin $q$ and direction $p$, is included in $Q$ in a neighborhood of $q$.*

**Algorithm** Let $M \in \mathbb{N}^*$ be the maximum number of reflections. Given a point $q^{(t)} \in Q$, the algorithm is as follow (Fig. 1):

**Figure 1 Hamiltonian Monte Carlo with reflections.**

1. Choose the traveling time $L \sim \text{unif}(0,1)$
2. Pick the momentum $p \sim \mathcal{N}(0, I_n)$
3. If the flow $\tilde{\Phi}_L(q^{(t)}, p)$ is defined and does not involve more than $M$ reflections between $t = 0$ and $L$, and if $\tilde{\Phi}_L(q^{(t)}, p) \in Q$
   - Take $q^{(t+1)} = \tilde{\Phi}_L^{(q)}(q^{(t)}, p)$
   - Else, take $q^{(t+1)} = q^{(t)}$

**Figure 2 Reflection of the HMC trajectory on the boundary of the polytope $K$: evolution of a single HMC step, starting from $q^{(0)}$; the trajectory successively reflects on hyperplanes, before stopping at $q^{(n_L)}$.**




For a fixed $L > 0$, steps from 2. and 3. define a Markov kernel $P_{\pi,L}$. The full algorithm (steps from 1. to 3.) define a Markov kernel $P_\pi$ that can be expressed with $P_{\pi,L}$.

For $L > 0$, let $\Gamma_L$ be the largest subset of $\Gamma$ where $\tilde{\Phi}_L$ is defined, admits no more than $M$ reflections, and the trajectory do not finish in a singularity at time $L$. $\Gamma_L$ is open and therefore measurable. Let

$$\bar{\Phi}(q,p) = \begin{cases} \tilde{\Phi}_L(q,p) & \text{if } (q,p) \in \Gamma_L \\ (q,p) & \text{if } (q,p) \notin \Gamma_L \end{cases}$$

the application from $\Gamma$ to $\Gamma$ which corresponds to steps 3 and 4.

**Remark 4.** *For any point $(q,p) \in \Gamma$, if $L$ is small enough, $(q,p) \in \Gamma_L$. However, if $L$ is too large, $\Gamma_L$ could be empty.*

## 2.2 Measure invariance via detailed balance

We recall the definition of detailed balance:

**Definition 1** (detailed balance / reversible)**.** *A Markov chain $P$ is said to satisfy detailed balance (or reversible) with respect to the measure $\pi$ if for every $A$ and $B$ measurable,*

$$\int_B P(x,A)\pi(dx) = \int_A P(x,B)\pi(dx)$$

Let $A$ and $B$ be measurable subsets of $Q$. Then let

$$A_B = \{(q,p) \in \Gamma_L | q \in A, \bar{\Phi}(q,p) \in B \times \mathbb{R}^n\}$$

the subset of $\Gamma$ of all positions in $A$ with momenta that brings them in $B$ after time $L$. Similarly, we define

$$\Psi(q,p) = (\bar{\Phi}^{(q)}(q,p), -\bar{\Phi}^{(p)}(q,p))$$

on $\Gamma$.

**Lemma 1.** *The maps $\bar{\Phi}$ and $\Psi$ preserve the Lebesgue measure on $\Gamma_L$. ie for every $A \subset \Gamma_L$ measurable, $\lambda(\bar{\Phi}^{-1}(A)) = \lambda(A)$*

*Proof.* Clearly it is enough to prove that $\tilde{\Phi}_t$ preserves the Lebesgue measure. In [30], it is proved that for a fixed step size, the Euler method for numerical integration applied to the Hamiltonian flow with reflections, gives rise to a flow that preserves the Lebesgue measure. Letting the step size going to zero, we see that $\tilde{\Phi}_t$ is the pointwise limit of transformations that preserves the Lebesgue measure which implies that $\tilde{\Phi}_t$ preserve the Lebesgue measure. □

**Lemma 2.** *1. $\Psi(\Gamma_L) \subset \Gamma_L$*

*2. $\Psi \circ \Psi = I$ on $\Gamma_L$*

*3. For any measurable sets $A$ and $B$ of $\mathbb{R}^n$,*

$$B_A = \Psi(A_B)$$

*4. $H(\Psi(q,p)) = H(q,p)$ for all $(q,p) \in \Gamma$.*

*Proof.* 1. and 2. are simple consequences of the reversibility of the Hamiltonian flow with reflections.
3. Let $A$ and $B$ be measurable sets of $\mathbb{R}^n$.
$\Psi_q(A_B) \subset B$ and $\Psi(\Psi(A_B)) = A_B$ thus $\Psi(A_B) \subset B_A$.
By symmetry, $\Psi(B_A) \subset A_B$. Hence by composing with $\Psi$: $\Psi(\Psi(B_A)) \subset \Psi(A_B)$. Using 2., we deduce $B_A \subset \Psi(A_B)$.
4. is clear.

□

**Theorem 1.** *$P_{\pi,L}$ and $\pi$ satisfy detailed balance.*

*Proof.* Let $A$ and $B$ be measurable sets of $\mathbb{R}^n$. The left hand side of the detailed balance equation becomes

$$\begin{aligned}
\int_A P_{\pi,L}(q,B)d\pi(q) &= \int_A P_{\pi,L}(q,B)\exp(-U(q))dq \\
&= \int_A \Big[ \int_{\{p|(q,p)\in\Gamma_L, \bar{\Phi}^{(q)}(q,p)\in B\}} \exp(-\|p\|^2)dp \\
&+ \int_{\{p|(q,p)\notin\Gamma_L, \bar{\Phi}^{(q)}(q,p)\in B\}} \exp(-\|p\|^2)dp \Big] \exp(-U(q))dq \\
&= \int_{A_B} \exp(-H(q,p))dqdp + \int_A \int_{\{p|(q,p)\notin\Gamma_L\}} 1_B(q)\exp(-H(q,p))dqdp \\
&= \int_{A_B} \exp(-H(q,p))dqdp + \int_{A\cap B} \int_{\{p|(q,p)\notin\Gamma_L\}} \exp(-H(q,p))dqdp.
\end{aligned}$$

By symmetry:

$$\int_B P_{\pi,L}(q,A)d\pi(q) = \int_{B_A} \exp(-H(q,p))dqdp + \int_{A\cap B} \int_{\{p|(q,p)\notin\Gamma_L\}} \exp(-H(q,p))dqdp$$

Using lemma 2 and the measure conservation of lemma 1 we obtain:

$$\begin{aligned}\int_{A_B} \exp(-H(q,p))dqdp &= \int_{\Psi(B_A)} \exp(-H(q,p))dqdp \\ &= \int_{B_A} \exp(-H(\Psi(q,p)))dqdp \\ &= \int_{B_A} \exp(-H(q,p))dqdp\end{aligned}$$

Which concludes the proof. □

**Theorem 2.** *$P_\pi$ satisfies detailed balance with respect to $\pi$.*

*Proof.* this is a direct consequence of theorem 1 □

## 2.3 Convergence result

Detailed balance ensures that $\pi$ is invariant by $P_\pi$, but it does not imply the convergence of the $P^n$ to $\pi$ by itself. In this section, we prove additional results to get such a convergence result. This requires extra assumptions on $Q$.

We recall the following definition and theorem from [31].

**Definition 2.** *A subset $C \subset \mathcal{X}$ is small (or, $(n_0, \epsilon, \nu)$-small) if there exists a positive integer $n_0$, a real $\epsilon > 0$, and a probability measure $\nu(.)$ on $\mathcal{X}$ such that the following minorisation condition holds:*

$$P^{n_0}(x,.) \geq \epsilon\nu(.) \quad x \in C$$

*i.e. $P^{n_0}(x,A) \geq \epsilon\nu(A)$ for all $x \in C$ and all measurable $A \subset \mathcal{X}$*

Intuitively, the previous definition states that whatever the starting point $x$–whence the adjective small, the iterated kernels $P^{n_0}(x,.)$ cover a common measure $\nu(.)$. Note that the value of $n_0$ is the number of steps needed to reach $\nu$ – and can be equal to one.

**Theorem 3.** *Consider a Markov chain with invariant probability distribution $\pi(.)$. Suppose that $\mathcal{X}$ is small (i.e., the entire state space is small). Then the chain is uniformly ergodic, and in fact*

$$\|P^n(x,.) - \pi(.)\|_{TV} \leq (1-\epsilon)^{\lfloor n/n0 \rfloor} \tag{3}$$

*for all $x \in \mathcal{X}$, where $\lfloor r \rfloor$ is the greatest integer not exceeding $r$ and the norm is the total variation.*

The following theorem provides a sufficient condition for $Q$ to be small with respect to $P_\pi$.

**Theorem 4.** *If $Q$ if convex and the gradient of the potential energy $\nabla U$ is bounded on $Q$, then $Q$ is small for $P_\pi$.*

*Proof.* We assume without any loss of generality that $0 \in Q$.

Let $q^{(1)}, q^{(2)} \in Q$. One way to get a trajectory going from the point $q^{(1)}$ to a point very close to $q^{(2)}$, is to select a very high momentum $p_\alpha = \frac{1}{\alpha}(q^{(2)} - q^{(1)})$ and a short time $t_\alpha = \alpha$ with $\alpha > 0$. When $\alpha \to 0$, the potential energy term $U$ of the Hamiltonian becomes less and less relevant, thus the trajectory converges to a straight line and $\lim_{\alpha \to 0} \Phi^q_{t_\alpha}(q^{(1)}, p_\alpha) = q^{(2)}$.

