Random walks for estimating densities of states and the volume of convex bodies in high dimensional spaces

Augustin Chevallier

UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

THÈSE DE DOCTORAT

Marches aléatoires pour l'estimation de densités d'états et de volume de convexes en grande dimension

Augustin Chevallier

ABS, Inria Sophia Antipolis Méditerranée

Présentée en vue de l'obtention du grade de docteur en Informatique d'Université Côte d'Azur
Dirigée par: Frédéric Cazals
Soutenue le: 08/04/2019

Devant le jury, composé de:
T. Lelièvre, Professeur, Ecole des Ponts ParisTech
D. Wales, Professor, University of Cambridge
J-D. Boissonnat, Directeur de Recherches, INRIA
A. Trouvé, Professeur, CMLA
C. H. Robert, Directeur de Recherches, CNRS
Frédéric Cazals, Directeur de Recherches, INRIA

Marches aléatoires pour l'estimation de densités d'états et de volume de convexes en grande dimension

Jury:

Directeur

F. Cazals	Directeur de Recherches	Inria

Rapporteurs

T. Lelièvre	Professeur	Ecole des Ponts ParisTech
D. Wales	Professor	University of Cambridge

Examinateurs

J-D. Boissonnat	Directeur de Recherches	INRIA
A. Trouvé	Professeur	CMLA
C. H. Robert	Directeur de Recherches	CNRS

Invités

Random walks for estimating densities of states and the volume of convex bodies in high dimensional spaces

This manuscript introduces new random walks for the computation of densities of states, a central problem in statistical physics, and the computation of the volume of polytopes.

First, we focus on the Wang-Landau (WL) algorithm, a recently developed stochastic algorithm computing the density of states of a physical system. We propose an efficient random walk that uses geometrical information to circumvent the following inherent difficulties: avoiding overstepping strata, toning down concentration phenomena in high-dimensional spaces, and accommodating multidimensional distribution. Experiments on various models stress the importance of these improvements to make WL effective in challenging cases. These improvements make it possible to compute density of states for regions of the phase space of small biomolecules.

Second, we study Hamiltonian Monte Carlo (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 provide a convergence bound, paving the way to more precise mixing time analysis, and a robust implementation based on multi-precision arithmetic – a mandatory ingredient. We compare HMC with HAR within the polytope volume algorithm by Cousins and Vempala. The tests, conducted up to dimension 50, show that the HMC random walk outperforms HAR.

Finally, using Wang-Landau requires specifying the system handled, providing a suitable random walk to explore the definition domain, and possibly accommodate different variants of Wang-Landau. We present the first generic (C++) implementation providing all such ingredients.

Keywords: Monte-Carlo; statistical physics; importance sampling; MMC; Hamiltonian Monte Carlo; convex volume

Marches aléatoires pour l'estimation de densités d'états et de volume de convexes en grande dimension

Ce manuscrit présente de nouvelles marches aléatoires pour le calcul des densités d'états, un problème central en physique statistique, et le calcul de volume de polytopes.

Tout d'abord, nous étudions Wang-Landau (WL), un algorithme stochastique récemment développé pour calculer la densité d'états d'un système physique. Nous proposons une marche aléatoire efficace qui utilise l'information géométrique pour contourner les difficultés suivantes: éviter de sauter des strates, atténuer les phénomènes de concentration en grandes dimensions, et gérer les distribution multimodales. Les expériences montrent que ces améliorations sont critiques dans les cas difficiles, et permettent de calculer la densité des états pour des régions de l'espace des phases de petites biomolécules.

Puis nous étudions Hamiltonian Monte Carlo (HMC) avec réflexions sur les limites d'un domaine, offrant une alternative à Hit-and-run (HAR) pour échantillonner une distribution dans un domaine borné. Nous fournissons une borne de convergence, ouvrant la voie à une analyse plus précise du mixing time, et une implémentation robuste basée sur l'arithmétique multiprécision – un ingrédient obligatoire. Nous comparons HMC à Hit-and-run au sein de l'algorithme de volume polytope de Cousins et Vempala. Les essais, effectués jusqu'à la dimension 50, montrent que HMC surpasse HAR.

Enfin, l'utilisation de WL nécessitant de spécifier le système étudié, de fournir une marche aléatoire, et éventuellement d'incorporer des variantes de WL, nous présentons la première implémentation générique (C++) fournissant tous ces ingrédients.

Mots-clés: Monte-Carlo; physique statistique; importance sampling; MMC; Hamiltonian Monte Carlo; volume de convexe

Acknowledgments

Je tiens en premier lieu à remercier Frédéric Cazals, tout d'abord pour m'avoir donné un sujet vaste et intéressant, puis pour m'avoir accordé sa confiance et laissé une grande liberté dans l'exploration de celui-ci. Merci Frédéric pour tes conseils avisés sur la reproductibilité des expériences, et aussi pour m'avoir aidé à développer mes compétences en informatique.

Je remercie Tony Lelievre et David Wales d'avoir accepté de faire un rapport sur mes travaux.

Je voudrais aussi remercier l'équipe du CERMICS pour les enrichissantes discussions et particulièrement Tony Lelievre, Gabriel Stoltz et Benjamin Jourdain.

Un grand merci aussi aux organisateurs du programme "Complex High-Dimensional Energy Landscapes" à l'IPAM, ainsi qu'aux nombreux participants.

Je tiens à remercier Stefan Engblom pour m'avoir initié à la recherche et conforter mon choix d'associer mathématiques et calcul scientifique.

Merci aussi à Florence pour m'avoir si souvent aidé à résoudre mes tracasseries administratives.

Je remercie mes enseignants de classe préparatoire, en particulier Mr Génaux, Mr Suffrin et Mr Paviet, pour m'avoir donné goût aux mathématiques et à la physique.

Merci à Romain et Simon pour avoir égayé la vie au bureau avec des discussions intéressantes et plaisantes. Comment aurais-je pu découvrir les perles de youtube sans vous? Mais aussi merci pour m'avoir si souvent aider avec Python, R et C++.

Zoé, Boris, Christophe, Sarah et Romain merci pour tous ces moments de détente. J'attends d'ailleurs la prochaine raclette avec impatience ! David ne le sait pas encore, mais elle est prévue chez lui. Un merci spécial à Zoé qui a répondu avec patience et clarté à mes nombreuses questions de biologie et m'a ainsi orienté/poussé vers des mathématiques liées à la biologie.

Merci à David pour avoir été là/présent, pour les nombreuses discussions et sorties à Nice. Toujours prêt à échanger des idées de maths, mais aussi simplement à prendre une bière !

Enfin, un très grand merci à ma famille et à Bathilde, toujours présents au quotidien, aussi bien dans la détente que dans le travail. Merci pour le soutien permanent que vous m'avez apporté, ainsi que pour les nombreuses relectures et conseils.

Contents

List of Figures

List of Tables

Preface

This PhD started with the observation that the computational aspects of free energy computations for biomolecules and convex volume algorithms share many similarities. The overall goal was to bring improvements to both of these fields using hindsights coming from both fields.

I spent considerable time learning bits of statistical mechanics, which would pay out later due to its strong links with the Hamiltonian Monte Carlo (HMC) algorithm. In addition, we were confronted with a large amount of different computational approaches developed during the last 30 years for statistical physics: metadynamics, Wang-Landau, and others. In this context, we decided to focus on the Wang-Landau algorithm for two reasons. First it is a widely used algorithm, with many successful applications. Second, detailed mathematical analysis and explanations were available thanks to [LSR10]. In addition, many different aspects of the algorithm received attention. The learning rule was extensively studied, comparing the flat histogram criterion with deterministic rules and other proposed rules [SA11, WSTP15]. It was proposed to smooth or dynamically split the histogram [SNY11, BJMD13] or even replace it with a continous representation [FLE19]. Others suggested merging the Wang-Landau algorithm with parallel tempering [RKIdP03, JH06]. Finally, a wealth of physical systems were studied, from discrete models [LC10, SA11] to complex molecules such as met-enkephalin [JH06, RKIdP03, OMG10, SNY11, PCA$^+$06].

The physical systems studied triggered the development of various move sets. For example, in molecular studies, various move sets were proposed, based on molecular dynamics [RKIdP03, SNY11], on internal coordinates (dihedral angles) [OMG10, PCA$^+$06], or variants [MGCM12, BJMD13].

For convex volume computation, the situation is simpler: the community iteratively improved both the algorithm and the analysis of the theoretical complexity, starting with an $O^\star(n^{23})$ complexity [DFK91] and ending with an $O^\star(n^3)$ algorithm at the start of the PhD – see [CV18] and references therein.

We observed that while the random walks used for convex algorithm were general, every practitioner seemed to be using a custom random walk for Wang-Landau. Furthermore, the random walk used for Wang-Landau didn't seem to seek efficiency with respect to the Wang-Landau algorithm, often simply sampling the Boltzmann distribution. With that in mind, I proposed a new random walk specifically tailored for Wang-Landau making use of the topography of the energy landscape by using the gradient of the energy and alleviating some measure concentration phenomenons occuring in high dimensional spaces.

Naturally, we tried to bring these improvements to the computation of the volume of convex bodies, and specifically, polytopes. However these efforts were fruitless. One of the main problems of the random walk used for convex computation is that they tend to get stuck in corners. Using principles borrowed from the random walks introduced for Wang-Landau, it was possible to create a random walk leaving corners easily. On the other hand, it proved difficult to allow this random walk to *reach* corners easily. Hence, to preserve detail balance moves, leaving corners will be overwhelmingly rejected, defeating the purpose of the random walk!

The situation was unlocked by a paper on billiard walk [GP14], presenting a new random walk that does not get stuck in corners sampling the uniform distribution of a bounded domain. The billiard walk can be interpreted as an Hamiltonian Monte Carlo walk with reflections on the boundary and a constant potential energy, therefore the extension to non constant densities could be made by changing the potential energy. It turned out that this idea was already studied in a different context [AD15], but without a complete proof of convergence and without the application to convex volume computation. This random walk was then successfully used for convex volume computation, reducing the practical complexity of the algorithm. A

proof of convergence with convergence speed in the case of convex is given, laying the groundwork for more detailed complexity analysis.

These two random walks, for Wang-Landau and convex volume computation, required a lot of software development. For HMC with boundaries, rounding problems caused the random walk to leave the convex when using double precision floating point numbers, and we had to introduce the use of arbitrary precision numbers using IRRAM with the help of Sylvain Pion from INRIA Bordeaux. For Wang-Landau, as we have seen before a lot of different variants exist, and no general purpose Wang-Landau software was available. Hence I had to develop a general purpose implementation flexible enough to accommodate many different possible variations, while being robust and fast. Both of these projects required a significant amount of coding time and C++ coding skills.

The manuscript is organised as follows: first a preliminary chapter that introduces the necessary concepts of statistical physics, extracts the core ingredients of both Wang-Landau and the convex volume computation algorithms, and finally reviews the sampling algorithms. Then we present the three main contributions: the random walk for Wang-Landau, Hamiltonian Monte Carlo with boundaries for convex volume computation (with the software aspects), and finally the software introduced for Wang-Landau.

Chapter 1

Preliminaries

1.1 Statistical physics

This section aims to present a narrow part of statistical physics relevant to the computational problems at hand. First Hamiltonian dynamics will be presented, as it is the fondation of statistical mechanics and at the heart of the Hamiltonian Monte Carlo sampling algorithm that will be seen later on.

Then we present (with no direct use for this manuscript, but of great interest) the NVE ensemble and how the Birkhoff theorem can (sometimes) be used to justify statistical mechanics and the canonical measure.

After that, we present the NVT ensemble, which will be the setting used for the rest of the manuscript when dealing with statistical mechanics. In this setting, we define the partition function. Furthermore, we present the different definitions of the Free energy, even though we will only use one of them. While it was not strictly required for the scope of this manuscript to give these different definitions, I believe that for an unaware reader reading the literature, considerable confusion stems from them.

1.1.1 General results on Hamiltonian dynamic

Hamiltonian dynamics properties play a central role in statistical physics. Furthermore, it is at the core of the Hamiltonian Monte Carlo sampling strategy that will be discussed later. Therefore we give a short introduction to Hamiltonian dynamics and present key results.

Definition of the Hamiltonian flow

We consider a physical system with n degrees of freedom and an Hamiltonian H on $\mathbb{R}^{2n}$:

$$H(q,p) = U(q) + K(p)$$

where $K(p) = \frac{1}{2}\|p\|^2$ is the kinetic energy and the potential energy $U : \mathbb{R}^n \to \mathbb{R}$ is a smooth enough function with bounded gradient.

Consider the system of differential equations

$$\begin{cases} \dfrac{dq_i}{dt} = \dfrac{\partial H}{\partial p_i}, \forall i = 1, \dots, n \\ \dfrac{dp_i}{dt} = -\dfrac{\partial H}{\partial q_i}, \forall i = 1, \dots, n \end{cases} \tag{1.1}$$

Cauchy-Lipischitz theorem implies that for each $z = (q,p) \in \mathbb{R}^{2n}$ there is a unique maximal solution $\varphi_z(t)$ such that $\varphi_z(0) = z$. Since the gradient of U is bounded $\varphi_z(t)$ is defined for all $t \in \mathbb{R}$. We set the flow

$$\Phi_t(z) = \varphi_z(t).$$

Since the system 1.1 does not depend on the time t, the uniqueness of solutions implies that Φ is a flow acting on $\mathbb{R}^{2n}$ which means that

$$\Phi_0 = Id \text{ and } \Phi_t \circ \Phi_s = \Phi_{s+t}$$

for all t and s in $\mathbb{R}$. Moreover, for each t, $z \to \varphi_z(t)$ is differentiable and therefore Φ_t is a diffeomorphism (from the above equation, $\Phi_t^{-1} = \Phi_{-t}$).