This intuition is formalized using the scaled variables

$$\begin{cases} \tilde{q}(t) &= q(\alpha t) \\ \tilde{p}(t) &= \alpha p(\alpha t). \end{cases} \tag{4}$$

The equation of motion becomes:

$$\frac{d\tilde{q}}{dt}(t) = \tilde{p}(t) \tag{5}$$

$$\frac{d\tilde{p}}{dt}(t) = \alpha^2 \nabla_q U(\tilde{q}(t)). \tag{6}$$

which defines a flow $\phi(\alpha, t, q, p)$. It should be noted that $\phi(-\alpha, t, q, p) = \phi(\alpha, t, q, p)$ for every $\alpha, t, q, p$ and that $\phi$ is correctly defined for $\alpha = 0$. In this case, it is easy to see that:

$$\phi(0, t, q, p) = q + pt. \tag{7}$$

As $\pi$ is the restriction of a positive smooth function, $\phi$ is defined for every $(\alpha, t, q, p) \in [-1, 1] \times \mathbb{R}^+ \times \mathbb{R}^n \times \mathbb{R}^n$.

Furthermore, $\alpha^2 \nabla_q U$ is smooth. Hence, using the differentiability of the solutions of differential equations on parameters and initial conditions (see [32]), we see that $\phi$ is $C^2$ on $]-1, 1[ \times Q \times \mathbb{R}^+ \times \mathbb{R}^n$.

Since $Q$ is open, there exists $\rho > 0$ such that $B(0, 2\rho) \subset Q$. Let $\nu$ be the measure on $Q$ which is the Lebesgue measure on $B(0, \rho/2)$ and zero outside the ball $B(0, \rho/2)$.

Our aim is to show that there exists $\epsilon > 0$ such that for every $q \in Q$, $P_\pi(q, .) \geq \epsilon\nu(.)$. Note that we would only need that $P_\pi^{n_0}(q, .) \geq \epsilon\nu(.)$ for some $n_0$, but since $Q$ is convex we will be able to take $n_0 = 1$.

The density of the probability measure associated with momenta is $\exp(-1/2\|p\|^2)$, and taking into account the rescaling of the momenta, we define the probability density

$$\gamma(p) = \alpha \exp(-\frac{1}{2}\|\frac{1}{\alpha}p\|^2). \tag{8}$$

$Q$ is bounded, hence there exists $\epsilon > 0$ such that for every $(q, p) \in \{(q, p) | q \in Q, p \in B(-q, \rho)\}$,

$$\gamma(p) > \epsilon$$

Let

$$A = \{(\alpha, t, q, p) : |\alpha| \leq 1/2, t \in [1/2, 3/2],\ q \in \bar{Q},\ p \in B(-q, \rho)\}.$$

$A$ is compact, hence the first and second order derivatives of $\phi$ are bounded on $A$. Thus there exists $Z > 0$ such that for every $(\alpha, t, q, p) \in A$,

$$\|\phi^q(\alpha, t, q, p) - \phi^q(0, 1, q, p)\| \leq (\alpha + |t - 1|)Z \tag{9}$$

and

$$\|d_p\phi^q(\alpha, t, q, p) - d_v\phi^q(0, 1, q, p)\| \leq (\alpha + |t - 1|)Z. \tag{10}$$

Using equation 7, the two above inequalities are equivalent to

$$\|\phi^q(\alpha, t, q, p) - (q + p)\| \leq (\alpha + |t - 1|)Z \tag{11}$$

and

$$\|d_p\phi^q(\alpha, t, q, p) - Id_{\mathbb{R}^n}\| \leq (\alpha + |t - 1|)Z \tag{12}$$

The determinant is continuous, so there exists $0 < \beta < 1$ such that $\|d_p\phi^q(\alpha, t, q, p) - Id_{\mathbb{R}^n}\| < \beta$ implies

$$1/2 < |d_p\phi(\alpha, t, q, p)| < 2. \tag{13}$$

Finally, using Lemma 3 together with the fact that $\nabla U$ is bounded, we deduce that there exists $\alpha_0 > 0$ such that for every $0 < \alpha \leq \alpha_0$ and every $q \in Q$ and $p \in B(-q, \rho)$, the trajectory $\phi^q(\alpha, t, q, p)$ for $t \leq 1$ stays in $Q$. It follows that $\phi$ and $\Phi$ coincide, for there is no reflection.

Let $\alpha = \min(\frac{\rho}{4Z}, \frac{1}{2Z}, \frac{\beta}{2Z}, \alpha_0) > 0$. Let any $t \in [1 - \frac{\beta}{2Z}, 1]$ and $q$ be in $Q$. By lemma 4 below, the map

$$f : p \to \phi^q(\alpha, t, q, p)$$

is a $C^1$ diffeomorphism from $B(-q, \rho)$ to $V = \phi^q(\alpha, t, q, B(-q, \rho))$ and by Lemma 5 (applied to $f$ translated by $-q$), $B(0, \rho/2) \subset V$. Furthermore, equation 13 implies that for every $q' \in V$

$$|df^{-1}(q')| > 1/2. \tag{14}$$

Hence, the push forward measure $\xi$ of $\nu$ by $f$ is non zero on $B(0, \rho/2)$, and has a density

$$\xi(q') = \nu(f^{-1}(q'))|df^{-1}(q')| \geq \epsilon/2 \tag{15}$$

on $B(0, \rho/2)$. Hence, under the condition that the travel time $t$ is in $[1 - \frac{\beta}{2Z}, 1]$, we get the following transition probability:

$$P(q, .|t \in [1 - \frac{\beta}{2Z}, 1]) \geq \frac{\epsilon}{2}\nu(.)$$

For the random walk, $t$ is sampled uniformly in $[0, 1]$, hence

$$P(q, .) \geq \frac{\epsilon}{2}\frac{\beta}{2Z}\nu(.)$$

which concludes the proof. □

**Remark 5.** *The previous proof uses $n_0 = 1$. We believe it should be possible to extend the proof to non convex $Q$ by taking $n_0 > 1$, and some extra regularity assumptions on $Q$.*

### 2.3.1 Lemmas used in the proof

**Lemma 3.** *Let $Q$ be an open convex subset in $\mathbb{R}^n$ that contains the ball $B(0, 2\rho)$, let $x$ be a point in $Q$ and let $v$ be in $B(0, \rho) - x$. Let $f : \mathbb{R}^n \to \mathbb{R}^n$ be a continuous and bounded map. Suppose that $(x(t), v(t)) \in \mathbb{R}^{2n}$ is the solution of the Cauchy problem*

$$\begin{aligned} x(0) &= x \\ v(0) &= v \\ x'(t) &= v(t) \\ v'(t) &= af(t) \end{aligned}$$

*where $a$ is a real number. If $|a| \leq \frac{\rho}{\|f\|_\infty}$ then for all $t \in [0, 1]$, $x(t)$ is in the convex hull of $x$ and of the ball $B(0, 2\rho)$ and therefore in $Q$.*

*Proof.* Suppose $|a| \leq \frac{\rho}{\|f\|_\infty}$. By the mean value theorem, $v(t) = v + w(t)$ where $\|w(t)\| \leq |a|\|f\|_\infty t \leq \rho t$ for $t \geq 0$. The derivative of function $y(t) = x(t) - vt$ is $w(t)$, therefore by the mean value theorem, $y(t) = y(0) + tz(t)$ where $\|z(t)\| \leq \rho t/2$. Therefore for all $t \in [0, 1]$,

$$\begin{aligned} x(t) &= y(t) + vt \\ &= (1 - t)x + t((v + x) + z(t)) \end{aligned}$$

is in the convex hull of $x$ and of the ball $B(0, 2\rho)$. □

**Lemma 4.** *Let $B$ be an open ball in $\mathbb{R}^n$ and let $\varphi : B \to \mathbb{R}^n$ be a differentiable map. If for each $x$ in $B$, $\|d\varphi(x) - Id\| < 1$, then $\varphi$ is a diffeomorphism.*

*Proof.* The only thing to prove is that $\varphi$ is one to one. Let $x \neq y$ be two points in $B$. Consider the map $f : t \in [0, \|y-x\|] \to \varphi(x+tu) \cdot u$ where $u = \frac{y-x}{\|y-x\|}$. Using Schwarz inequality and the assumption we obtain

$$\begin{aligned} f'(t) &= d\varphi(x+tu)(u) \cdot u \\ &= Id(u) \cdot u + (d\varphi(x+tu) - Id)(u) \cdot u \\ &\geq 1 - \|d\varphi(x+tu) - Id\|\|u\|\|u\| \\ &> 0. \end{aligned}$$

It follows that $f(\|y-x\|) > f(0)$. Now $f(0) = \varphi(x) \cdot u$ and $f(\|y-x\|) = \varphi(y) \cdot u$, hence $\varphi(x) \neq \varphi(y)$. □

**Lemma 5.** *Let $B = B(0,r)$ be a closed ball in $\mathbb{R}^n$ of center $0$ and radius $r > 0$ and let $\varphi : B \to \mathbb{R}^n$ be a one to one continuous map. If for all $x$ in $B$, $d(x, \varphi(x)) \leq r/4$ then $\varphi(B)$ contains the ball $B(0, r/2)$.*

*Proof.* By Jordan-Brouwer Theorem, the complement in $\mathbb{R}^n$ of the image $\Sigma$ of the sphere $S = \partial B$ has exactly two connected components $C_1$ and $C_2$, one which is bounded, say $C_1$, and one which is not. By assumption the open ball $\overset{\circ}{B}(O, r/2)$ doesn't intersect $\Sigma$, hence is included in $C_1$ or $C_2$.

The image $E = \varphi(\overset{\circ}{B})$ of the interior of the ball $B$ is included in $C_i$, one of the two connected components of $\mathbb{R}^n \setminus \Sigma$. On the one hand, by Jordan-Brouwer invariance of the domain theorem, $E$ is open. On the other hand, $E = \varphi(B) \cap C_i$ and since $\varphi(B)$ is compact, $C_i \setminus E$ is open. Now $C_i$ is connected, hence $E$ or $C_i \setminus E$ is empty. Therefore $E = C_i$.

Since $\varphi(B)$ is compact, $E$ is bounded, and therefore $E = C_i = C_1$. Moreover, by assumption, $\varphi(0) \in B^o(0, r/2) \cap E$ which implies that $B^o(0, r/2) \subset E$. □

# 3 Application: computing the volume of a polytope

In this section, we specialize our generic algorithm (Algorithm 1) so as to use it as a building block of the polytope volume calculation from [12]. The corresponding pseudo-code is provided in the supplemental section 6.

## 3.1 Volume algorithm

Consider a polytope defined by linear inequalities $Ax \leq b$. The algorithm used in [12] computes the volume of such a polytope with target relative error $\varepsilon$. The principle is a multi-phase Monte Carlo computation, which splits the calculation into $m$ steps. Let $\{f_0, \ldots, f_{m-1}\}$ be $m$ isotropic Gaussian distributions i.e. $f_i(x) = \exp(-\|x\|^2/(2\sigma_i^2))$ with $\sigma_i = 1/\sqrt{2a_i}$, or equivalently $f_i(x) = \exp(-a_i\|x\|^2)$, such that the first one is highly concentrated around a point deep inside the convex, and the last one is an almost flat distribution. The volume calculation reduces to computing the telescoping product

$$\mathrm{Vol}(K) = \int_K f_0(x)dx \frac{\int_K f_1(x)dx}{\int_K f_0(x)dx} \cdots \frac{\int_K dx}{\int_K f_{m-1}(x)dx} \equiv \int_K f_0(x)dx \prod_{i=1,\ldots,m} R_i \qquad (16)$$

Each of these ratio are estimated using Monte-Carlo integration with respect to the density

$$\pi_i(x) = \frac{f_i(x)}{\int_K f_i(y)dy} 1_K(x) \tag{17}$$

via importance sampling. The method then uses as core block an algorithm sampling the previous distribution, usually using HAR. In our case, HMC with reflections will be used instead. For $X_1, ..., X_k$ consecutive points given by the random walk, the Monte-Carlo estimation $R_i^k$ is given by:

$$R_i^{(k)} = \frac{1}{k} \sum_{j=1}^{k} \frac{f_i(X_j)}{f_{i-1}(X_j)}. \tag{18}$$

Instead of using a fixed number of points for the Monte-Carlo integration, a stopping criterion is introduced by [12]. Let $\varepsilon' = \varepsilon/\sqrt{m}$; this is the relative ratio error allocated for each ratio $R_i$ estimation. Consider a sliding window of size $W(= 4n^2 + 500)$ ($n$ is the dimension). When $W$ consecutive estimated ratios $R_i^{(k-W+1)}, ..., R_i^{(k)}$ are within $\varepsilon'/2$, the convergence for $R_i$ is declared.

## 3.2 HMC algorithm

This specialization has two advantages: first, trajectories have analytic expressions – see also [33]; second, the intersection between a trajectory and $n$ hyperplanes has a simple analytical expression.

**Analytical trajectories.** As recalled in section 3.1, importance sampling is used to estimate the ratios $R_i$. Hence, we build an HMC method to sample from $\pi_i(x)$ from Eq. (17).

Let $U(q) = \log(\exp(-a_i||q||^2)) = -a_i||q||^2$ and $H(q,p) = U(q) + 1/2\|p\|^2$. Note that the normalization constant $\frac{1}{\int_K f_i(y)dy}$ was discarded because it does not change the trajectory (its gradient is 0). Rewriting the dynamical system associated to this Hamiltonian yields the following differential equations:

$$\frac{d^2q_j}{dt^2}(t) = -2a_i q_j(t) \text{ for } j \leq n. \tag{19}$$

Each coordinate is independent and has a solution of the form

$$q_j(t) = C_j \cos(wt + \phi_j) \tag{20}$$

With $C_j, w, \phi_j \in \mathbb{R}$. The parameter $\Phi_j$ satisfies the following 2 equations:

$$\begin{cases} \cos(\phi_j) & = \frac{q_j(0)}{C_j} \\ \sin(\phi_j) & = \frac{-p_j(0)}{\omega C_j} \end{cases} \tag{21}$$

Thus we deduce:

$$\begin{cases} \omega & = \sqrt{2a_i} \\ C_j & = \sqrt{q_j(0)^2 + p_j(0)^2/w^2} \\ \phi_j & = \arctan\left(-\frac{p_j(0)}{q_j(0)\omega}\right) + 1_{\{q_j(0)<0\}}\pi \end{cases} \tag{22}$$

It should be noted that these equations are the same for any choice of coordinates as long as the basis is orthonormal. This allows us to choose a basis suited to the computations we want to do.

**Collision with convex boundary.** To restrict the sampling algorithm to the convex $K$, trajectories should reflect on the boundary of $K$. The convex $K$ is defined by a set of hyperplanes. Hence, we need to compute the intersection time of a trajectory with each hyperplane and take the smallest time. Thankfully, there is an analytical expression. The hyperplanes are defined by a matrix $A$ and a vector $b$, and hyperplane $i$ is defined by the equation

$$(Ax)_i = b_i. \tag{23}$$

To compute the collision time with an hyperplane $H$, we make the following remark: let $n_H$ be the normal of the hyperplane. We can complete $n_H$ to an orthonormal basis. In this basis, the collision time depends only on what happens for the coordinate on $n_H$. Consider $q_{n_H}(t) =< q(t), n_h >$ and $p_{n_H}(t) =< p(t), n_H >$ the coordinates along the normal of the hyperplane. Let $\omega_{n_H}$,$C_{n_H}$ and $\phi_{n_H}$ be the parameters of the trajectory along direction $n_H$. Finding the intersection times is equivalent to solving the equation for $t$:

$$C_{n_H} \cos(\omega_{n_H} t + \phi_{n_H}) = b_i \tag{24}$$

We deduce that if $|C_{n_H}| < b_i$, there is no solution, and else, the following times are solution:

$$\begin{cases} t_1 &= \left(\arccos(b_i/C_{n_H}) - \phi_{n_H}\right)/\omega_{n_H}. \\ t_2 &= \left(-\arccos(b_i/C_{n_H}) - \phi_{n_H}\right)/\omega_{n_H}. \end{cases} \tag{25}$$

One solution corresponds to the entry into $K$, while the other corresponds to the exit out of $K$. We select the exit trajectory via a dot product between the velocity at $t_1$ and $t_2$ and the outward normal $n_H$. In the sequel, the corresponding value is denoted $t_c$ for time of collision.

### 3.3 Travel time choice with respect to $a_i$

The algorithm described in Fig.1 can be slightly generalised by choosing a travel time $L$ uniformly in $[0, L_{max}]$ instead of $[0, 1]$. We propose here a strategy to chose $L_{max}$ with respect to the parameter $a_i$ of the Gaussian we are trying to sample ($\sigma = 1/\sqrt{2a_i}$).

We distinguish two main cases here. First, if $a_i$ is large, then the probability measure is concentrated in a neighbourhood of 0, and the trajectories will seldom hit the boundaries and the strategy is largely unaffected by the convex. Second, if $a_i$ is close to 0, the distribution is close the uniform distribution of the convex.

**Case 1.** If $a_i$ is large, we consider $q_j(t)$ be a solution of the dynamical system 19 and we introduce space-time rescaling for the trajectory:

$$\bar{q}_j(t) = \sqrt{a_i} q_j(t/\sqrt{a_i}).$$

It satisfies eq. 19 with $a_i = 1$:

$$\frac{d^2 \bar{q}_j}{dt^2}(t) = \bar{q}_j(t)$$

with initial condition

$$\frac{d\bar{q}_j}{dt}(0) = \frac{dq_j}{dt}(0) \sim N(0, 1). \tag{26}$$

It follows that in the absence of convex boundaries, the rescaled process $\bar{q}$ is sampling the distribution associated to $a_i = 1$, i.e. the standard normal distribution. Therefore, any sensible value for $L_{max}(1)$ taken for $a_i = 1$ should be scaled as $L_{max}(a_i) = L_{max}(1)/\sqrt{a_i}$.