Notation. For a vector X in $\mathbb{R}^{2n}$, X^q denotes the projection of X defined by the first n components and X^p the projection defined by the last n components. So that Φ_t^q is the projection of the Hamiltonian flow in the position space and Φ_t^p its projection in the momentum space.

Properties of the Hamiltonian flow

The simple calculation

$$\frac{d}{dt}H(\varphi_z(t)) =< \nabla H(\varphi_z(t)), \varphi_z'(t) >= 0$$

where $<,>$ is the dot product, implies the important fact:

Theorem. 1.1. *The flow Φ_t preserves the Hamiltonian H.*

Schwarz's theorem implies that the divergence of the vector field $(\frac{\partial H}{\partial p}, -\frac{\partial H}{\partial q})$ vanishes. It implies that the Hamiltonian flow preserves the Lebesgue measure of $\mathbb{R}^{2n}$.

Theorem. 1.2 (Liouville's theorem)**.** *The lebesgue measure λ is invariant by the flow Φ_t, i.e. for all t and all mesurable subset A,*

$$\lambda(\Phi_t^{-1}(A)) = \lambda(A).$$

The following crucial theorem is deduced from Liouville's theorem:

Theorem. 1.3. *Every measure with a density μ that is a function of H, ie $\exists f$ such that $\mu(x) = f(H(x))$, is invariant by Φ_t.*

Proof. For every measurable subset A in $\mathbb{R}^{2n}$,

$$\begin{aligned}
\mu(\Phi_t^{-1}(A)) &= \int_{\Phi_t^{-1}(A)} f(H(x))d\lambda(x) \\
&= \int 1_A \circ \Phi_t(x)\, f(H(x))d\lambda(x) \\
&= \int 1_A \circ \Phi_t(x)\, f(H(\Phi_t(x)))d\lambda(x) \\
&= \int 1_A(x)\, f(H(x))d\lambda(x) \\
&= \mu(A).
\end{aligned}$$

□

Remark 1.1. *This result can also be derived from the Liouville equation by noticing that the Poisson bracket $\{H, \mu\}$ is 0 for any smooth density satisfying the hypothesis $\mu(x) = f(H(x))$.*

1.1.2 Statistical ensembles

We consider a system of N particles, an Hamiltonian H and a measure μ on the phase space $\Gamma = \mathbb{R}^{3N}$.

Observable

In the context of statistical ensembles, a macroscopic observable $\bar{f}$ is the average over the phase space:

$$\bar{f} = \int_\Gamma f(q,p) d\mu(q,p)$$

with μ a measure invariant by the Hamiltonian flow that depends on the statistical ensemble considered.

NVE - Microcannonical

For the microcannonical ensemble, we consider the energy (potential + kinetic) E fixed. Since the Hamiltonian H is Φ_t-invariant, the level set $\Sigma_E = H^{-1}(\{E\})$ is stable, i.e. $\Phi_t(\Sigma_E) \subset \Sigma_E$.

For a given energy E, we can build a measure μ on Σ_E that is invariant by the flow of the Hamiltonian. The measure μ has the following density REF:

$$\mu(x) = \frac{d\sigma_E}{\|\nabla H(x)\|} \tag{1.2}$$

where $d\sigma_E$ is the surface element induced on the manifold Σ_E by the Lebesgue measure of Γ.

In this specific case, it is possible to give some justifications to the definition of an observable. The assumption is that at a macroscopic level, the observables are *time averages*: any measurement we make is not instantaneous. As the movements at a molecular level are extremely fast, the average is taken for a long time with respect to the system, even if it is a short time for us.

Using results from ergodic theory, it is possible to justify that –in some cases – the time average is equal to the spacial average:

Theorem. 1.4 (Birkhoff)**.** *Let $(X, \mathcal{A}, \mu)$ be a σ-finite measure space and $\tau : (x,t) \in X \times \mathbb{R} \to \tau(t,x) = \tau_t(x) \in X$ be a measurable flow that preserves μ. Then for all functions $g \in \mathcal{L}^1(\mu)$ and for all positive T, the integral $\int_{[0,T]} g(\tau_t(x))dt$ is well defined for almost all x and*

$$\frac{1}{T}\int_{[0,T]} g(\tau_t(x))dt \underset{T\to+\infty}{\longrightarrow} \bar{g}(x)$$

for almost all x where $\bar{g}$ is an invariant measurable function $\bar{g}$, i.e., a measurable function such that for all t, $\bar{g} \circ \tau_t = \bar{g}$ almost everywhere. Moreover, if the flow is ergodic and if $\mu(X)$ is finite, then $\bar{g}$ is almost everywhere equal to the constant $\frac{1}{\mu(X)}\int_X g(x)dx$.

Definition. 1.1 (Ergodic flow)**.** *A subset A is said to be invariant by a flow ϕ_t if for every t, $\phi_t^{-1}(A) = A$. A flow ϕ_t is said to be ergodic if every measurable invariant subsets by the flow is either of measure 0 or its complement is of measure 0.*

As the measure μ is invariant by the Hamiltonian flow, if the Hamiltonian flow is ergodic, we have for any observable

$$\frac{1}{T}\int_{[0,T]} f(\Phi_t(x))dt \underset{T\to+\infty}{\longrightarrow} \bar{f} \tag{1.3}$$

meaning that the observable is the average over *almost all* trajectories of the system. Of course the flow is not always ergodic, and no general results can be stated. Even trivial examples yield non ergodic flows. For example, a double well potential where the energy barrier between the two wells prevents any trajectory with energy lower than the barrier from leaving its starting well [LSR10].

Remark 1.2. *If the Hamiltonian flow is ergodic, simply following the Hamiltonian will provide some sort of sampling (albeit not random) of the iso-energy surface Σ_E sufficient to compute the integral of any function in $\mathcal{L}^1(\mu)$.*

NVT - Cannonical

The NVT ensemble now considers our system in contact with a thermostat, that is another system so large that the studied system's influence on the thermostat is negligible. In particular the temperature T of the thermostat is constant.

In this setting, the canonical distribution is the Boltzmann measure with density:

$$\mu(q,p) = e^{-\frac{H(q,p)}{k_B T}} \tag{1.4}$$

where k_B is the Boltzmann constant. As per Th. 1.3, this measure is invariant by the Hamiltonian flow. It can be derived as the maximum of the entropy, however the derivation is outside the scope of this thesis. Much more could be said about the NVT ensemble, but this manuscript focusing on computational aspects only requires the definition of the Boltzmann distribution. For further reading, I heartily recommend [LSR10] and [Ada06].

1.1.3 Partition function and Free energy

In the previous section, the measures μ (for NVT and NVE) were not renormalised. We call *partition function* the renormalisation factor

$$Z = \int_\Gamma e^{-\frac{H(q,p)}{k_B T}} dqdp. \tag{1.5}$$

The absolute free energy is

$$F = -k_B T \ln(Z) \tag{1.6}$$

For hamiltonians of the form $H(q,p) = U(q) + K(p)$ (i.e. separable), the contributions of the momentums can be analysed separately:

$$Z = \int_\Gamma e^{-\frac{H(q,p)}{k_B T}} dqdp = \int_{\Gamma_q} e^{-\frac{U(q)}{k_B T}} dq \int_{\Gamma_p} e^{-\frac{K(p)}{k_B T}} dp \tag{1.7}$$

where $\Gamma = \Gamma_q \times \Gamma_p$. For classical systems, the kinetic energy $K(p)$ can be written as $K(p) = \frac{1}{2} p^T M p$ where M is the mass matrix

$$M = \begin{pmatrix} m_1 I_3 & & 0 \\ & \ddots & \\ 0 & & m_N I_3 \end{pmatrix}.$$

In this setting, the momentum contribution to the partition function has the following analytical expression [LSR10]

$$\int_{\Gamma_p} e^{-\frac{K(p)}{k_B T}} dp = (2\pi k_B T)^{\frac{3N}{2}} \prod_{i=1}^{N} m_i^{3/2} \tag{1.8}$$

Therefore the important part of the partition function $\int_{\Gamma_q} e^{-\frac{U(q)}{k_B T}} dq$ depends only on the potential energy U. We might abuse notations and write $Z = \int_{\Gamma_q} e^{-\frac{U(q)}{k_B T}} dq$ as the momentum contribution is simply a known multiplicative constant.

Partial free energy. We consider a subset A of the phase space Γ. We define the partial free energy for A as

$$Z_A = \int_A e^{-\frac{H(q,p)}{k_B T}} dqdp \tag{1.9}$$

The quantity $P_A = \frac{Z_A}{Z}$ represents the probability of being in subset A. That means that if one observes the system for a very long time, the proportion of the total time spent in A is P_A. Notice that considering another subset B, the quantity $\frac{P_A}{P_B}$ can be computed without computing the full partition function Z since

$\frac{P_A}{P_B} = \frac{Z_A}{Z}\frac{Z}{Z_B} = \frac{Z_A}{Z_B}$, which gives us the proportion of time spent in A with respect to time spent in B. Going further, using results from transition state theory, it is possible to compute the instantaneous probability of switching from A to B if they share a border by integrating over their boundaries. However, this is beyond the scope of this manuscript.

Consider g the pushfoward measure of the Lebesgue measure on A by the Hamiltonian H i.e. for any $e_0 < e_1$,

$$g([e_0, e_1]) = \int_A 1_{[e_0,e_1]}(H(q,p))dqdp. \tag{1.10}$$

Under reasonable regularity assumptions on H, g has a density $g(x)$, and therefore the partition function at temperature T can be written as

$$Z_A = \int_{\mathbb{R}} g(x)e^{-\frac{x}{k_B T}}dx$$

Therefore, if one knows g, the partition function Z_A can be computed at any temperature T. It allows to deduce thermodynamical quantities that can be computed as derivatives of the partition function such as the heat capacity [Cai11]. The quantity g is often denoted as the *Density of State* (DoS) in the literature.

Remark 1.3. *Using the decomposition of the partition function of eq. 1.7, the density of state can be defined relatively to the potential energy function U (on Γ_q) instead of the full Hamiltonian H (on Γ).*

Remark 1.4. *If the set A is not bounded, the density of states g can be infinite.*

Reaction coordinate and free energy. Another definition of the free energy relying on reaction coordinates is commonly used. An in depth discussion of the definition can be found in [LSR10], including the computation presented here. We will not use it in our work although it is widely used by practitioners. This definition justifies the use of often found term Free Energy Landscape.

A *reaction coordinate* is a function $\xi : \Gamma_q \to \mathbb{R}^m$ with $m < 3N$. The marginal distribution μ^ξ on $\mathbb{R}^m$ is defined by

$$\int_A d\mu^\xi(z) = \mu(\xi^{-1}(A)) \tag{1.11}$$

for every measurable subset A in $\mathbb{R}^m$, it is the pushforward of the Boltzmann distribution. When ξ is smooth with gradient which doesn't vanish, this measure has a density [LSR10]:

$$\mu^\xi(z) = \frac{1}{Z}\int_{\Sigma_z \times \mathbb{R}^{3N}} e^{-\frac{H(q,p)}{k_B T}} \frac{\sigma_{\Sigma_z}(dq)}{\|\nabla\xi(q)\|}dp. \tag{1.12}$$

where $\Sigma_z = \xi^{-1}(z)$ and $\sigma_{\Sigma_z}(dq)$ is the surface element induced on the manifold Σ_z by the Lebesgue measure of Γ_q. The free energy with respect to the reaction coordinate ξ is then defined as the function

$$F(z) = -k_B T \ln\left(\mu^\xi(z)\right). \tag{1.13}$$

Note that in statistical physics text books the absolute free energy is a number while the free energy with respect to a reaction coordinate is a function. This is why practitioners talk about the Free Energy Landscape by analogy with the Potential Energy Landscape [**?**, Wal03, LSR10].

Remark 1.5. *For $\xi(x) = U(x)$, the measure μ^ξ is the density of state with respect to the potential energy.*

1.2 Randomized algorithms using sampling

In this section, q and p will denote probabilities and not the position and momentum.

1.2.1 Monte Carlo

Let f be a function of $\mathbb{R}^n$ to $\mathbb{R}$ and p a probability density on $\mathbb{R}^n$. We consider the problem of numerically evaluating

$$\mathbb{E}_p[f] = \int_{\mathbb{R}^n} f(x)p(x)dx. \tag{1.14}$$

Using i.i.d. $X_i \sim p$, Monte-Carlo integration gives the following estimation of $\mathbb{E}_p[f]$:

$$Z_N = \frac{1}{N}\sum_{i=1}^{N} f(X_i). \tag{1.15}$$

Let $\sigma^2 = \mathbb{E}_p[f^2] - \mathbb{E}_p[f]^2$. Z_N is a random variable and one has:

$$\begin{cases} \mathbb{E}[Z_N] = \mathbb{E}_p[f] \\ \text{Var}\,[Z_N] = \sigma^2/N \end{cases} \tag{1.16}$$

The quality of a Monte Carlo estimation is determined by its variance. Therefore, the number of required samples N for a good estimation increases with σ^2.

Deterministic integral evaluation methods are based on the properties of f and their costs are strongly dependent on the dimension. However the Monte-Carlo integration can be reformulated in a dimension independent way. Let π the pushforward measure on $\mathbb{R}$ of p by f. We recall the definition of the pushforward measure, for any measurable subset A:

$$\pi(A) = \int_{\mathbb{R}^n} 1_A(f(x))p(x)dx \tag{1.17}$$

Let $Y_i = f(X_i)$. Then $Y_i \sim \pi$ and are i.i.d. . Therefore we can rewrite the approximation as

$$Z_N = \frac{1}{N}\sum_{i=1}^{N} Y_i$$

which is the Monte-Carlo methods applied to $\int_{\mathbb{R}} \pi(x)dx$ with $p = \pi$, $n = 1$ and $f : x \to x$.