**Case 2.** When $a_i$ goes to 0, the previous strategy leads $L_{max}^{a_i} \to \infty$. We argue that in this case, the trajectories converges to billiard trajectories with reflections on the boundary and that the optimal $L_{max}$ should be therefore close to the optimal $L_{max}$ for $a_i = 0$. However, this optimal value has not been studied to the best of our knowledge.

**Values used in experiments.** Wwe chose the following $L_{max}$:

$$\begin{cases} L_{max}(a_i) & = 1/\sqrt{a_i} \quad \text{if } a_i > 1 \\ & = 1 \quad \text{if } a_i \leq 1 \end{cases} \tag{27}$$

## 3.4 HMC implementation based on interval arithmetic

### 3.4.1 Robustness issues

The algorithm as stated before is prone to numerical rounding errors. As a particular case, one may consider the situation where rounding errors would be such that the point computed on the HMC trajectory would be outside the convex. More generally, all geometric constructions and geometric predicates on them potentially raise robustness issues – see list in section 3.4.3.

As discussed in Introduction, we guarantee robustness using the Exact Geometric Computation paradigm. More specifically, recall that efficient arithmetic operations usually combine two ingredients: first, an interval representation of the numbers, as non overlapping intervals yield exact predicates; second, an arbitrary precision representation of the interval bounds, as precision can be increased so as to yield exact predicates and constructions of controlled accuracy. In the sequel, we use the `iRRAM` library which provides these two ingredients [34].

For the sake of clarity, all functions for which the `iRRAM` library plays a key role are highlighted in blue.

### 3.4.2 `iRRAM` and used features

We represent points as d-dimensional points whose coordinates are of the `iRRAM` number type. In `iRRAM`, a real number is represented by two types of data: firstly a symbolic representation memorizing the way it was defined (type of function and pointers to the operands), and secondly a numeric approximation using an interval with rational endpoints guaranteed to enclose the exact real value. The accuracy of the latter interval can be increased if needed, by recomputing it using recursively increased precision of the operands intervals that defines it. Therefore, for a real number $t$, the `iRRAM` encoding $q^{\text{iRRAM}}(t)$ of the position $q(t)$ of the HMC trajectory is numerically represented by a d-dimensional box certified to contain the exact real position.

The `iRRAM` number type enjoys two specific operations which are key in our implementation:

- operator $x < y$: predicate answering the $<$ comparison operator. If the interval representations of $x$ and $y$ overlap, these intervals are automatically refined, a feature called *precision refinement* thereafter.
- Near_inf_double(iRRAM $x$) : returns the nearest double $<$ the `iRRAM` number $x$.

### 3.4.3 Robust operations

Our robust implementation calls for the following operations (i) Computing the trajectory parameters – Eq. (22), (ii) Finding the exit intersection time – Eq. (25), (iii) Finding the smallest exit intersection time amongst all hyperplanes, and (iv) Evolving the trajectory. In the sequel,

we detail these operations, and refer the reader to the SI section 6; in particular, Algorithm 2 refines Algorithm 1 based on these robust primitives.

**Trajectory parameters.** Following Eq. (22), we construct the numbers $C_j$, $w$ and $\phi_j$ as `iRRAM` number types. See Algorithm 3.

**Exit intersection time $t_c$ with one hyperplane.** Intersecting the trajectory with a hyperplane yields two solutions (Eq. 25) respectively exiting and entering the convex. The collision time $t_c$ corresponds to the former. Assuming that the normal vector to the hyperplane is oriented outwards, the value $t_c$ is such that $< p(t_c), n_H >> 0$. The evaluation of this predicate triggers a precision refinement if needed.

**Smallest exit intersection time.** Since the boundary of $K$ involves several hyperplanes, the nearest one, which corresponds to the smallest exit time, must be determined. To do so, we first construct the intersection time $t_{c_i}$ with respect to each hyperplane. Then we compute $t_c = \min_i t_{c_i}$.

This calculation is tantamount to sorting the individual intersection times, which in turn requires the comparison operator $<$. `iRRAM` provides such an operator, which triggers precision refinement if needed. See Algorithm 4.

**Remark 6.** *We note that in case the trajectory would hit a face of dimension $< d-1$, an equality between exit times occurs. The absence of separation bound in* `iRRAM` *does not allow us to handle such cases, and an infinite refinement loop is entered. However, for each starting point, the measure of velocities leading to such sets is null. Practically, such a case was never faced–as expected.*

**The Is_strictly_in_convex($q$) predicate.** To constrain the trajectory within the convex, we resort to a predicate telling whether a given position $q$ belongs to the interior $K^{\mathrm{o}}$ of $K$. This predicate, denoted Is_strictly_in_convex(iRRAM_point_d $p$), checks $< op, n > < b_i$ holds for every hyperplane, and triggers the `iRRAM` refinement if needed so. (Nb: $< \cdot, \cdot >$ stands for the dot product of two vectors.)

**Performing one HMC step.** Equipped with the previous operations, our robust implementation – Algorithm 2, hinges on two operations:

- Calling the predicate Is_strictly_in_convex($q(t_c^<)$). Recall that in `iRRAM`, a d-dimensional point is represented as a box. This is in particular the case for the collision points with the hyperplanes, and for the final point returned. To ensure that all such points are strictly within the convex, we call the aforementioned Is_strictly_in_convex() .
- Computing the nearest inferior double $t_c^< =$ Near_inf_double($t_c$). The intersection point between the trajectory and a hyperplane is defined analytically – Eq. (25). The `iRRAM` representation of the collision time is an interval certified to contain the exact solution, and the corresponding d-dimensional point $q^{\mathrm{iRRAM}}(t_c^<)$ is represented as a box. We note that the box $q^{\mathrm{iRRAM}}(t_c^<)$ intersects the interior $K^{\mathrm{o}}$ of the convex $K$. Indeed:
  - the exact collision point $q(t_c)$ lies on its defining hyperplane i.e. $q(t_c) \in H_i$
  - by definition of $t_c^<$, the exact embedding $q(t_c^<)$ satisfies $q(t_c^<) \in K^{\mathrm{o}}$
  - the `iRRAM` box $q^{\mathrm{iRRAM}}(t_c^<)$ corresponding to $t_c^<$ intersects $K^{\mathrm{o}}$ since

$$q^{\mathrm{iRRAM}}(t_c^<) \ni q(t_c^<) \in K^{\mathrm{o}}$$

### 3.5 Cube: HMC mixing time is $O(\log n)$

We make a small digression in the case of the cube. It turns out that in this special case, the mixing time of the HMC random walk is $O(\log n)$ with $n$ the dimension, and the average complexity per step (the average number of calls to the oracle per step) is linear with the dimension. We assume without any loss of generality that the cube is $[0,1]^n$. Rigorously:

**Theorem 5.** *Let $P_k^{(n)}$ the distribution after $k$ steps in dimension $n$ and $g_n$ an isotropic Gaussian of parameter $a_i$ restricted to $[0,1]^n$. Then there exists $0<\rho<1$ such that for every $\epsilon>0$, every $n>1$ and every $x\in\mathbb{R}^n$, we have for $k\geq(\log n-\log\epsilon)/(\log 1/\rho)$:*

$$\|P_k^{(n)}(x,.)-g_n\|_{TV}\leq\epsilon. \tag{28}$$

*where the norm is the total variation. Furthermore, the average number of reflections per step is $O(n)$.*

*Proof.* We consider the canonical basis of $\mathbb{R}^n$. As shown before in Eq. (20), the trajectory coordinates associated to the Gaussian are all independent from each other. Furthermore, when a reflection with a boundary occurs, it means that one of the coordinates reached 0 or 1. The reflection simply switches the sign of the momentum for this coordinate, leaving other coordinates unchanged. Finally, the initial momentum vector is sampled according to $p(0)\sim\mathcal{N}(0,I_n)$, therefore each coordinate of $p(0)$ is sampled from an independent Gaussian $\mathcal{N}(0,1)$ in $\mathbb{R}$.
Hence we conclude that each coordinate has the behavior of a 1-dimensional HMC random walk sampling a 1-dimensional Gaussian, all independent from each other.

Let us consider the 1-dimensional random walk for a given Gaussian. We write $P_k(x,.)$ the distribution after $k$ steps starting from $x\in[0,1]$, and $g$ the probability density associated with the restriction of the Gaussian to $[0,1]$. Using theorem 3 combined with theorem 4 for the 1-D Gaussian restricted to $[0,1]$, we deduce that there exists $0<\rho<1$ such that for all $x\in[0,1]$,

$$\|P_k(x,.)-g\|_{TV}\leq\rho^k.$$

Observe that writing $P_k^{(n)}$ the distribution after $k$ steps in dimension $n$ and $g_n$ the Gaussian restricted to $[0,1]^n$, we have

$$\begin{cases}P_k^{(n)}(x,y)=P_k(x_1,y_1)P_k^{(n-1)}((x_2,...,x_n),(y_2,...,y_n)),\\ g_n(x)=g(x_1)g_{n-1}((x_2,...,x_n))\end{cases} \tag{29}$$

The total variation distance can be written as

$$\begin{aligned}
\|P_k^{(n)}(x,.) - g_n\|_{TV} &= \frac{1}{2}\int_{[0,1]^n} \left|P^{(n)}(x,y) - g_n(y)\right| dy \\
&= \frac{1}{2}\int_{[0,1]\times[0,1]^{n-1}} \left|P_k(x,y_1)P_k^{(n-1)}(x,y_2) - g(y_1)g_{n-1}(y_2)\right| dy_1 dy_2 \\
&= \frac{1}{2}\int_{[0,1]\times[0,1]^{n-1}} \left|(P_k(x,y_1) - g(y_1) + g(y_1))\, P_k^{(n-1)}(x,y_2) - g(y_1)g_{n-1}(y_2)\right| dy_1 dy_2 \\
&\leq \frac{1}{2}\int_{[0,1]\times[0,1]^{n-1}} \left|(P_k(x,y_1) - g(y_1))\, P_k^{(n-1)}(x,y_2)\right| dy_1 dy_2 \\
&+ \frac{1}{2}\int_{[0,1]\times[0,1]^{n-1}} \left|g(y_1)P_k^{(n-1)}(x,y_2) - g(y_1)g_{n-1}(y_2)\right| dy_1 dy_2 \\
&\leq \frac{1}{2}\int_{[0,1]} |P_k(x,y_1) - g(y_1)|\, dy_1 \\
&+ \frac{1}{2}\int_{[0,1]^{n-1}} \left|P_k^{(n-1)}(x,y_2) - g_{n-1}(y_2)\right| dy_2
\end{aligned}$$

We deduce

$$\|P_k^{(n)}(x,.) - g_n\|_{TV} \leq n\|P_k(x,.) - g\|_{TV} \leq n\rho^k \tag{30}$$

Hence for a fixed $\epsilon$, if $k$ satisfies $n\rho^k \leq \epsilon$, then for every $x$, $\|P_k^{(n)}(x,.) - g_n\|_{TV} \leq \epsilon$. Thus we take $k \geq (\log n - \log\epsilon)/(\log 1/\rho)$, and the mixing time is $O(\log n)$.

In addition, as each coordinate is from each other, the total number of reflections is the sum of reflections per coordinate. Hence, the number of reflections is proportional to the dimension. □

Using the previous strategy for the choice of the maximum travel time, we formulate the following conjectures:

**Conjecture on the mixing time.** There exists a constant $C$ such that for every isotropic Gaussian associated to $a_i$, such that for every $\epsilon > 0$, every $n > 1$ and every $x \in \mathbb{R}^n$, we have for $k \geq C(\log n - \log\epsilon)$

$$\|P_k^{(n)}(x,.) - g_n\|_{TV} \leq \epsilon$$

where the norm is the total variation. Furthermore, the average number of reflections per step is $O(n)$.

**Conjecture on the complexity.** The algorithm from [13, 14] generates one point from a warm start with complexity $O^*(n^2\sigma)$ complexity ($O^*$ means ignoring *log* terms). With $O(n)$ phases – each with a constant number of points, one gets an overall complexity of $O^*(n^3)$.

With our random walk, the number of steps is $O^*(1)$ to generate one point, with each step costing $O^*(n)$ calls to oracle. This leads in our case to a total complexity of $O^*(n^2)$.

# 4 Experiments

## 4.1 Implementation and code availability

**Implementation.** The implementation of our algorithm for the particular case of a convex polytope $Q = K$ is provided in the Structural Bioinformatics Library (SBL, `http://sbl.inria.`

`fr`), a state-of-the-art environment targeting molecular simulation at large [35]. The corresponding package, Hamiltonian_Monte_Carlo is ascribed to the *Core / Geometry-Topology* component of the library. The user manual of the package, as well as a jupyter notebook illustrating the main functionalities, can be accessed from `https://sbl.inria.fr/doc/Hamiltonian_Monte_Carlo-user-manual.html`.

**Robustness.** As explained in section 3.4.2, the main number type used to ensure robustness is the `iRRAM` number type. However, for the particular case of a polytope, we also need to check that the starting point belongs to the interior of $K$. (For a starting point on the boundary, one would have to check that the initial velocity is such that the trajectory enters the interior of the convex.) Since `iRRAM` does not have separation bounds to decide equality to zero, to perform this initial check, we instantiate the predicate Is_strictly_in_convex using the number type `CGAL::Lazy_exact_nt<CGAL::Quotient<CGAL::MP_Float»`.

We stress in passing that from dimension 10 onward, repeated runs using doubles systematically yield a significant fraction of point outside the convex, leading to crashes.

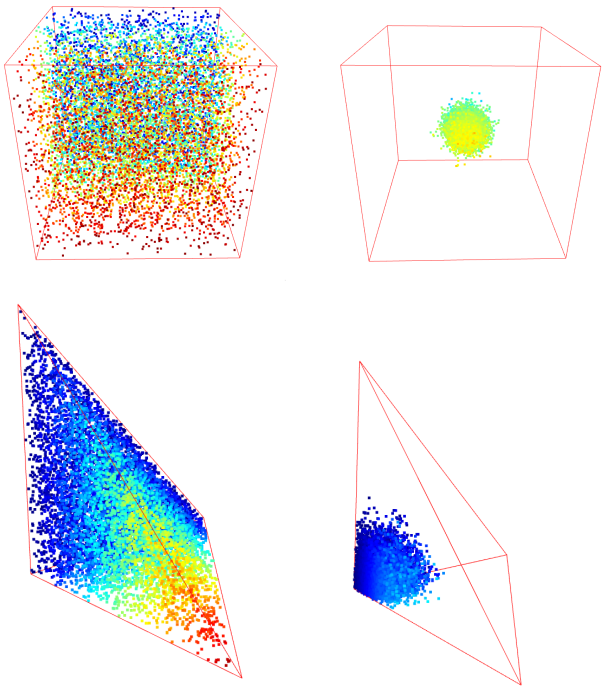
**Volume calculations.** As described in section 3, we also embed our random walk in the framework of [12]. We reuse the MATLAB code provided by [12] and adapt it so as to call our HMC random walk instead of the usual HAR random walk.

## 4.2 Illustrations of the HMC random walk

Our first illustration features samples generated by HMC for the cube $[-1, 1]^3$ and the standard simplex $\sum x_i \leq 1$ in dimension three. Varying $\sigma$ rapidly yields concentrated samples (Fig. 3).

Our second illustration shows the ability of the algorithm to escape corners. As opposed to Hit-And-Run, HMC indeed escapes corners much faster, yielding an an almost uniform distribution after 10 steps even when the dimension increases (Fig. 4).

**Figure 3 HMC for the cube $[-1,1]^3$ and the standard simplex in dimension 3: role of $\sigma$.** First column: $\sigma = 1$; second column: $\sigma = 0.01$. In all cases, $n = 10,000$ samples.



## 4.3 Analysis

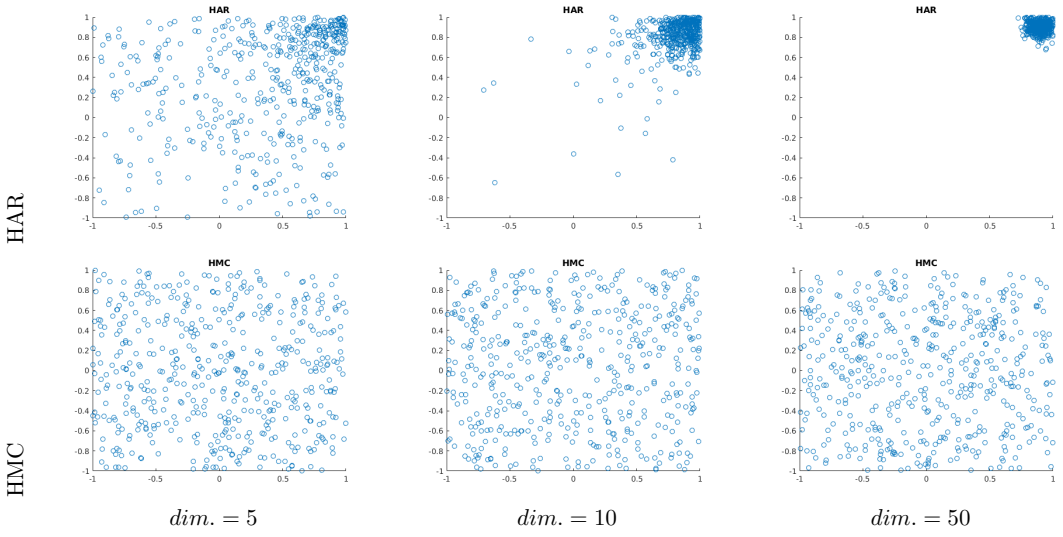
### 4.3.1 Volume computation

The goal of this experiment is twofold: first estimate the complexity (i.e. the number of calls to the oracle) with respect to the dimension. Then study the influence of the choice of the maximum travel time for HMC. The MATLAB implementation of the volume computation algorithm from [12] introduced the stop criterion of Eq. (18). Intuitively, the sliding window size $W$ should be at least as large as the mixing time of the chosen random walk. In the case of HAR, the mixing time increases with the dimension, hence the value of $W$ chosen by [12] depends on the dimension: $W = 500 + 4n^2$. However, in the HMC case, we hope for a smaller mixing time and especially a smaller growth rate with respect to the dimension. Therefore, the growth rate of $W$ used in [12] might be too large and impede the convergence speed for HMC. For that reason, we modified the MATLAB code to allow for different $W$.