Hence Monte-Carlo integration convergence speed depends only on the pushfowrad measure π, making it suited for high dimensional problems. This is well illustrated by the following example: let B be a measurable subset of $A = [0,1]^n$ with volume V, let $f = 1_B$ and let

$$I = \int_{[0,1]^n} f(x)dx$$

Here, $p = \lambda$ the Lebesgue measure on $[0,1]^n$, therefore the variables X_i are independent and uniformly distributed in $[0,1]^n$. Therefore the variables $Y_i = f(X_i)$ are independent Bernoulli random variables of parameter V. Thus the Monte Carlo method is exactly the same *for any* n and depends only on V.

1.2.2 Limitation of Monte-Carlo integration and measure concentration

We have previously seen that Monte-Carlo integrations seem to be independent from the dimension. However that is not completely true: the measure π (keeping the same notations) usually depends on the dimension. Consider the following example:

$$I_n = \int_{[0,1]^n} 1_{[0.1,0.9]^n}(x)1.25^n dx \tag{1.18}$$

It is straightforward to see that $I_n = 1$ for all $n \geq 1$. Let us have a look at the variance of the Monte-Carlo estimator:

$$\begin{aligned}\sigma^2 &= \int_{[0,1]^n} 1_{[0.1,0.9]^n}(x) 1.25^{2n} dx - \left(\int_{[0,1]^n} 1_{[0.1,0.9]^n}(x) 1.25^n dx \right)^2 \\ &= 1.25^n - 1\end{aligned}$$

which explodes with the dimension. In practice, Monte-Carlo integration is affected by the phenomenon known as concentration of measure in high dimension.

Concentration of measure – volume. Intuitively, concentration of measure in high dimension implies that most of the volume of a given set is concentrated near its boundary. A striking example is the ball of radius 1 and center 0 in $\mathbb{R}^n$: B_n. Then consider the balls B_n^ϵ of radius $1 - \epsilon$. Then for all ϵ

$$\frac{V(B_n^\epsilon)}{V(B_n)} = (1-\epsilon)^n \xrightarrow[n\to\infty]{} 0$$

Meaning that all the volume gets concentrated on a very narrow band near the boundary of the ball when the dimension increases, and it happens exponentially fast. This trivial result has direct implication for Monte-Carlo integration: any function whose supporting domain (i.e the subset where it is non zero) is defined by an equation of the type $\|x\| \leq r$ will see the volume of the supporting domain shrink to 0 exponentially fast if the total integration domain is for example the unit ball and $r < 1$.

Statistical Physics. In statistical physics, the computation of the partition function $Z_A = \int_A e^{-U(q)/k_B T} dq$ of a subset A is a prime example of this problem. Let's study a central example: the harmonic potential $U(q) = \|q\|^2$. Assume for the sake of simplicity that A is the unit ball (therefore, all energies are in the range $[0, 1]$). For the contribution of each energy level to the integral, two phenomenons compete. As we have seen before, for n large, most of the mass is concentrated near the unit sphere. However, the function $e^{-U(q)/k_B T}$ decreases exponentially with $\|q\|$. This implies an equilibrium where most of the contributions to the integral come from energies in an interval $[u_0, u_1]$ with $0 < u_0 < u_1 < 1$ (depending on the temperature T). Since most of the volume is concentrated near the unit sphere, the volume of $U^{-1}([u_0, u_1])$ is extremely small compared to the volume of the unit sphere. Therefore, when performing a Monte-Carlo integration (assuming one samples the Lebesgue measure of the unit ball), most of the points will fall out of the contributing area, leading to high errors. In other words, the variance of the Monte-Carlo estimator of the partition function Z_A is very high, making Monte-Carlo estimation impractical.

1.2.3 Importance sampling

Importance sampling is a general technique which allows the variance of the Monte-Carlo estimator for an integral to be reduced. The goal is still to compute $\mathbb{E}_p[f]$. Let q be another probability on $\mathbb{R}^n$. We rewrite $\mathbb{E}_p[f]$ as

$$\begin{aligned}\mathbb{E}_p[f] &= \int_{\mathbb{R}^n} f(x)p(x)dx \\ &= \int_{\mathbb{R}^n} \frac{f(x)p(x)}{q(x)} q(x)dx \\ &= \mathbb{E}_q[fp/q]\end{aligned}$$

Using i.i.d. RV $Y_i \sim q$, we modify the estimator to

$$Z'_N = \frac{1}{N} \sum_{i=1}^{N} f(Y_i) \frac{p(Y_i)}{q(Y_i)}. \tag{1.19}$$

Denoting $\sigma_q^2 = \mathbb{E}_p[f^2 p/q] - \mathbb{E}_p[f]^2$, one has:

$$\begin{cases} \mathbb{E}[Z'_N] = \mathbb{E}_p[f] \\ Var(Z'_N) = \sigma_q^2/N \end{cases} \tag{1.20}$$

Remark 1.6. *If $q(x) = f(x)p(x)/\mathbb{E}_p[f]$ then $\sigma_q = 0$. In this case, the estimator has 0 variance and will give the correct result for any N. Unfortunately, this requires knowing the result beforehand.*

Choosing a probability q roughly proportional to fp reduces σ_q^2 and therefore the variance of the estimator Z'_N .

Renormalisation factor and computation of the partition function. We have seen before that computing the partition function of a system is a hard problem. A natural idea is to use samples from the Boltzmann distribution (which can be obtained by a molecular dynamic simulation) as an importance sampling scheme. Indeed, with the Boltzmann distribution μ, the function to integrate would be constant. However this fails because of the renormalisation factor of μ: for a given subset A, the density $\mu(q)$ is equal to $\frac{e^{-U(q)/k_B T}}{Z_A}$ and not $e^{-U(q)/k_B T}$. Hence:

$$Z_A = \int_A e^{-U(q)/k_B T} dq = \int_A Z_A \mu(q) dq \tag{1.21}$$

which requires knowing Z_A. If one wants to use sampling according to the Boltzmann distribution, the only way is to compute the inverse of Z_A (assuming the volume of A is known):

$$\frac{Vol(A)}{Z_A} = \int_A e^{U(q)/k_B T} \mu(q) dq \tag{1.22}$$

which is once again a difficult integral to estimate.

1.2.4 Multiphase Monte Carlo

Multiphase Monte Carlo (MMC) is a loosely defined technique that consists in splitting a Monte Carlo integration that is too complex on its own into multiple manageable steps called *phases*. As this term covers many different situations, I will simply give an example here, with intuition on how it reduces the error compared to standard Monte Carlo.

Let A be a compact subset of $\mathbb{R}^n$, and $f : A \longrightarrow \mathbb{R}$. The goal is to compute $I = \int_A f(x)dx$, and we assume that the integral I is too hard to compute via traditional Monte-Carlo integration. In this case, a Multiphase Monte Carlo technique could be to introducing a sequence of function $f_0, ..., f_m = f$ such that $\frac{f_i}{f_{i-1}} \in [1/2, 3/2]$ and $I_0 = \int_A f_0$ is easy to compute. Then the integral is rewritten as

$$I = I_0 \prod_{i=1}^{m} \frac{I_i}{I_{i-1}}$$

where $I_i = \int_A f_i(x)dx$. Each ratio $R_i = \frac{I_i}{I_{i-1}}$ can be computed using importance sampling:

$$R_i = \frac{I_i}{I_{i-1}} = \int_A \frac{f_i(x)}{f_{i-1}(x)} \frac{f_{i-1}(x)dx}{I_{i-1}}$$

by sampling from the probability $\frac{f_{i-1}(x)dx}{I_{i-1}}$ (using any sampling technique described in section 1.3), and the estimation of a single ratio is called a *phase*, for a total of m *phases*.

Since the ratio $\frac{f_i}{f_{i-1}}$ is in $[1/2, 3/2]$, the variance of the Monte-Carlo estimator of each ratio can be bounded by a constant divided by the number of samples.

Denoting the ratio $\frac{I_i}{I_{i-1}}$ as R_i and its Monte Carlo estimator as $\hat{R}_i$, we deduce that there exists $C > 0$ (in this specific case, $C = 4$) such that

$$\frac{Var(\hat{R}_i)}{R_i^2} \leq \frac{C}{N} \tag{1.23}$$

where N is the number of samples. Since the estimation of the successive ratios are independent, we deduce the following variance for the product of the ratios:

$$Var\left(\prod_{i=1}^{m} \hat{R}_i\right) = \prod_{i=1}^{m}(Var(\hat{R}_i) + R_i^2) - \prod_{i=1}^{m} R_i^2$$

Hence

$$\begin{aligned}\frac{Var(\prod_{i=1}^{m} \hat{R}_i)}{\prod_{i=1}^{m} R_i^2} &= \prod_{i=1}^{m}\left(\frac{Var(\hat{R}_i)}{R_i^2} + 1\right) - 1 \\ &\leq \left(\frac{C}{N} + 1\right)^m - 1\end{aligned}$$

Using the fact that $log(1 + x) \leq x$ and that for $0 < x < 1$, $e^x - 1 < \frac{7}{4}x$, we deduce that for $N > Cm$,

$$\frac{Var(\prod_{i=1}^{m} \hat{R}_i)}{\prod_{i=1}^{m} R_i^2} \leq \frac{m7C}{4N} \tag{1.24}$$

The variance of the product estimation grows linearly with the number of terms m. For a target relative variance ϵ, choosing N as

$$N = \frac{m7C}{4\epsilon} \tag{1.25}$$

will ensure a total relative variance less than ϵ. Therefore, the number of samples per ratio is proportional to m.

Starting from $f_0(x) = 1$, it is possible to build a sequence of functions f_i with bounded ratio and such that $Var(f_i)$ increases exponentially with i. Conversely, it shows that it is possible in some cases to compute extremely difficult integrals with a number of *phases* m depending on the log of the variance of the standard Monte-Carlo estimator! The final complexity is $O(m^2)$ since $O(m)$ samples are requires per phase.

To sum up: problems with exponential complexity using standard Monte-Carlo can be solved in polynomial time by a clever use of MMC.

Remark 1.7. *Here we assumed that $\frac{f_i}{f_{i-1}} \in [1/2, 3/4]$, however it is not required. First, observe that*

$$\frac{Var(\hat{R}_i)}{R_i^2} = \frac{Var(\frac{f_i}{f_{i-1}})}{NR_i^2}.$$

Now, assuming that $\frac{Var(\frac{f_i}{f_{i-1}})}{R_i^2} \leq C$, the same reasoning can be followed up to Eq. 1.25. Second, observe that with this new assumption, the ratio $\frac{f_i}{f_{i-1}}$ can be very large, as long as its relative variance is $O(1)$. In practice, this allows the number of phases m to be reduced and therefore the total complexity to be reduced.

Remark 1.8. *It is sometimes more convenient to estimate $R_i' = \frac{1}{R_i} = \int_A \frac{f_{i-1}(x)}{f_i(x)} \frac{f_i(x)dx}{I_i}$. In this setting, the variance analysis cannot be done since $\hat{R}_i' = 0$ can have a non zero probability and therefore $1/\hat{R}_i'$ has infinite variance. However, a similar analysis can be done in probability, removing the bad cases, and similar complexities $N = O(m)$ can be obtained in specific settings, see e.g. [KLS97].*

We can summarise the previous discussion with the following precise statement:

Theorem. 1.5. *Let $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ and let $f_0, ..., f_m = f$ be a sequence of functions from $\mathbb{R}^n$ to $\mathbb{R}$. Let denote*

$$I = \int f\,dx \text{ and } I_i = \int f_i\,dx$$

the integrals of the functions f and f_i, and

$$\sigma_i^2 = Var_{p_{i-1}}\left(\frac{f_i}{f_{i-1}}\right)$$

the variance with respect to the probability p_i with density $p_i(x) = \frac{f_i(x)}{I_i}$.

Let denote $\hat{R}_i$ the Monte-Carlo estimator of $R_i = \frac{I_i}{I_{i-1}}$ using a sample of size N from the probability p_{i-1}. Assuming I_0 is known, the estimator $\hat{I}$ is defined as:

$$\hat{I} = I_0 \prod_{i=1}^{m} \hat{R}_i.$$

Let $C = \max_{i=1...m}(\frac{\sigma_i^2}{R_i^2})$. If $N > Cm$ then

$$\frac{Var(\hat{I})}{I^2} \leq \frac{m7C}{4N}$$

Hence for a target relative variance $\epsilon < 1$ of $\hat{I}$, the number of samples N can be chosen as

$$N = \frac{m7C}{4\epsilon}$$

1.2.5 Density estimation - relative error

Consider a probability density function $g : [0,1] \longrightarrow \mathbb{R}^+$ (such that $\int_0^1 g(x)dx = 1$). The problem at hand is to estimate g from i.i.d. random variables $X_i \sim g$.

The simplest method is to divide $[0,1]$ into small intervals $0 = a_0 < a_1 < ... < a_k = 1$, and build a histogram estimator $\hat{g}$ from this discretisation. Using N samples:

$$\hat{g}(x) = \sum_{i=1}^{k} 1_{[a_i,a_{i+1}]}(x) \frac{1}{(a_{i+1} - a_i)N} \sum_{j=1}^{N} 1_{[a_i,a_{i+1}]}(X_j)$$

Other estimators such as the kernel density estimators are available. However, all these common estimators share one common trait: they provide good estimations in absolute error, but not in relative error. In other words, if g is very close to 0 in some regions, $\hat{g}$ will be very poorly estimated in a relative sense: the histogram estimator will simply estimate the density to 0, while a kernel density estimator will have "leftovers" from the kernels.

We can define a relative error as follows, assuming that $g(x) > 0$ for $x \in [0,1]$:

$$e_{relative} = \int_0^1 \frac{|\hat{g}(x) - g(x)|}{g(x)} dx$$

Importance sampling for density estimation. As importance sampling allows badly behaved integrals to be computed, it makes density estimations with small relative error possible. Let $q : [0,1] \longrightarrow \mathbb{R}^+$ a probability density on $[0,1]$. Then let

$$h(x) = C\frac{g(x)}{q(x)} \tag{1.26}$$

with C a renormalisation constant such that h is a probability density. Let $\hat{h}$ be an estimator of h. We build a new estimator $\hat{g}_{is}$ for g:

$$\hat{g}_{is}(x) = C\hat{h}(x)q(x) \tag{1.27}$$

As with importance sampling for integral computation, if $q = g$, then the estimator has 0 variance. Moreover, if q is close to g, for example if there exists α such that $\frac{g(x)}{q(x)} \in [1-\alpha, 1+\alpha]$, then we can control the relative error of the estimator g_{is}. In the special case of a histogram estimator, the number of points in each bin follows a multinomial distribution, and it is possible to give bounds for the variance of the estimation of each bin.

Application to statistical physics. A natural course of action is to estimate the Density of State g defined by Eq. (1.10). However, we face a problem: g is *a priori* not a probability measure, and worse, g is not a finite measure. First, we need to restrict ourselves to a bounded domain A of Γ_q. Then we can compute the renormalised Density of State $\frac{g(x)}{\int_{\mathbb{R}} g(y)dy}$.

Therefore, if the volume of A is known, we can compute the Density of State and the partition function since $\int_{\mathbb{R}} g(y)dy = Vol(A)$. If the volume is not known, we can only compute the density of state and the partition function up to a multiplicative function. However, the constant does not depend on the temperature, hence we can get thermodynamic quantities such as the heat capacity up to a multiplicative constant, which still allows to determine phase transitions for example. Another way of proceeding could be to compute the volume of a subset of A of the form $A \cap \{x | U(x) \in [u_0, u_1]\}$ which also allows a correct renormalisation to be computed.

Remark 1.9. *It would also be possible to sample the Boltzmann distribution in Γ_q instead of the Lebesgue measure and compute the (renormalised) pushforward measure of the Boltzmann distribution on Γ_q. This would remove the need to take a subset of Γ_q as the Boltzmann distribution has finite measure under reasonable assumptions on the potential energy U. However, it would be difficult to compute any renormalisation factor. A discussion about the practical implications of a change of measure for a histogram estimation is discussed in Section 4.2.3.*

1.2.6 Wang-Landau

The Wang-Landau algorithm was first introduced in the context of statistical physics to compute the Density of State, that is the pushforward measure g (to follow the notations from section 1.2.5) of the Lebesgue (or any measure π) measure by the potential energy function, with a low relative error. Essentially, it estimates θ^*, a discretised version of g (ie a histogram).

To fix notations, consider a probability distribution with density $\pi(x)$ defined on a subset $\mathcal{E} \subset \mathbb{R}^n$. Let $U : \mathcal{E} \longrightarrow \mathbb{R}$ be a potential energy function, and $U_{min} = U_0 < U_1 < ... < U_d = U_{max}$ be a discretisation of the energy range. We consider a partition of $\mathcal{E}$ into so-called strata

$$\{\mathcal{E}_1, \dots, \mathcal{E}_d\} \text{ with } \mathcal{E}_i = U^{-1}([U_{i-1}, U_i]). \tag{1.28}$$

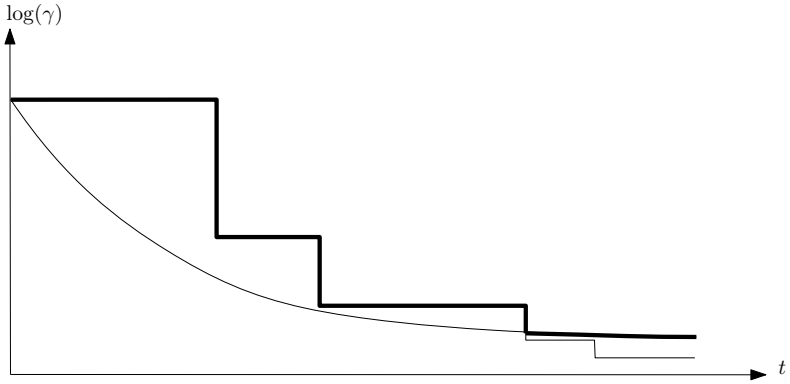
Let $J(x)$ be the index such that $x \in \mathcal{E}_{J(x)}$. The goal of WL is to estimate

$$\theta_i^* = \pi(\mathcal{E}_i) = \int_{\mathcal{E}_i} \pi(x)dx, \forall 1 \leq i \leq d. \tag{1.29}$$

To achieve this goal, the Wang-Landau algorithm relies on Multiphase Monte-Carlo and importance sampling to build a sequence $\theta(t)$ of estimations of θ^* that converges to θ^*. Following the idea of section 1.2.5, at step t, a point is sampled according to an analogous of distribution p in Eq (1.26):

$$\pi_{\theta(t)}(x) = \left(\sum_{i=1}^{d} \frac{\theta_i^*}{\theta_i(t)} \right)^{-1} \frac{\pi(x)}{\theta_{J(x)}(t)} \tag{1.30}$$

Figure 1.1 Wang-Landau: evolution of the log of the learning rate. The stairway curve corresponds to the halving rule–which yields a saturation of the error. The smooth $1/t$ curve yields convergence. Practically, the two strategies are combined to improve the convergence speed: one starts with the halving rule, switching to the $1/t$ rule when the two curves meet [BP07a]. Note the length of plateaus to move from t to $t+1$ depends on the random walk–whence the depicted variability.



Points are sampled using an MCMC method (see section 1.3): $x_{t+1} \sim P_\theta(x_t, .)$ where P_θ is a Markov kernel that leaves π_θ invariant. The construction of P_θ will be discussed in greater details in chapter 2.

Now, if WL was strictly following section 1.2.5, we would sample N points with respect to $\pi_{\theta(t)}$, compute a histogram estimation of $\pi_{\theta(t)}$, and finally deduce $\theta(t+1)$ using an analogous to eq 1.27. However, WL only samples one single point x_t from $\pi_{\theta(t)}$, and instead of building a histogram (which is impossible with a single point), the density is updated with

$$\theta_{J(x_t)}(t+1) = \theta_{J(x_t)}(t)\gamma \tag{1.31}$$

where $\gamma > 1$ is called the learning rate. To converge to θ^*, the convergence factor γ is made to converge to 1.

How to decrease the learning rate γ requires a small discussion. Historically [WL01], the rule used the Flat Histogram criterion. Let $\nu_t(i)$ be the number of samples up to iteration t falling into bin $\mathcal{E}_i$. The vector $\{\nu_t(i)\}$ is said to verify the **flat histogram** (FH) criterion provided that, given a constant c:

$$\max_{i=1,\ldots,d} \mid \frac{\nu_t(i)}{t} - \frac{1}{d} \mid < c. \tag{1.32}$$

If the criterion is verified, γ is decreased using $\gamma = \sqrt{\gamma}$. The hope associated with this rule was an exponential decay of the relative error. However, what is described in section 1.2.4 allows exponentially difficult problems to be tackled, but the relative error only decreases in $\frac{1}{\sqrt{N}}$ with N the number of steps in the end. In practice, for the WL algorithm it means that the error first decreases exponentially fast in the sense that it goes from an exponentially difficult problem to a manageable one. However, hoping for exponential convergence for this manageable problem is unrealistic, and it turns out that this strategy leads to a saturation of the error [BP07a, BP07b]. As a side note, the flat histogram variant seems to be sensitive to the particular analytical form of the update rule [JR14]. To circumvent the error saturation, the rate $\gamma_t = exp(1/t)$ rule was proposed [BP07a]. Practically, one combines the two update strategies by starting with the flat histogram strategy and switching to the latter strategy as soon as the proposed γ is smaller than $exp(1/t)$ – [BP07a] and Fig. 1.1.

Algorithm 1 Wang Landau

Set $\theta = (1/d, ..., 1/d)$
Set exponential regime = True
Set $\gamma = \gamma_0$ with $\gamma_0 > 1$
while $t < t_{max}$ **do**
  Sample $x_{t+1} \sim P_\theta(x_t, .)$
  Set $\theta_{J(x_{t+1})} = \gamma\ \theta_{J(x_{t+1})}$
  Renormalise θ
  if Exponential regime **then**
    if Flat histogram **then**
      $\gamma = \sqrt{\gamma}$
    if $\gamma < \exp(\frac{1}{t+1})$ **then**
      Set exponential regime = False
      Set $\gamma = \exp(\frac{1}{t+1})$
  else
    $\gamma = \exp(\frac{1}{t+1})$

Theoretical convergence

The theoretical convergence has been studied [FJK$^+$15], using suitable assumptions on (i) the equilibrium measure, (ii) the Metropolis-Hastings kernel (see section 1.3), and (iii) the sequence of learning rates. Under these assumptions, the Wang-Landau algorithm has been proven to converge. The authors proved a central-limit like theorem which give a theoretical convergence speed of $O(1/\sqrt{n})$ where n is the number of steps.

1.2.7 Convex Volume

A polytope $K \subset \mathbb{R}^n$ can be represented as the convex hull of its vertices (V-polytope) or as the intersection of half-spaces (H-polytope). Volume computations are #-hard both for V-polytopes and H-polytopes [DF88]. Here the complexity is either the number of calls of an oracle stating whether a point is in the convex, or the number of constructions of the intersection between a line and the boundary of the convex. Intuitively, the convex hull of a polynomial number of random points in a convex has a volume that goes to zero exponentially fast with the dimension. It can be shown that there is no fast i.e. polynomial time algorithm computing the volume accurately [BF87, Lev97].

For these reasons, probabilistic methods have been developed. We say that a randomized algorithm approximate the volume within a relative error of ε with a probability at least $1 - \delta$ if the approximation V of $\mathrm{Vol}(K)$ satisfies

$$(1 - \varepsilon)\mathrm{Vol}(K) \leq V \leq (1 + \varepsilon)\mathrm{Vol}(K). \tag{1.33}$$

The first probabilistic algorithm [DFK91] has a complexity $O^\star(n^{23})$ with the dimension (where the notation $O^\star(n^2 3)$ ignores log factors), and is overall a polynomial in n, $\frac{1}{\varepsilon}$ and $\log\left(\frac{1}{\delta}\right)$. Most works have chosen to focus on the complexity growth with the dimension, omitting the dependency in ε and δ. Following that, the complexity was gradually improved. In the next two sections, we present two of the latest algorithms proposed, with respective theoretical complexities $O^*(n^5)$ [KLS97] and $O^*(n^3)$ [CV18].

In the following, it is assumed without any loss of generality that the origin 0 is in the convex K. It is also assumed that $r > 0$ and $R > 0$ such that $B(0, r) \subset K \subset B(0, R)$ are known, and that the convex is in isotropic position: $\frac{R}{r} = O(n)$. To make these assumptions true, the convex should undergo a rounding algorithm that we will not focus on here.

Concentric balls algorithm: another density estimation algorithm using MMC

The algorithm introduced in [KLS97] is in essence a Multiphase Monte Carlo algorithm. Let $r_i = 2^{i/n} r$ be a growing sequence of radius and consider the intersection $K_i = B(0, r_i) \cap K$. For $m = n \log(R/r)$, the radius

is $r_m = R$ and thus $K_m = K$. For $i = 0$, $K_0 = B(0, r_0)$. Using the convexity of K, it is easy to prove the following proposition:

Proposition. 1. *For every $i > 0$:*

$$Vol(K_i) \leq Vol(K_{i+1}) \leq 2\, Vol(K_i)$$

Following the idea of section 1.2.4, we rewrite the volume of K as

$$\mathrm{Vol}(K) = \mathrm{Vol}(K_0) \prod_{i=1}^{m} \frac{\mathrm{Vol}(K_i)}{\mathrm{Vol}(K_{i-1})} \tag{1.34}$$

To estimate $\prod_{i=1}^{m} \frac{\mathrm{Vol}(K_{i-1})}{\mathrm{Vol}(K_i)}$, each ratio $R'_i = \frac{\mathrm{Vol}(K_{i-1})}{\mathrm{Vol}(K_i)}$ is estimated separately (see remark 1.8). Following the notations of section 1.2.4, the functions f_i are the indicator functions of K_i, and each R'_i is simply estimated by sampling points uniformly in K_i and counting the proportion in K_{i-1}.

Therefore, we take $N = O(m)$ for each ratio estimation, leading to a $O(m^2)$ total number of samples. Since $m = O(n \log(n))$, the algorithm requires $O^*(n^2)$ samples. The final complexity $O^*(n^5)$ comes from the sampling part: we assumed that we could sample points uniformly in each K_i. It turns out that sampling a point (almost) uniformly costs $O(n^3)$ calls to oracle, leading to a final complexity $O(n^2 n^3) = O(n^5)$.

Remark 1.10. *The previous algorithm is actually computing a histogram of the pushforward measure of the Lebesgue measure on K by the function $x \longrightarrow \|x\|$ with bins $[r_i, r_{i+1}]$.*

Gaussian cooling

This algorithm introduced in [CV18] is based on the same idea as the previous, but instead of using a sequence of indicator functions, it uses a sequence of Gaussians $f_i(x) = e^{-\frac{\|x\|^2}{2\sigma_i^2}}$ of variance σ_i^2. The concentric ball algorithm requires $O^*(n)$ phases because the ratio between the volume of two consecutive balls is bounded by 2. The Gaussian cooling bypasses this limitation by relying on remark 1.7, leading to a drastic reduction of the number of phases required.