**Statistics.** We collect the following statistics:

- the relative error $\mid V - \mathrm{Vol}(K) \mid / \mathrm{Vol}(K)$ with $V$ the estimated volume as a function of the dimension.
- Number of sampled points for a single volume computation.
- Complexity, i.e. the number of calls to the oracle. For HAR, this is equal to the number of sampled points. For HMC, it takes the number of reflections into account.

**Figure 4 HMC for the cube $[-1,1]^n$: ability to escape corners, and comparison to Hit-and-run.** For $n > 3$, the plot displays projections of the first two coordinates. Simulations were started from a corner ($q_i^{(0)} = 0.9$). Samples generated after 10 steps of HAR or HMC, repeated 500 times – whence 500 samples. A nearly flat isotropic Gaussian distribution ($\sigma = \sqrt{500}$) was used to approach the uniform distribution.




- for HMC, the average number of calls to the oracle per point sampled.

**Parameters used.**

- Window size. Our experiments cover the following cases:
  - dimension independent: $W = 10, 30$.
  - dimension dependent: $W = 30 + 4\sqrt{n}, 30 + 4n, 30 + 4n^{1.5}, 30 + 4n^2$.
- Maximum travel time for HMC. Our experiments cover travel times between $t_{min}$ the radius of the inscribed ball of the convex and $t_{max}$ the diameter of the convex.
- Target error $\varepsilon = 0.1$ – see Section 3.1 and [12]. Note that to comply a targeted error $\varepsilon$, we will show that the of the window size is crucial.

**Experiments.** The overall goal of the experiments is to determine an empirical growth rate of the complexity with respect to the dimension for the algorithm with HMC and compare with HAR. Ideally, one would try every parameters for a set of dimensions, then study the optimum parameters for each dimension, deduce the scaling of these parameters with the dimension. However this requires a very large number of simulations. Hence we settle for two experiments with one parameter fixed for each of them:

- (Experiment 1) we fix the maximum travel time at a reasonable value (as we will see in (Experiment 2) ). Then we plot the error and complexity for dimensions 10...50 with different stopping criterion $W$. Then we select suitable values for $W$ and use them to estimate the complexity growth rate with the dimension.

- (Experiment 2) we fix the stopping criterion ($W$) and vary the maximum travel time for several dimensions.

#### 4.3.2 Models

There are many available polytopes to study. We make the choice to select a few representatives ones in terms of difficulties we want to handle and then present a detailed analysis. First, we choose the cube, as it has good theoretical properties while being a difficult case for the HAR. Second, we choose the simplex, as it has sharp angles and is a widely used convex shape. For the comparison with HAR, we take the isotropic simplex since it is already rounded. However for the travel time experiment, we take the standard simplex for the simplicity of it's geometric features such as the diameters of the circumscribed and inscribed spheres.

#### 4.3.3 Running times

Volume calculations presented thereafter were performed on a laptop computer. In dimension 50, each individual calculation computation time is of the order of 10 seconds, so that running times are not further analyzed.

### 4.4 Tests on volume calculations

#### 4.4.1 Complexity analysis – stopping criterion

We ran the volume computation 50 times for each dimension using different values for $W$. From section.3.5, we recall that the mixing time of the HMC random walk is known in the case of the cube and is $O(log(n))$. Our intuition for the value of $W$ defining the stop criterion is that it is linked to the mixing time of the random walk. Hence we expect that using $W = cst$ for HMC on the cube would lead to a very slow growth of the error with the dimension. Since the maximum dimension is 50 and $log(50) \approx 1.7$, we do not expect to see the effect of the log with our dimension range. In addition, we expect super logarithmic values of $W$ to yield a relative error decrease when the dimension increases.

Since the algorithm is targeting a relative error $\varepsilon$, we wish to identify values of $W$ yielding a constant relative error whatever the dimension. With that in mind, for HMC, we expect to eliminate values of $W$ for which the error would decrease with the dimension. On the contrary, for HAR, we expect to eliminate values of $W$ for which the error increases with the dimension, since $W$ might not grow as fast as the mixing time. We test both the cube (Fig. 5) – a model for which HMC is well understood, and the isotropic simplex (Fig.6).

To study the complexity i.e. the number of calls to the oracle, we perform a linear regression on the complexity curves (Figs. 5, 6, bottom plots). Remarkably, all correlation coefficients obtained were superior to 0.997. The complexity growth rates are summarized in Fig.7.

As expected, the error explodes for HAR when $W$ is too small, so that plausible values for $W$ are $W = 30 + 4n^{1.5}$ and $W = 30 + 4n^2$. For the cube, the complexities are $O^*(n^{2.08})$ and $O^*(n^{2.45})$ respectively. If the original choice of $W$ with $n^2$ is correct [12], we see a $O(n^{0.55})$ improvement over the theoretical complexity of $O^*(n^3)$.

For HMC, the error clearly decreases for $W = 30 + 4n^{1.5}$ and $W = 30 + 4n^2$, but we cannot firmly discard any other $W$ since no clear increase or decrease is apparent. For the cube, we expected that the error would decrease for any super logarithmic value of $W$. We conjecture that we cannot see the decrease in relative error because of a too small dimension range. Another possibility would be a too aggressive cooling schedule as in [12], but we believe this unlikely.

The experimental complexity for the cube using $W = cst$ is $O(n^{1.27})$, while the theoretical complexity is $O^*(n^2)$, representing an improvement of $O(n^{0.73})$, slightly larger than for HAR. For the simplex, the results are overall similar.

**Figure 5 Cube: relative errors (top) and complexities i.e. number of calls to the oracle(bottom) for volume computation – HAR (left) vs HMC (right)** with the maximum travel time fixed at 1 for HMC. All quantities are averaged over 50 runs.

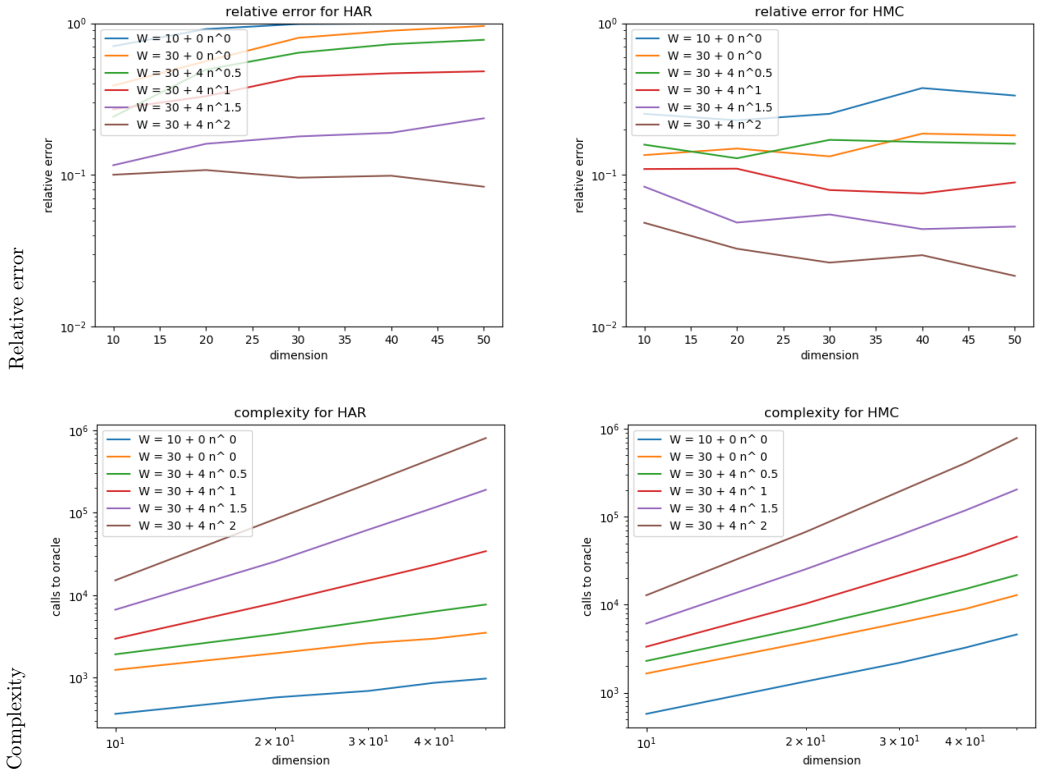


**Figure 6 Iso simplex: relative errors (top) and complexities i.e. number of calls to the oracle (bottom) for volume computation – HAR (left) vs HMC (right)** with the maximum travel time fixed at 1 for HMC.All quantities are averaged over 50 runs