Without going into too much details, the algorithm is split in two parts. First the Gaussian volume of the convex is computed (i.e. for $\sigma = 1$), then the volume is computed, each phase requiring $O^*(n^3)$ calls to oracle.

The first part starts with $\sigma_0^2 \sim \frac{1}{n}$ and stops at $\sigma_m^2 = 1$ with

$$\sigma_i^2 = \sigma_{i-1}^2 \left(1 + \frac{1}{\sqrt{n}}\right).$$

This requires $O(\sqrt{n})$ phases, and thus the total number of samples is $O^*(n)$. In addition, for $\sigma = O(1)$, the complexity for generating each sample is $O^\star(()n^2)$, leading to the complexity $O^*(n^3)$ for the computation of the Gaussian volume.

The second part starts with $\sigma_0^2 = 1$ and ends with $\sigma_m^2 = O(n)$ which is large enough to be almost flat. In this case, a better bound for the ratio of two consecutive Gaussian is available, leading to the following cooling:

$$\sigma_i^2 = \sigma_{i-1}^2 \left(1 + \frac{\sigma_{i-1}}{\sqrt{n}}\right). \tag{1.35}$$

Here, the number of calls to oracle required to generate a single sample for a Gaussian of variance σ is $O(\sigma^2 n^2)$ which is bad because $\sigma_m = O(n)$. However, for a given starting σ_k, the number of phases to double σ_k using the previous cooling schedule is $O(\frac{\sqrt{n}}{\sigma_k})$ and so is the number of samples per phase. Therefore, the total complexity to double σ_k is $O(n^2\sigma_k^2)O(\frac{\sqrt{n}}{\sigma_k})O(\frac{\sqrt{n}}{\sigma_k}) = O(n^3)$ which does not depend on σ_k. To go from $\sigma_0^2 = 1$ to $\sigma_m^2 = O(n)$, $O(\log(n))$ doubling of σ is required, hence the total complexity of $O^*(n^3)$.

All in all, this algorithm improves both the number of phases and the complexity of generating each sample, leading to an overall complexity of $O^*(n^3)$ for the computation of the volume of a convex.

1.3 Sampling strategies

The previous section mostly assumed that for any distribution π, a sampling algorithm existed. The purpose of this section is to fill this hole and give a quick review of the different methods to sample random variables from a density π.

1.3.1 Inverse of the Cumulative Density Function method

When $n = 1$, a straightforward way to sample point from π is to use the inverse of the cumulative density function (CDF) $F(x) = \int_{]-\infty,x]} \pi(x)dx$.

Asssuming one can sample $Y \sim \text{unif}([0, 1])$, we define $X = F^{-1}(Y)$, then π is the density of X.

The main drawback of this method is that it requires the computation of the inverse of the CDF, which does not always have an analytic form (for example the gaussian inverse CDF).

1.3.2 Rejection sampling

This method works on $\mathbb{R}^n$ with any probabilty density π [RC13].

Let π and μ be two probabilitiy densities on $\mathbb{R}^n$ such that $\pi \leq M\mu$ for some constant $M \geq 1$. If one can sample a random variable Y with density μ then it is easy to sample a random variable X with density π. Consider an auxiliary random variable U which is uniformly distributed in the interval $[0, 1]$ and which is independent from Y. Consider a sample $(y_1, \ldots, y_m)$ of Y and a sample $(u_1, \ldots, u_m)$ of U. For each i, if $u_i M\mu(y_i) \leq \pi(y_i)$, y_i is accepted otherwise y_i is rejected. It is easy to prove that the accepted y_i form a sample for the density π.

The main drawback of this method is that it can reject a large proportion of y_i. It will be the case when the ratio π/μ has large values.

1.3.3 Markov Chain Monte Carlo (MCMC)

MCMC allows arbitrary distributions on $\mathbb{R}^n$ to be sampled, and is at the core of many algorithms. Its principle is straightforward: to sample a measure π, we build a Markov chain P such that π is the only invariant measure.

Convergence theorems can be found in [RR04a].

Definition. 1.2 (stationary measure)**.** *A measure π is stationary with respect to a Markov chain P if for all A measurable,*

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

Definition. 1.3 (detailed balance / reversible)**.** *A Markov chain P is said to satisfy detailed balance (or reversible) with respect to the measure π if for every A and B measurable,*

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

It is easy to show that if π satisfies detailed balance, then π is a stationary distribution.

Remark 1.11. *Satisfying detailed balance is not sufficient for convergence to a stationary measures. Examples of additional assumptions required can be found in [RR04a]. The proof of convergence of chapter 3 uses such assumptions.*

Metropolis-Hasting

We assume there is a Markov kernel whose probability transition has positive density q with respect to the Lebesgue measure. Given a probability density π on $\mathcal{E}$, Metropolis-Hasting builds a Markov kernel P_π from q. It satisfies detailed balance for π which implies that π is a stationary density for P_π.

Starting from x_t, the construction is as follow:

- sample y according to $q(x_t, .)$
- compute $\alpha(x_t, y) = \min(\frac{\pi(y)q(x_t,y)}{\pi(x_t)q(y,x_t)}, 1)$
- choose $x_{t+1} = \begin{cases} y & \text{with probability} \quad \alpha(x_t, y) \\ x_t & \text{with probability} \quad 1 - \alpha(x_t, y) \end{cases}$

Which gives the following transition kernel:

$$P_\pi(x, dy) = \alpha(x, y)q(x, dy) + \delta_x(dy) \int (1 - \alpha(x, z))q(x, dz) \tag{1.36}$$

where δ_x is the Dirac measure at x.

It is straightforward to check that detailed balance holds [Tie98].

Remark 1.12. *The choice of q is crucial for the convergence speed to the the stationary distribution π.*

1.3.4 Hamiltonian Monte Carlo (HMC)

In the following, we survey the principal ingredients of HMC, referring the reader for the surveys [Bet17, BBKG18] for the details.

From statistical physics to sampling

HMC is an MCMC method that takes root in statistical physics. Recall the result from section 1.1.1: for an Hamiltonian H, the flow of the Hamiltonian preserves any probability distribution that is a function of H (Th. 1.3). In particular, the Boltzmann distribution $\mu(q, p) = e^{\frac{-H(q,p)}{k_B T}}$ is preserved and is effectively sampled by molecular dynamic simulations for systems in contact with a thermostat. The idea behind HMC is to find a suitable Hamiltonian such that sampling the Boltzmann distribution will imply sampling from the desired distribution π.

However finding H such that the Boltzmann distribution and π coincide is not practical in general as it would require π to have a particular form. Hence we only work with the potential energy U. For separable Hamiltonians, the Boltzmann distribution can be written as a product measure: $\mu(q, p) = e^{\frac{-U(q)}{k_B T}} e^{\frac{-K(p)}{k_B T}}$. Therefore taking $U(q) = -\log(\pi(q))$ leads to the following Boltzmann distribution:

$$\mu(q, p) = \pi(q)e^{\frac{-K(p)}{k_B T}}$$

which is a product measure. Hence taking the marginal of the Boltzmann distributions on the positions yields the desired distribution π. The final ingredient is to replace the thermostat by periodically resampling the velocities, bringing an equivalent to collisions with particles from the thermostat.

Algorithm

Now, in what follows, we will omit the temperature T and the Boltzmann constant k_B and work with a dimension-less Hamiltonian H. Assuming that $U(q) = -\log(\pi(q))$ and $K(p) = \frac{1}{2}\|p\|^2$, the HMC algorithm in its simplest form reads as follow:

1. choose a travel time $L > 0$

2. sample the momentum vector $p' \sim \mathcal{N}(0,1)$

3. $(q_{t+1}, p_{t+1}) = \Phi_L(q_t, p')$

Remark 1.13 (Choice of kinetic energy)**.** *It is perfectly possible to change the kinetic energy K of the hamiltonian, as long as one matches step 2 to reflect this change. However, the usual kinetic energy $K(p) = \frac{1}{2}\|p\|^2$ is the most widely used, and as such this work will not investigate the use of other kinetic energies.*

For a given L, the algorithm defines a Markov kernel that we denote P_{HMC}. We can further decompose this kernel as the composition of the two kernels P_{Φ_t} and P_K where P_{Φ_t} is the kernel corresponding to step 3 and P_K to step 2. Then the following theorem holds:

Theorem. 1.6. *μ is invariant by P_{Φ_t} and P_K.*

Proof. μ invariant by P_{Φ_t} is a corollary of theorem 1.3. The invariance by P_K is trivial. □

Theorem. 1.7. *μ is invariant by P_{HMC}.*

Proof. This is a trivial corrolary of the previous theorem. □

Geometric intuition

What precedes explains why HMC samples the correct distributions, but says nothing about the *quality* of the sampling or why one would choose HMC over for example Metropolis-Hasting. Indeed, the success of HMC relates to its ability to cope with cases where the mass of the target distribution is concentrated in a *typical set* of small volume. In such cases, classical *random walks / move sets* used by the Metropolis-Hastings algorithm face difficulties to propose moves inside this set – since they explore the entire space. The interest of HMC is precisely to target such *typical set*, by building a dynamical system which is diffusive in the vicinity of this level set. This is achieved by defining a dynamic system in phase space, namely positions and velocities. Recall that the Hamiltonian $H(q,p)$ is defined as the sum of a potential energy $U(q)$ and a kinetic energy $K(q,p)$.

Geometrically, the level sets of the Hamiltonian (defined by $H = E(= cst)$) define a foliation of the phase space. With this in mind, HMC inherently combines two ingredients: random transitions between energy level sets (achieved by momentum resampling), and an exploration of a given level set by deterministic Hamiltonian trajectories. The orbit of a trajectory i.e. the locii of points visited in phase space may cover the whole level set or a subset of it, and it is important to ensure ergodicity – the temporal expectation over the trajectory should converge to the spatial average over the orbit.

Practical implementation - numerical integration

The main difficulty in using HMC is the generation of Hamiltonian trajectories. When such trajectories can be computed analytically, then, a trivial sampling algorithm is readily available. If not, symplectic numerical integrators, which aim at preserving phase space volume, are used. Numerical errors inherent to the discretization are handled using a Metropolis-Hasting scheme on phase space. Since in this manuscript, exact trajectories are known, we do not dwell into further details a refers the intersted reader to [Bet17, Rad12].

1.3.5 Uniform sampling in a bounded domain

It is a common case to require sampling from a bounded domain $A \in \mathbb{R}^n$. For example, for the convex body algorithm, uniform or Gaussian sampling in a convex is required.

Ball walk

Let's analyse what happens for a simple Ball walk: for a point x_t, the random walk picks a point x_{t+1} uniformly in the ball $B(x_t, r)$, with $r > 0$. To respect the boundary without introducing bias in the sampled measure, if a point is sampled outside of A when generating $t + 1$, then we pick $x_{t+1} = x_t$.

Not moving when the point sampled is outside of A (i.e. $x_{t+1} = x_t$) is necessary to sample the correct distribution. Consider the sequence x'_t, constructed from the same sequence of random numbers, with the only difference being that for a point x'_t, we sample point in the ball centered on x'_t until a point in A is found, and defined x'_{t+1} as this point. Here, the sequence x'_t is simply the sequence x_t with the "multiplicity" removed. Then the sequence x'_t does not satisfy detailed balance for the Lebesgue measure of A. Indeed, consider a point $x \in A$ such that $B(x, r) \subset A$, and a point $y \in B(x, r)$ near the boundary of A such that $Vol(B(y, r) \cap A) < Vol(B(y, r))$. The probability of going from x to y is

$$p'_{x,y} = \frac{1}{Vol(B(x, r))}$$

while the probability of going from y to x is

$$p'_{y,x} = \frac{1}{Vol(B(x, r) \cap A}$$

and therefore, $p'_{x,y} \neq p'_{y,x}$. Thus, detailed balance is not satisfied. Another intuitive explanation is that points near the boundary are less accessible, and therefore counting them several times is required.

Should the set A be (locally) a cone, the Ball walk faces difficulties to leave the neighbourhood of the apex of the cone, especially in high dimensions, since most of the volume of a ball with a center near the apex will fall outside of the cone, leading to a high rejection rate. In particular, this phenomenon occurs near the vertices of a polytope.

Hit and Run

Hit and Run is another type of random walk widely used for convex volume computation that automatically generates points in the domain A. Starting from x_t, it goes as follows to generate x_{t+1}:

- sample a direction $u \in S^{n-1}$
- compute the intersections y_1 and y_2 of the line $x_t + \mathbb{R}u$ and the boundary of A.
- sample x_{t+1} uniformly in $[y_1, y_2]$.

which trivially satisfies detailed balance for the Lebesgue measure. For other measures, it is sometimes possible to modify the last step to satisfy detailed balance. This is the case for Gaussian distributions restricted to a convex.

Performances are better than the Ball random walk, but it also faces difficulties near the apex of a cone, as the distance $\|y_1 - y_2\|$ will with a very high probability be very small if x_t is close to the apex, implying that the move will be small (which is still better than not moving at all).

Chapter 2

Wang-Landau Algorithm: an adapted random walk to boost convergence

2.1 Introduction

The Wang-Landau algorithm for density of states calculations. The derivation of observable properties of (bio-)molecular systems at thermodynamic equilibrium relies on statistical physics, with the formalism of stochastic ensembles playing a pivotal role[Wal03, LSR10, Jan12, LB14]. Amidst the various algorithms available, the Wang-Landau (WL) algorithm [WL01, LTE04] is now well known and widely used despite its recent inception, in particular due to its simplicity and genericity. The WL algorithm estimates the density of states (DOS) of a system, which is especially useful to compute partition functions in statistical physics, and more generally observables–e.g. the average energy or the heat capacity. Estimating the DOS is especially challenging in presence of broken ergodicity; in that case, the presence of multiple energy wells prevents the system to efficiently sample the PEL, as it remains confined in selected wells [Pal82, WS14].

To review previous work, it is important to recall that the WL algorithm falls in the realm of adaptive MCMC sampling algorithms. In a nutshell, WL returns an estimation of the DOS in terms of histogram. The bins of the histogram correspond to a partitioning of the energy range of the system. The algorithm resorts to importance sampling, using a biasing function derived from the current estimation of the DOS. Since the limit distribution sought is defined by the density of states, the random walk is build from the Metropolis-Hastings algorithm (M-H), using the current DOS estimate in the rejection rate. (We note in passing that since the DOS values used to define transition probability depend on the history, WL is not a Markov process.) Additionally, a so-called flat histogram rule may be used to count the visits in each energy stratum and update the learning rate when all strata have been evenly visited. These main ingredients recalled, one may observe that numerous improvements were made to the original algorithm [WL01], both in terms of design and analysis of performances. The first key improvement has been the 1/t algorithm which solved the so-called error saturation problem [BP07a, BP07b], in which a constant error on DOS estimates was incurred, due to a too fast reduction of the learning rate. Another key initiative has been to tune the random walk and the energy discretization [BJMD13], as large bins may hinder convergence by keeping the system trapped. To avoid this pitfall, a dynamic maintenance of bins has been proposed, in order to maintain a proper balance of samples across a stratum. Concomitantly, a random walk defined from a mixture of Gaussians has been introduced, in order to attempt moves of the proper size. In a different vein, it has been proposed to speed up convergence resorting to parallelism via multiple walkers [LTE04]. However, this approach should be taken with care, as problems arise when a large number of walkers are used [BP16].

On the mathematical side, for the WL algorithm variant using the flat histogram, the importance of the analytical form of the DOS update rule was established [JR14]. For WL with a deterministic adaptation of the learning rate, to which the 1/t variant belongs, the correctness of the DOS estimates was proved, regardless of the particular analytical expression of the update rule [FJK$^+$15].

Applications. Application-wise, WL has been used on a variety of physical systems, and more recently to biomolecules. Thermodynamics properties of RNA secondary structures were estimated using the WL algorithm [LC10]. Properties of clusters and peptides (up to 8 a.a.) were studied in [PCA+06]. Likewise, the thermodynamics properties of misfolded (containing a helix structure rather than a β-sheet) proteins, such as those involved in mad cow and Creutzfeldt-Jakob diseases, were studied by feeding a coarse grain protein to the 1/t WL algorithm variant [OMG10]. In a similar spirit, a modified flat rule histogram was used in [SA11] to study properties of polymers on a lattice, in the HP model. However, processing continuous models of protein of significant size has remained out of reach so far [JP16].

Contributions. The random walk and the energy discretization influence one another: the average step size of the random walk should be dependent on the size energy bins. For large energy strata, the step size should be large, and small for narrow energy bins. Thus the random walk and bin sizes should not be independent, and the step size of the random walk should depend on local information. Such intricacies have precluded the development of effective WL algorithms to to handle systems as complex as bio-molecules, and the goal of this paper is precisely to improve the convergence speed of the algorithm, especially in high dimensional settings. (NB: our focus is not on asymptotic convergence properties.)

To make a stride towards circumventing these observations, we make three contributions targeting improvements of the convergence speed (section 2.3.1). First, we design a random walk which takes the bin size and local geometric information into account to avoid overstepping strata (section 2.3.2). Second, we tackle the so-called measure concentration problem inherent to high dimensional spaces, which is especially pregnant near local minima, and which decreases convergence speed exponentially fast with the dimension (section 2.3.3). Finally, we introduce a darting move for multimodal distributions (section 2.3.4). In addition, we provide a generic WL implementation, which allows tuning all key building blocks. The source code is being integrated to the SBL Structural Biology Library – see [CD17] and `http://sbl.inria.fr`.

2.2 The Markov kernel P_θ

We briefly recall the notations introduced for Wang-Landau in section 1.2.6. Let $\pi(x)$ be a probability density defined on a subset $\mathcal{E} \subset \mathbb{R}^n$. Let $U : \mathcal{E} \longrightarrow \mathbb{R}$ a potential energy function, and a discretisation of the energy range into $U_{min} = U_0 < U_1 < ... < U_d = U_{max}$. We consider a partition of $\mathcal{E}$ into strata

$$\{\mathcal{E}_1, \ldots, \mathcal{E}_d\} \text{ with } \mathcal{E}_i = U^{-1}([U_{i-1}, U_i]). \tag{2.1}$$

Let $J(x)$ be the index such that $x \in \mathcal{E}_{J(x)}$. The goal of WL is to estimate

$$\theta_i^* = \pi(\mathcal{E}_i) = \int_{\mathcal{E}_i} \pi(x)dx, \forall 1 \leq i \leq d. \tag{2.2}$$

Let $\theta(t)$ be the approximation of θ^* at time t. The detailed description of the Wang-Landau algorithm found in section 1.2.6 assumes that for each θ (representing an histogram), there exists a Markov chain with Markov kernel denoted P_θ such that the probability

$$\pi_\theta(x) = \left(\sum_{i=1}^{d} \frac{\theta_i^*}{\theta_i} \right)^{-1} \frac{\pi(x)}{\theta_{J(x)}} \tag{2.3}$$

is invariant by P_θ.

2.2.1 Construction of the Markov chain

The Markov chain with kernel P_θ is constructed using the Metropolis-Hastings algorithm found in section 1.3.3. We assumes there exists a random walk with transition probability q (common to all θ), and Metropolis-Hastings yields the following transition kernel:

$$P_\theta(x, dy) = q(x, y)\alpha(x, y)dy + \delta_x(dy) \int_{\mathcal{E}} (1 - \alpha(x, z))q(x, dz), \tag{2.4}$$

with

$$\alpha(x,y) = 1 \wedge \frac{\pi_\theta(y)q(y,x)}{\pi_\theta(x)q(x,y)} = 1 \wedge \frac{\pi(y)\theta_{J(x)}q(y,x)}{\pi(x)\theta_{J(y)}q(x,y)} \tag{2.5}$$

which leaves π_θ invariant. The corresponding Markov chain is constructed by drawing a point $y \sim q(x_t, .)$ starting from x_t using the random walk q, then accepting y with probability α. If the point is accepted, $x_{t+1} = y$, otherwise, $x_{t+1} = x_t$.

2.2.2 Convergence rate: further insights

The convergence speed of the algorithm is tightly coupled to the mixing times of the Markov chains P_θ, which is roughly the time t it take for $P_\theta^t(x, .)$ to converge to π_θ for any x, where $P_\theta^t(x, .)$ is $P_\theta(x, .)$ iterated t times. Hence, the choice of the underlying random walk q used to build P_θ is crucial.

In many cases, the bottleneck for the mixing time is the visit of all the energy levels. In [BJMD13], a refinement rule for the discretisation is provided as well as a rule to find suitable parameters for a multi-modal isotropic Gaussian random walk. The paper of Born et al. do not provide any explicit insights on the link between the random walk and the discretisation. However, they use a symmetric random walk. Such random walk will sample the space uniformly. Hence, to obtain a high transition probability between two energy levels, the ratio of their respective volumes must be controlled: should the ratio be too small (or to high), the probability of proposing a move going from the smallest energy level to the biggest is so small that it never occurs. Observe that this restriction vanishes in using a non symmetric random walk, a strategy we will be using.

For multi-modal distributions, the difficulty to switch from one mode to another can also be a bottleneck for the mixing time. In [ASV01], a strategy called darting is proposed. It consists in attempting long range jumps between regions associated to precomputed modes. The knowledge of the volume of the targeted regions allows one to guarantee detailed balance [SW11] whence a procedure sampling the desired distribution. Note that for molecular systems, where Boltzmann distribution yields one mode for each local minimum of the potential energy, local minima can be obtained by gradient descents and associated search methods such as basin hopping and variants [LS87, RDRC16].

2.2.3 MCMC and adaptivity

For general MCMC algorithm, it has been shown that an adaptive random walk can lead to erroneous results [RR07]. Practically, for a given probability π, there might exist a sequence P_i of Markov kernels with limiting distribution π for all i such that for a given X_0, the sequence of random variables defined by $X_i \sim P_i(X_{i-1}, .)$ does not converge to the limiting distribution π. This does not affect the Wang-Landau algorithm itself. However, any adaptivity must be stopped before the end of the algorithm. The choice we make is to stop any adaptivity on q once the flat histogram has been met a given number of times. This number is denoted N_{FHE} in the sequel.

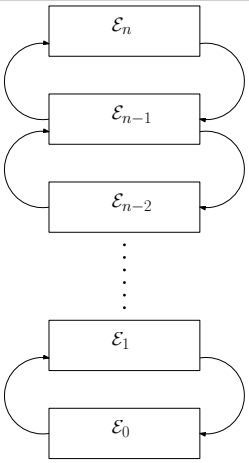
2.3 Improving convergence speed

2.3.1 Rationale

The performances of WL result from a subtle interplay between various ingredients, notably the energy discretization, the topography of the landscape, and the random walk. The improvements presented thereafter target the following difficulties:

- Difficulty 1 – section 2.3.2: topography adapted random walk to avoid overstepping strata. The random walk should exploit the geometry of the landscape, to foster the diffusivity between strata.
- Difficulty 2 – section 2.3.3: curse of dimensionality and concentration phenomena. In high dimensions, when the probability mass is concentrated in a *small* typical set, move sets exploring uniformly the

Figure 2.1 Connectedness between strata is prime to fast convergence. Energy levels may be seen as the nodes of a graph and may be connected in a variety of ways. In this work, we exploit a random walk aiming at describing a *ladder* to connect these nodes.



entire space face difficulties to sample such sets. We introduce a biasing strategy (in terms of directions for the move set), promoting diffusivity between strata.

- Difficulty 3 – section 2.3.4: multimodal distributions. To deal with the case of multimodal distributions, we resort to darting, a strategy meant to connect parts of the energy landscape which are separated by regions of low probability.
- Difficulty 4 – section 2.3.5: energy range discretization. Slow mixing of the random walk may be due to an inappropriate energy discretization. We resort to a refinement strategy to fix such problems.

All in all, we aim for a *ladder-like* random walk as described in Fig. 2.1 which connects each energy level with the one bellow and the one on top with an as high as possible probability.

2.3.2 Overstepping strata

Problem

Strata of small *thickness* tend to be stepped over. This typically happens when the landscape is steep‘ or the discretization is fine. It is thus important to adapt the travel distance of the random walk in such regions.

A Gaussian mixture identical for all strata has been used [BJMD13]. However, the mixture is symmetric (see section 2.2.2) and does not exploit the geometry of the landscape.

Solution

We estimate the local "steepness " of the energy function using a Taylor expansion of the energy. We use an order two expansion since the gradient vanishes near local minima.

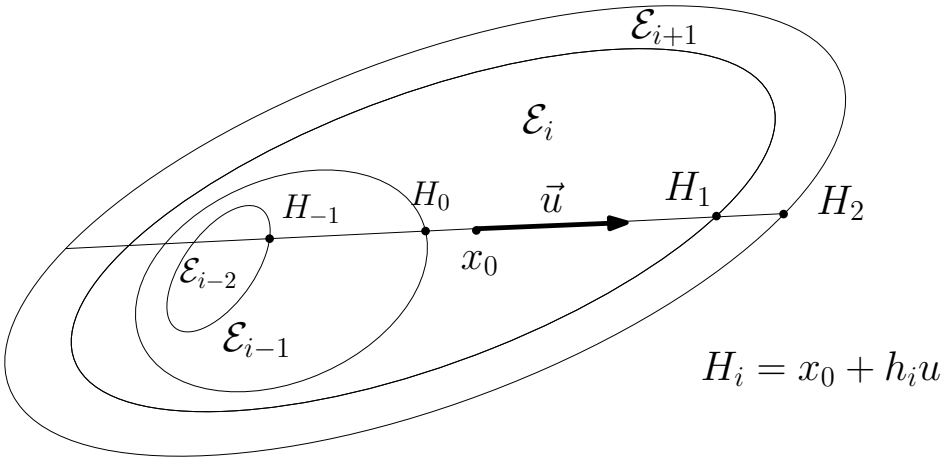
We first sample a direction $\vec{u}$ uniformly at random in the unit sphere $\mathbb{S}^{n-1}$. We then compute the Taylor expansion in the direction u with $h \in \mathbb{R}$:

$$U(x + h\vec{u}) = U(x) + h(\nabla U \cdot \vec{u}) + 1/2h^2(\vec{u}^T \mathrm{Hess}\, \vec{u}). \tag{2.6}$$

Assuming that x is in $\mathcal{E}_i$, we compute using the Taylor expansion the interval $[h_0, h_1]$ such that for $h \in [h_0, h_1]$ (Fig. 2.2)

$$x + h\vec{u} \in \mathcal{E}_i \tag{2.7}$$

Figure 2.2 Exploiting the geometry of the landscape to avoid overstepping strata: move from $x_0 \in \mathcal{E}_i$ should either stay in $\mathcal{E}_i$, move to $\mathcal{E}_{i-1}$, or move to $\mathcal{E}_{i+1}$. The intersection between a random line through x_0 with the level set surfaces of a quadratic approximation of the potential yields points $\{X_i\}$ from which the random walk is defined – see main text.



Doing the same for $\mathcal{E}_{i-1}$ and $\mathcal{E}_{i+1}$ yields $[h_{-1}, h_0]$ and $[h_1, h_2]$. The last steps are to pick any of these 3 intervals with probability 1/3 and to sample h uniformly in the chosen interval.