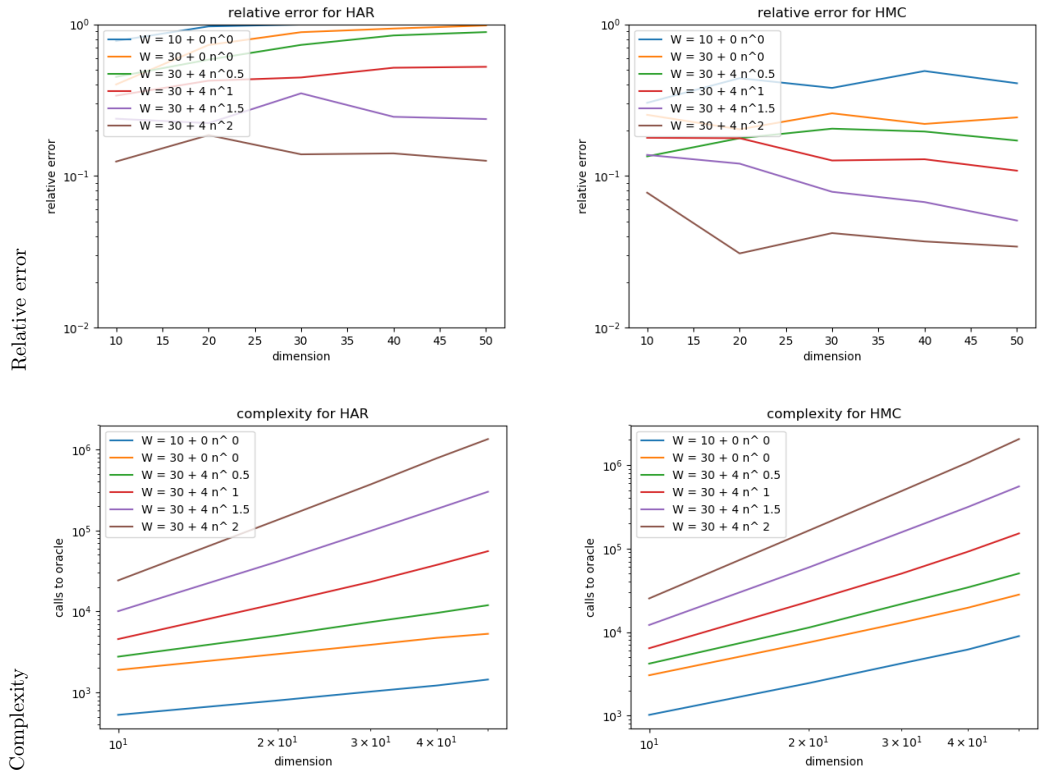



**Figure 7 Scaling of the complexity with the dimension, computed using the same data as Fig.5 and Fig.6.** Plausible complexities are highlighted in yellow. Exponents for the complexity growth rates were obtained with a linear regression on the complexity curves of Figs. 5, 6 – see main text.

| Window size | complexity | | | |
|---|---|---|---|---|
| | HMC | | HAR | |
| | Cube | Iso Simplex | Cube | Iso Simplex |
| $W = 10$ | $O(n^{1.27})$ | $O(n^{1.33})$ | $O(n^{0.60})$ | $O(n^{0.1})$ |
| $W = 30$ | $O(n^{1.25})$ | $O(n^{1.36})$ | $O(n^{0.64})$ | $O(n^{0.64})$ |
| $W = 30 + 4n^{0.5}$ | $O(n^{1.39})$ | $O(n^{1.54})$ | $O(n^{0.86})$ | $O(n^{0.90})$ |
| $W = 30 + 4n^{1}$ | $O(n^{1.77})$ | $O(n^{1.95})$ | $O(n^{1.54})$ | $O(n^{1.51})$ |
| $W = 30 + 4n^{1.5}$ | $O(n^{2.17})$ | $O(n^{2.36})$ | $O(n^{2.08})$ | $O(n^{2.11})$ |
| $W = 30 + 4n^{2}$ | $O(n^{2.54})$ | $O(n^{2.71})$ | $O(n^{2.46})$ | $O(n^{2.50})$ |

# 5 Conclusion

Sampling a target distribution is a central problem in science and engineering. A core difficulty is to deal with high dimensional spaces, and strategies based on random walks proved instrumental both to gain theoretical understanding and develop effective methods. In this context, this paper makes three contributions.

First, we develop novel insights regarding Hamiltonian Monte Carlo (HMC) based strategies with reflections on boundaries. These strategies leverage the properties of billiard trajectories which escape corners easily contrarily to Hit and Run. Moreover, our intuition is that the effectiveness of the HMC strategies lies in reflections, which are instrumental to decorrelate points and therefore decrease the mixing time. We provide a detailed proof of detailed balance for HMC with reflections (which was not used before even for HMC without reflections) and the well connectedness of the random walk, leading to a convergence bound.

Second, in the particular case of polyhedral domains we present a robust implementation based on multi-precision arithmetic. This ingredient is mandatory to guarantee exact predicates and robust constructions, following the traditional terminology in robust geometric computations.

Third, we use our HMC random walk for polytope volume calculations, using it as an alternative to the celebrated Hit-and-run (HAR) random walk used in the practical volume algorithm by Cousins and Vempala. The tests, conducted up to dimension 50, show that the HMC random walk outperforms the HAR random walk.

Our work clearly leaves stimulating questions open, two of which are of prominent importance. The first one is the choice of the optimal travel time required to sample a convex uniformly, which would ideally be determined at runtime for each convex. The second one is the analysis of the incidence of reflections on the mixing time, so as to quantify the speed at which reflections decorrelate successive points.

**Acknowledgments.** We are grateful to B. Cousins and S. Vempala [12] for providing a high quality MATLAB code.

# References

[1] D. Landau and K. Binder. *A guide to Monte Carlo simulations in statistical physics*. Cambridge university press, 2014.

[2] C. Robert and G. Casella. *Monte Carlo statistical methods*. Springer Science & Business Media, 2013.

[3] Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng. *Handbook of Markov Chain Monte Carlo*. CRC press, 2011.

[4] M. Betancourt. A conceptual introduction to Hamiltonian Monte Carlo. *arXiv preprint arXiv:1701.02434*, 2017.

[5] N. Radford. MCMC using Hamiltonian Dynamics. In Galin L. Jones Steve Brooks, Andrew Gelman and Xiao-Li Meng, editors, *Handbook of Markov Chain Monte Carlo*, chapter 5, pages 113–162. Chapman & Hall/CRC, 2012.

[6] B. Büeler, A. Enge, and K. Fukuda. Exact volume computation for polytopes: a practical study. In *Polytopes – combinatorics and computation*, pages 131–154. Springer, 2000.

[7] M. Dyer and A. Frieze. On the complexity of computing the volume of a polyhedron. *SIAM Journal on Computing*, 17(5):967–974, 1988.

[8] I. Bárány and Z. Füredi. Computing the volume is difficult. *Discrete & Computational Geometry*, 2(4):319–326, 1987.

[9] S. Levy. *Flavors of Geometry*. Cambridge University Press, 1997.

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

[11] László L. Lovász and S. Vempala. Simulated annealing in convex bodies and an $O^*(n^4)$ volume algorithm. *Journal of Computer and System Sciences*, 72(2):392–417, 2006.

[12] B. Cousins and S. Vempala. A practical volume algorithm. *Mathematical Programming Computation*, 8(2):133–160, 2016.

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

[14] B. Cousins and S. Vempala. Gaussian cooling and $O^*(n^3)$ algorithms for volume and gaussian volume. *SIAM Journal on Computing*, 47(3):1237–1273, 2018.

[15] L. Lovász and R. Kannan. Faster mixing via average conductance. In *Proceedings of the Thirty-first Annual ACM Symposium on Theory of Computing*, STOC '99, pages 282–287, New York, NY, USA, 1999. ACM.

[16] L. Lovász. Hit-and-run mixes fast. *Mathematical Programming, Series B*, 86:443–461, 12 1999.

[17] L. Lovász and S. Vempala. Hit-and-run is fast and fun. *preprint, Microsoft Research*, 2003.

[18] L. Lovász and S. Vempala. Hit-and-run from a corner. In *Proceedings of the Thirty-sixth Annual ACM Symposium on Theory of Computing*, STOC '04, pages 310–314, New York, NY, USA, 2004. ACM.

[19] H. Haraldsdóttir, B. Cousins, I. Thiele, R. Fleming, and S. Vempala. CHRR: coordinate hit-and-run with rounding for uniform sampling of constraint-based models. *Bioinformatics*, 33(11):1741–1743, 2017.

[20] E. Gryazinaa and B. Polyak. Random sampling: Billiard walk algorithm. *arXiv*, 2014.

[21] I. Emiris and V. Fisikopoulos. Efficient random-walk methods for approximating polytope volume. In *Proceedings of the thirtieth annual symposium on Computational geometry*, page 318. ACM, 2014.

[22] I. Emiris and V. Fisikopoulos. Practical polytope volume approximation. *ACM Trans. on Math. Software*, 44(4), 2018.

[23] D. Levin and Y. Peres. *Markov chains and mixing times*, volume 107. American Mathematical Soc., 2017.

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

[25] F. Preparata and M. Shamos. *Computational geometry: an introduction*. Springer Science & Business Media, 1985.

[26] J.-M. Muller, N. Brunie, F. de Dinechin, C. Jeannerod, M. Joldes, V. Lefèvre, G. Melquiond, N. Revol, and S. Torres. Handbook of floating-point arithmetic. 2018.

[27] L. Kettner, K. Mehlhorn, S. Pion, S. Schirra, and C. Yap. Classroom examples of robustness problems in geometric computations. *Computational Geometry*, 40(1):61–78, 2008.

[28] C. Yap and T. Dubé. The exact computation paradigm. In *Computing in Euclidean Geometry*, pages 452–492. World Scientific, 1995.