Doing so effectively adapt the random walk to the local steepness of the energy landscape, allowing multiple scales. Even better, it also changes and adapt to the chosen direction $\vec{u}$.

If the direction u is sampled uniformly at random in the unit sphere $\mathbb{S}^{n-1}$, the probability of going from x to y for any x and y is:

$$q_{flat}(x,y) = \frac{\Gamma(n/2)}{2\pi^{n/2}\|y-x\|^{n-1}} \sum_{i=0}^{2} 1_{[h_{i-1},h_i]}(< y-x, u >) \frac{1}{3|h_i - h_{i-1}|}. \tag{2.8}$$

where the h_i are defined as before, and therefore can be computed from the Taylor expansion at x in direction $\frac{y-x}{\|y-x\|}$. Note that the term before the sum is symmetric in x and y, hence it simplifies when computing $\frac{q_{flat}(y,x)}{q_{flat}(x,y)}$.

Should the direction u be sampled with respect to a probability distribution with density P_{dir} on $\mathbb{S}^{n-1}$, the probability of going from x to y is:

$$q_{flat}(x,y) = P_{dir}\left(\frac{y-x}{\|y-x\|}\right) \frac{1}{\|y-x\|^{n-1}} \sum_{i=0}^{2} 1_{[h_{i-1},h_i]}(< y-x, u >) \frac{1}{3|h_i - h_{i-1}|} \tag{2.9}$$

Remark 2.1. *Equation 2.6 explicitly uses the Hessian. In practice, the second order directional term is estimated numerically using the gradient.*

Remark 2.2. *In the ideal case where the level sets are hyperplanes (meaning that the gradient dominates), the Metropolis-Hastings correction factor $\frac{q_{flat}(y,x)}{q_{flat}(x,y)}$ introduced by the non symmetry of the random walk when sampling a point in $\mathcal{E}_i$ starting from x_0 in $\mathcal{E}_1$ is $V(\mathcal{E}_i)/V(\mathcal{E}_1)$, which cancels out the metropolis acceptance ratio in Wang-Landau when θ is close to θ^*. It effectively bringing the asymptotic rejection rate of the Wang-Landau random walk to 0.*

2.3.3 High dimensionality and concentration

Problem

As noticed in section 2.3.1, we wish to design a random diffusive across strata, which requires moving down from $\mathcal{E}_i$ to $\mathcal{E}_{i-1}$ and moving up from $\mathcal{E}_i$ to $\mathcal{E}_{i+1}$. Without loss of generality, we focus on the descent in the

sequel.

Consider the problem of lowering the energy by moving from stratum $\mathcal{E}_i$ to $\mathcal{E}_{i-1}$ (Fig. 2.3), starting at point x_0. To do so, a direction in a cone of angle $\alpha(x_0, \mathcal{E}_{i-1})$ must be chosen because any direction outside of this cone never intersects $\mathcal{E}_{i-1}$. As the dimension increases, the probability of sampling a point in a cone of fixed aperture decreases exponentially with the dimension (Fig. 2.4), preventing the random walk to reach the stratum $\mathcal{E}_{i-1}$.

Figure 2.3 Reaching region $\mathcal{E}_{i-1}$ from $\mathcal{E}_i$: cone of suitable directions.

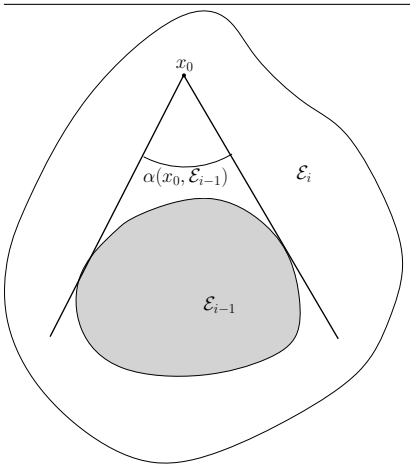
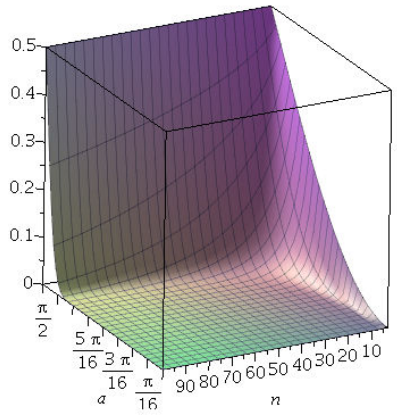


Figure 2.4 Ratio between the area of the spherical cap subtained by an angle θ and that of the whole n-dimensional hemisphere $S^{n-1}/2$. Ranges explored: dimension $n \in [3, 100]$, and angle $\theta \in [0, \pi/2]$.



Solution

The most straightforward way to overcome this problem is to decrease the bin size. Indeed doing so makes $\alpha(x_0, \mathcal{E}_{i-1})$ closer to $\pi/2$. In practice, the bin splitting strategy from [BJMD13] achieves this goal. However, the number of required strata increases with dimension, making this strategy less effective. Let $C_{\text{down}}(x_0, \mathcal{E}_{i-1}) \subset S^{n-1}$ the subset of direction which allow moving downward from a point $x_0 \in \mathcal{E}_i$. Ideally, we could avoid splitting bins if we could bias the choice of direction toward this subset. An approximation of $C_{\text{down}}(x_0, \mathcal{E}_{i-1})$ could be found using a full second order Taylor expansion (thus requiring the full Hessian matrix). However this would be computationally very costly and the sampling procedure on such an approximation is unknown. Therefore, we limit ourselves to isotropic estimations of $C_{\text{down}}(x_0, \mathcal{E}_{i-1})$, ie $C_{\text{down}}(x_0, \mathcal{E}_{i-1})$ is estimated by an isotropic cone of angle $\alpha(\mathcal{E}_i, \mathcal{E}_{i-1})$. In addition, we introduce two simplifying assumptions:

- the strata is not too wide,
- the curvature of the strata does not vary to much.

allowing us to approximate $C_{\text{down}}(x_0, \mathcal{E}_{i-1})$ by a cone of direction ∇U and aperture $C_{\text{down}}(\mathcal{E}_i, \mathcal{E}_{i-1})$. In other words, we use the same aperture for every points in a energy level. The final ingredient is to estimate the aperture $\alpha(\mathcal{E}_i, \mathcal{E}_{i-1})$ during the runtime of the algorithm until the flat histogram has been reached N_{FHE} times – see section 2.2.3.

Estimating the angle. For a stratum $\mathcal{E}_i$, we wish to select an angle amidst a predefined set $\{\alpha_i^{(0)}, ..., \alpha_i^{(k)}\}$. We apply the following procedure–which is independent from the generation of x_{t+1}. For a given point $x_t \in \mathcal{E}_i$

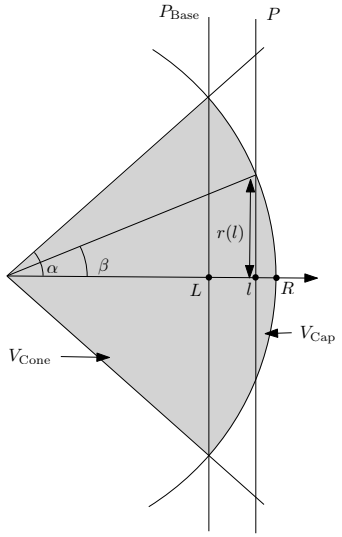
sampled by Wang-Landau, consider the cones of apex x_t, axis $\nabla U(x_t)$, and aperture angles $\alpha_i^{(j)}$. We sample M directions uniformly in each cone, and check for each such direction whether the stratum $\mathcal{E}_{i-1}$ can be reached. Once a prescribed number of points x_t has been processed, we compute the probability of picking a direction which reaches $\mathcal{E}_{i-1}$ (or respectively $\mathcal{E}_{i+1}$) for each cone. Then, we select the largest angle such that this probability is larger than a user defined threshold. If no such angle exists, we use the fallback strategy of section 2.3.5.

Remark 2.3. *The previous strategy calls for the following comments:*

- *The aperture angle of the cone is actually critical. Consider the set of directions delimited on S^{d-1} by a given cone. When the dimension increases, the mass of this set of directions concentrates on the boundary of the cone. Therefore, if the cone aperture is overestimated, sampled directions will end up with high probability in this region, and the corresponding random walk will miss the targeted stratum.*
- *Since the used cones are isotropic, the method is not suited to handle highly non isotropic cases. However, moderately non isotropic cases are handled as well – see Experiments.*
- *Even if a suitable cone is found by the previous procedure, the Metropolis Hasting acceptance rate might be low – for instance if strata are too wide or if the curvatures of level sets non constant.*

Uniform direction in a cone. Sampling a uniform direction in a cone is non trivial. In the following, we sketch the algorithm, and refer the reader to the supplemental for full details (Fig. 2.5 and SI section 2.6).

Figure 2.5 Uniform sampling within a conic region. The volume defined by the grey region is the union of a cone and of a spherical cap.



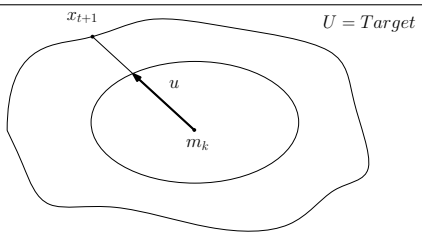
The algorithm proceeds in 3 steps:

- (i) Decide whether one samples from the cone or the spherical cap,
- (ii) Pick a slice in the cone or spherical cap,
- (iii) Sample the slice.

More formally:

- (i) Draw $u \in [0, V_{n,\alpha}^{\text{Cone}}(1) + V_{n,\alpha}^{\text{Cap}}(1)]$. If $u < V_{n,\alpha}^{\text{Cone}}(1)$, we pick in the cone (ie $l < L$), else we pick in the cap (ie $l > L$).

Figure 2.6 Darting: reaching a prescribed energy level set via line search in direction u. The line search starts from minimum m_k with target energy $Target$.



- (ii) Pick a slice of the cone or the cap at distance l from the center, using the density

$$f_{cone}(l) = C_{cone} r(l)^{n-1} 1_{l \leq L} = C_{cone} \tan(\alpha)^{n-1} l^{n-1} 1_{l \leq L} \tag{2.10}$$

  or

$$f_{cap}(l) = C_{cap} r(l)^{n-1} 1_{l > L} = C_{cap} (1 - l^2)^{\frac{n-1}{2}} 1_{l > L} \tag{2.11}$$

  with C_{cone} and C_{cap} normalization constants used to define probability densities.

- (iii) Draw uniformly at random in the corresponding $n-1$ ball, using the density from Eq. (2.23).

2.3.4 Handling multimodal distributions via darting

Problem

For classical random walks, transitions between minima of multimodal distributions are rare events, inducing long mixing times.

Solution

Assuming one has a *a priori* knowledge of the positions $m_1, .., m_K$ of theses minima, it is natural to introduce another type of move allowing jumps from one minima to another. To implement this, we use a darting strategy – see [ASV01, SW11] and section 2.2.2.

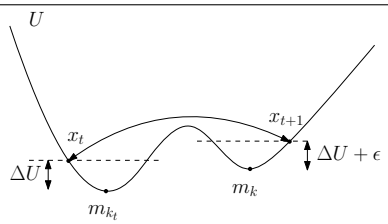
Darting in its simplest form defines a radius ρ, then add transitions between the balls $B(m_i, \rho)$. In practice, if $x_t \in B(m_i, \rho)$, one picks a ball at random, call its index j, and proposes the following move: $x_{t+1} \sim Unif(B(m_j, \rho))$. However the balls $B(m_i, \rho)$ do not match the level set surfaces of their respective basins. Hence $U(x_{t+1}) - U(m_j)$ might be much larger than $U(x_t) - U(m_i)$, leading to poor acceptance rates in the Wang-Landau algorithm.

Choice of the candidate point. Denoting m_{k_t} the local minimum whose basin contains x_t, define $\Delta U_t = U(x_t) - U(m_{k_t})$. Our rationale to optimize the acceptance ratio in Wang-Landau is to control both $\Delta U_t - \Delta U_{t+1}$ and the ratio $q(y,x)/q(x,y)$. For the former, we proceed as follows in two steps. First, we chose a target energy. For a given x_t, let k be the index of the minimum chosen at random. For some $\beta > 0$, we choose a target energy

$$T_U \sim Unif(U(m_k) + \Delta U_t - \beta, U(m_k) + \Delta U_t + \beta). \tag{2.12}$$

Second, we propose a point x_{t+1} such that $U(x_{t+1}) = T_U$ (Fig. 2.7). To this end, we sample a direction u uniformly in the ellipsoid defined by $v^T H(k) v = 1$ where $H(k)$ is the Hessian of U at m_k. Then we do a line search to find the intersection between the target energy T_U and the half line $m_k + \mathbb{R}^+ u$ (Fig. 2.6).

Figure 2.7 Handling multimodal distributions via darting: jumping between 2 minima. While darting, the difference of energy with the local minima is controlled to monitor the acceptance rate.



When to jump. The previous strategy requires a line search which is expensive if the target point is far from the local minimum. Furthermore, under some assumptions which are true if jumps are only allowed close to the local minima, the expression of the transition kernel can be simplified. Hence we introduce M a user defined parameter , and the darting move set is only used if $\Delta U_t \leq M$.