[29] CGAL, Computational Geometry Algorithms Library. http://www.cgal.org.

[30] H. Afshar and J. Domke. Reflection, refraction, and Hamiltonian Monte Carlo. In *Advances in Neural Information Processing Systems*, pages 3007–3015, 2015.

[31] G.O. Roberts and J.S. Rosenthal. General state space Markov chains and MCMC algorithms. *Probability Surveys*, 2004.

[32] James V. Burke. Continuity and differentiability of solutions. `https://sites.math.washington.edu/~burke/crs/555/555_notes/continuity.pdf`.

[33] A. Pakman and L. Paninski. Exact hamiltonian Monte Carlo for truncated multivariate gaussians. *Journal of Computational and Graphical Statistics*, 23(2):518–542, 2014.

[34] N. Müller. The iRRAM: Exact arithmetic in C++. In *Computability and Complexity in Analysis*, pages 222–252. Springer, 2001.

[35] F. Cazals and T. Dreyfus. The Structural Bioinformatics Library: modeling in biomolecular science and beyond. *Bioinformatics*, 7(33):1–8, 2017.

# 6 Supporting information: pseudo-code

In the following, we provide the pseudo-code for the high level description of the HMC algorithm (Fig. 1).

**Algorithm 1 Hamiltonian Monte Carlo step with real RAM arithmetic model**

```
1: HMC_step(q)
2: Choose a travel time L ~ Unif(0, L_max).
3: choose p ~ N(0, I_n)
4: Set dist = L
5: while dist > 0 do
6:    (intersection, t_c) ← Intersect_hyper_planes(q, p) // find intersection with hyperplanes
7:    if intersection = False OR dist < t_c then
8:       (q, p) = Update_positions_momenta(q, p, dist) // update traj. with distance dist
9:       Set dist = 0
10:   else
11:      (q, p) = Update_positions_momenta(q, p, t_c)
12:      Reflext_normal(p(t_c), n_c)
13:      Set dist = dist − t_c
```

**Algorithm 2 Hamiltonian Monte Carlo step with `iRRAM` number type**

```
1: HMC_step(q)
2: Choose a travel time L ~ Unif(0, L_max).
3: choose p ~ N(0, I_n) iRRAM REAL
4: Set dist = L
5: while dist > 0 do
6:    (intersection, t_c) ← Intersect_hyper_planes(q, p) with t_c an iRRAM REAL.
7:    t_c^< = Near_inf_double(t_c)
8:    if intersection = False OR dist < t_c^< then
9:       (q, p) = Update_positions_momenta(q, p, dist) // update trajectory with distance dist
10:   else
11:      (q, p) = Update_positions_momenta(q, p, t_c^<)
12:      Reflext_normal(p(t_c^<), n_c)
13:      Set dist = dist − t_c^<
14:   Is_strictly_in_convex(q)
15:   Return q
```

**Algorithm 3 Find trajectory parameters for a given direction.**

Compute_traj_params($q_{dir}, p_{dir}$)
Set $\omega = \sqrt{2a}$
Compute $C = \sqrt{q_{dir}^2 + p_{dir}^2/w^2}$
Compute $\phi = \arctan(-\frac{p_{dir}}{q_{dir}\omega})$
**if** $p_{dir} < 0$ and $\phi < 0$ **then**
  $\phi = \phi + \pi$
**if** $p_{dir} > 0$ and $\phi > 0$ **then**
  $\phi = \phi - \pi$
Return $[\omega, C, \phi]$

**Algorithm 4 Intersecting the trajectory with hyperplanes bounding the polytope**

```
Intersect_hyper_planes(q, p)
Set intersection = False
for each hyperplane H of equation (Ax)_i = b_i do
  Compute the outward pointing normal n_H to the hplane
  Compute the dot products q_{n_H} =< q, n_H > and p_{n_H} =< p, n_H >
  [ω, C, Φ] = Compute_traj_params(q_{n_H}, p_{n_H})
  if C > b_i then
    t1 = (arccos(b_i/C) − φ) /ω
    if t1 < 0 then
      t1 = t1 + 2π/ω
    t2 = (− arccos(b/C) − φ) /ω
    if t2 < 0 then
      t2 = t2 + 2π/ω
    t = min(t1, t2)
    if intersection = False then
      Set t_c = t
      Set t_c = n_H
      Set intersection = True
    else
      if t < t_c then
        Set t_c = t
        Set n_c = n_H
Return (intersection, t_c)
```

**Algorithm 5 Reflecting the normal**

Reflext_normal($p, n$)
$n' = n/\|n\|$ // unit normal
Return $p - 2 < p, n' > n'$

**Algorithm 6 Update trajectory with distance $t$**

```
Update_positions_momenta(q, p, t)
for i from 1 to n do
  [ω, C, Φ] = Compute_traj_params (q_i,p_i)
  Set q_i = C cos(ωt + φ)
  Set p_i = −ωC sin(ωt + φ)
Return (q, p)
```

# 7 Supporting information: results

To complement the analysis of section 4.4.1, we provide in the following plots with the variance of the statistics of interest (Figs. 8, 9, 10).

**Figure 8 Error analysis: variance as a function of the dimension. Model: isotropic simplex. (Left)** HAR **(Right)** HMC For the same window size, the variance of the error is lower for HMC.

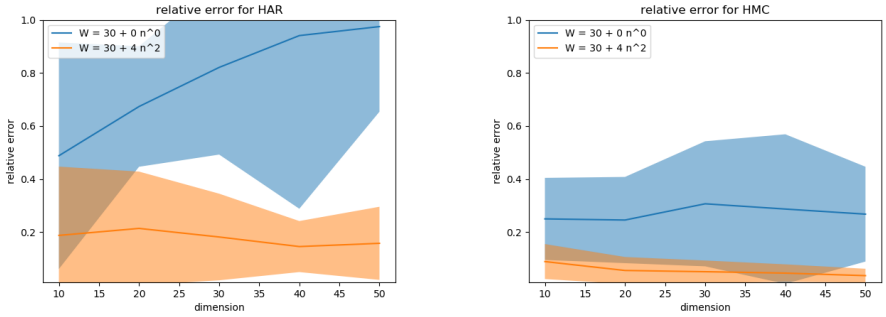



**Figure 9 Number of generated points: variance. Model: isotropic simplex. (Left) HAR (Right) HMC** The log scale hints at a polynomial number of points as a function of the dimension.

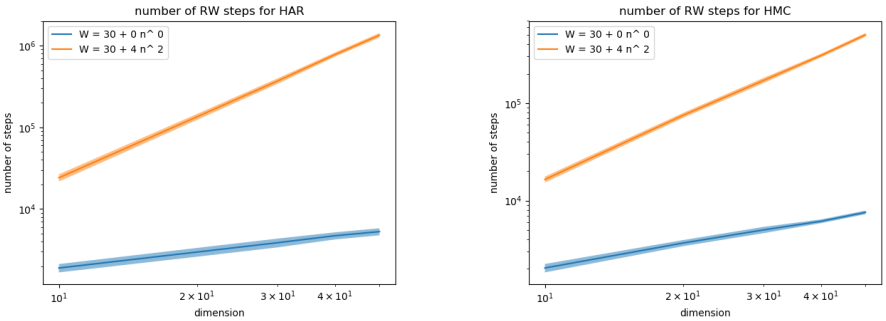
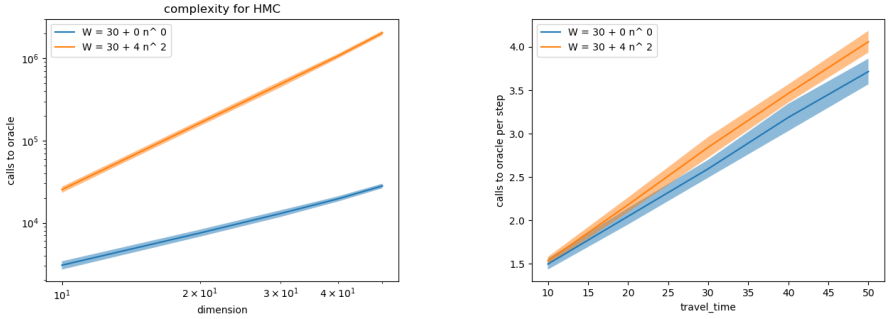
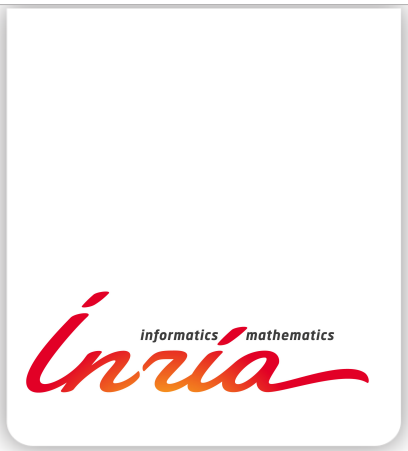


**Figure 10 (Left) Number of oracle calls for HMC (Right) Ratio between the number of oracle calls and the number of points generated** Plots for the isotropic simplex.






**RESEARCH CENTRE**
**SOPHIA ANTIPOLIS – MÉDITERRANÉE**

2004 route des Lucioles - BP 93
06902 Sophia Antipolis Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399