Transition kernel. Computing the transition kernel of the darting move is non trivial, but can be done using a suitable change of variable. The full computation is detailed in the appendix – section 2.7, however for the sake of brevity, we only give here the final result. For any x and y in $\mathbb{R}^n$, let k be the closest minima to y, $I_k = [U(m_k) + \Delta U - \beta, U(m_k) + \Delta U + \beta]$, λ_i and e_i the eigenvalues and eigenvectors of the Hessian of U at m_k. Finally, let

$$l = \sqrt{\sum_i \lambda_i < y - m_k, e_i >^2}. \tag{2.13}$$

Using the latter, the transition probability is given by:

$$q_{dart}(x, y) = \frac{1}{K} \frac{\Gamma\left(\frac{n}{2}\right) 2\beta}{2\pi^{\frac{n}{2}}} 1_{I_k}(U(y)) \frac{1}{l^n} \nabla U(y)^T (y - m_k) \prod_{i \leq n} \sqrt{\lambda_i}. \tag{2.14}$$

2.3.5 Splitting energy bins

As a general rule, more bins means aiming for a more precise description of the density of state. This implies a slower convergence speed. Thus we only split bins when the cone strategy fails, as a fallback. We monitor the failure of the cone strategy by computing the proportion of steps in which the random walk has the *possibility* to go up or down in energy (see 2.3.2) and the success rate of the metropolis hasting criterion when going up or down. If either of these statistics are too low for a given bin, the bin is halved.

2.3.6 The random walk

The final random walk combines the previous random walks. Let $p(x) = (p_{overstep}(x), p_{cone}(x), p_{darting}(x))$ such that $p_{flat}(x) + p_{cone}(x) + p_{darting}(x) = 1$ for all $x \in \mathbb{R}^n$. The final random walk is:

- chose one of the random walk at random with probability vector $p(x_t)$
- sample the point x_{t+1} according to the chosen random walk

Such random walk has the following transition probability:

$$q(x, y) = p_{overstep}(x) q_{flat}(x, y) + p_{cone}(x) q_{cone}(x, y) + p_{darting}(x) q_{darting}(x, y) \tag{2.15}$$

Remark 2.4. *Since the random walk q is used in the Metropolis-Hastings algorithm, it is crucial to be able to compute $q(x, y)$ for any x and y.*

2.4 Experiments

2.4.1 Setup

Statistics of interest

Our experiments target three points: correctness, mixing time, and ability to handle complex systems. For the sake of conciseness, time t refers to t steps of Wang-Landau.

Correctness, stability, and convergence. When an analytical solution for θ^* is known, we simply resort to the relative error for estimates at time t, defined by:

$$\text{error}(t) = \sum_i \frac{|\theta_i^* - \theta_i(t)|}{\theta_i^*}. \tag{2.16}$$

We plot this function along time.

When no analytical solution is known–see the dialanine model below, we assess convergence in two ways, based on several runs ($N = 60$). First, we plot an observable along time, akin to the partition function, at a fixed temperature:

$$\frac{Z}{\lambda(\mathcal{E})} = \frac{1}{\lambda(\mathcal{E})} \int_{\mathcal{E}} \exp\left(-U(x)/kT\right) \approx \sum_{\text{Energy levels U}} \theta_i^* \exp\left(-U/kT\right). \tag{2.17}$$

Second, we provide box plots on a per bin basis. We also resort to violin plots when more details are required–in terms of distribution modes.

Mixing time. A classical assessment of the mixing time is in terms of auto-correlation as a function of the lag time [RR01]. In our setting, where diffusivity across energy strata is targeted, a simpler proxy for the mixing time of P_θ is provided by the so-called climbing and descending times:

Definition. 2.1. *A* climb *across d strata is defined by two times t_0 and t_1 such that*

- $x_{t_0} \in \mathcal{E}_0$ *and* $x_{t_0-1} \notin \mathcal{E}_0$.
- $x_{t_1} \in \mathcal{E}_{d-1}$ *and* $\forall t \in [t_0, t_1[$, $x_t \notin \mathcal{E}_{d-1}$.

The climbing time *is then $t_1 - t_0$.*

Note that we do not normalize the climb times by the number of strata. Indeed, as seen in section 2.3.5, increasing the number of strata can decrease the mixing time. Hence the number of strata is a parameter tuned for convergence speed and therefore should not be taken into account when measuring mixing time.

In the context of multi-modal distributions, another proxy for the mixing is the time taken by the random walk to go from one mode to the other. To that end, we monitor the time evolution of the proportion of time spent in one of the modes.

Contenders

As a yardstick, we compare our random walk (section 2.3.6) against the isotropic Gaussian random walk. However, while our random walk do not require parameter tuning, the Gaussian variance needs to be tuned for a fair comparison [RR01]. Hence we compare with 3 Gaussian walks with high, adequate and low variances with respect to the tuned variance.

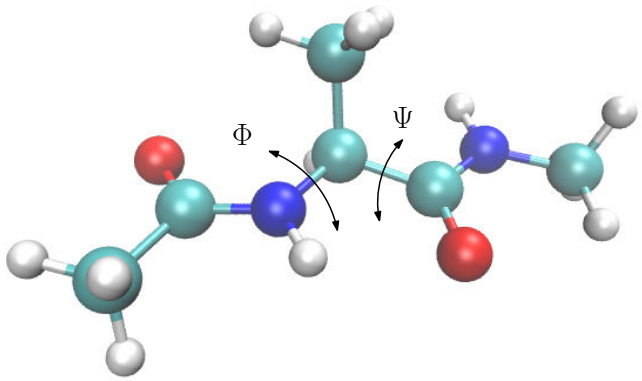
Models

Analytical models. We study three analytical models. The first model is the isotropic harmonic potential. The second is a non isotropic harmonic potential, to ensure that the algorithm behaves correctly in non trivial settings (Remark 2.3). The last model is a potential with two local minimum designed to study darting.

Molecular model. Finally, our last system is a classical toy molecular system, namely the blocked alanine peptide Ace-Ala-Nme (Fig 2.8), referred to as dialanine for the sake of conciseness. This system underwent extensive thermodynamic studies, using techniques as diverse as molecular dynamics [Smi99], Monte Carlo and energy landscape based methods [SW13], or dimensionality reduction methods [SCK10].

We use the amber99-sb force field in vacuum and aim to compute the density of state between -21 kcal/mol and 4 kcal/mol associated to one local minima – by enforcing the simulation to remain inside the basin of this local minimum ($\phi = 59.8862, \psi = -35.5193$).

Figure 2.8 Dialanine (Ace-Ala-Nme) and the two dihedral angles Φ and Ψ



Additional informations about my work on the implementation of force fields in the Structural Biology Library is available in Appendix A.1

2.4.2 Results

Single well potential

We study here the simple harmonic model

$$U(x) = \sum x_i^2$$

with state space the unit ball in dimension $n = 30$. Since the exact result is know, we plot the exact error.

Let us first focus on the error (Fig. 2.9(Top)). The cone strategy yields the best results; all remaining strategies fail to converge in the imparted time (10^7 steps). This radical improvement owes to a climb time orders of magnitude smaller for the cone strategy (Fig. 2.9(Bottom)).

The comparable performances of the isotropic Gaussian random walk with ours (with cone off) owes to the fact that on this example, with the chosen discretization, strata overstepping is not critical.

Single well potential - no isotropic

To challenge all methods with a non-isotropic case (Remark 2.3), we use the following non isotropic potential energy:

$$U(x) = \sum_{i=1}^{n} i x_i^2$$

Also in dimension $n = 30$, the results are on par with the isotropic case (Fig. 2.10).

Figure 2.9 Isotropic single well in dimension 30: comparison of the five random walks. The five random walks used are the three Gaussian based RW, plus the improved random walk with and without the cone improvement. **(Top) Comparison of the evolution of relative error – Eq. (2.16) (Bottom) Box plot of the climbing times.**

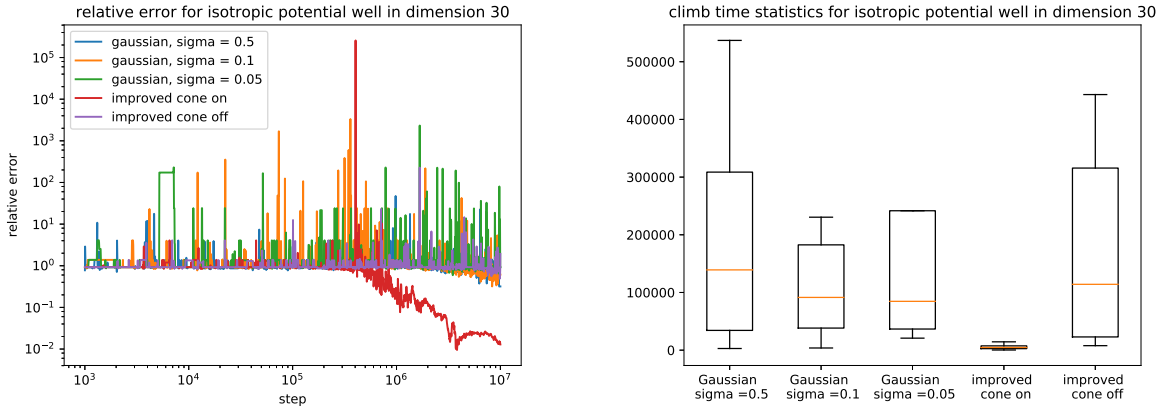


Figure 2.10 Non isotropic single well in dimension 30: comparison of the five random walks. The five random walks used are the three Gaussian based RW, plus the improved random walk with and without the cone improvement. **(Top) Comparison of the evolution of relative error – Eq. (2.16) (Bottom) Box plot of the climbing times.**

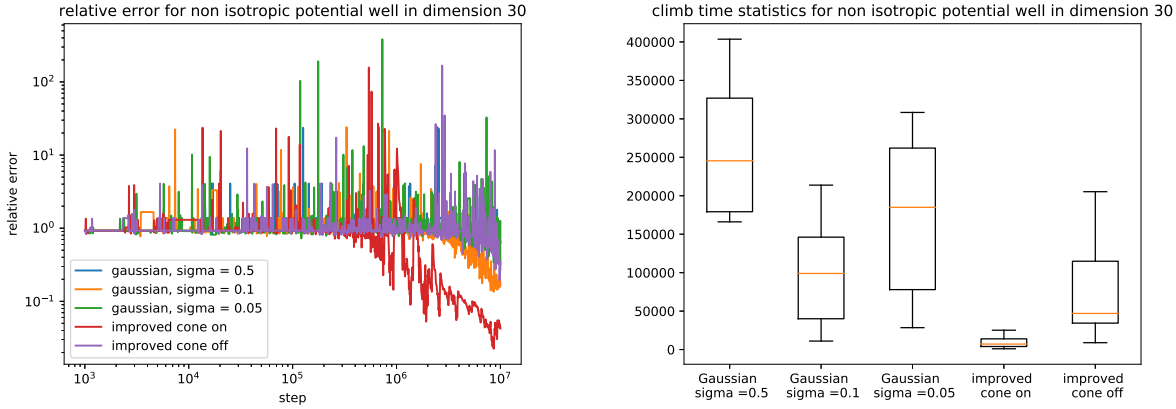
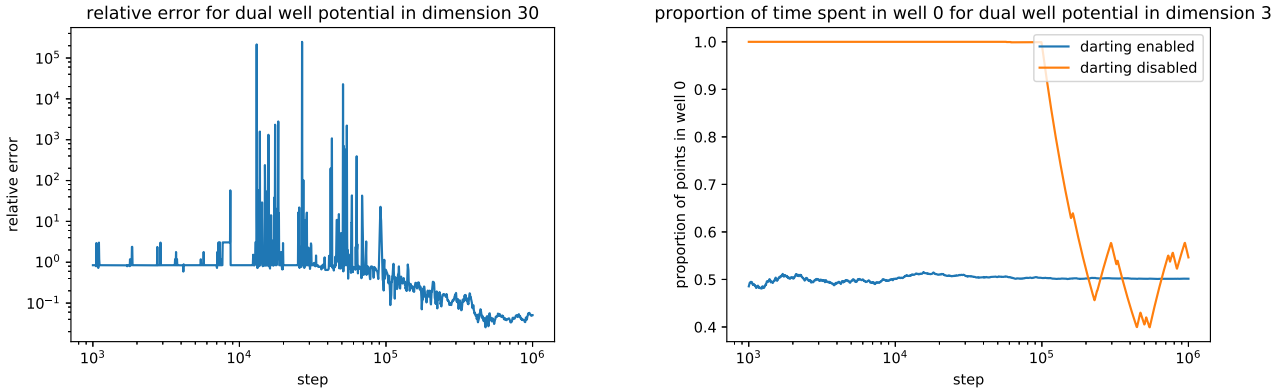


Figure 2.11 Dual well potential in dimension 30: analysis of darting. (Top) Evolution of relative error of Eq. (2.16) when using darting. (Bottom) Comparison of time spent in the first well with and without darting.



Dual wells potential: darting

To challenge darting, we use the usual one-dimensional dual well potential energy function $x^4 - x^2$, and add a quadratic potential in other dimensions:

$$U(x) = x_1^4 - x_1^2 + \sum_{i=2}^{n} x_i^2.$$

This potential energy has 2 local minimum at $(-\frac{1}{\sqrt{2}}, 0, ..., 0)$ and $(\frac{1}{\sqrt{2}}, 0, ..., 0)$. The additional coordinates makes it harder to travel from one minimum to the other by making it hard to choose suitable directions. The sampled energy range is $[-0.25, 1]$, allowing the random walk to pass from one minimum to the other.

We setup the darting move set with these two minima, and compare our random walk with and without darting.

Both methods yield correct values (Fig. 2.11(Top)). (Data not shown for darting disabled: since the potential energy function is symmetric for the first coordinate, the algorithm computes the correct value even if it never crosses the energy barrier.)

It appears that the random walk allows crossing the energy barrier almost instantly, while with darting disabled the first jump appears after 10^5 samples (Fig. 2.11(Bottom)). This induces a large difference in the mixing time.

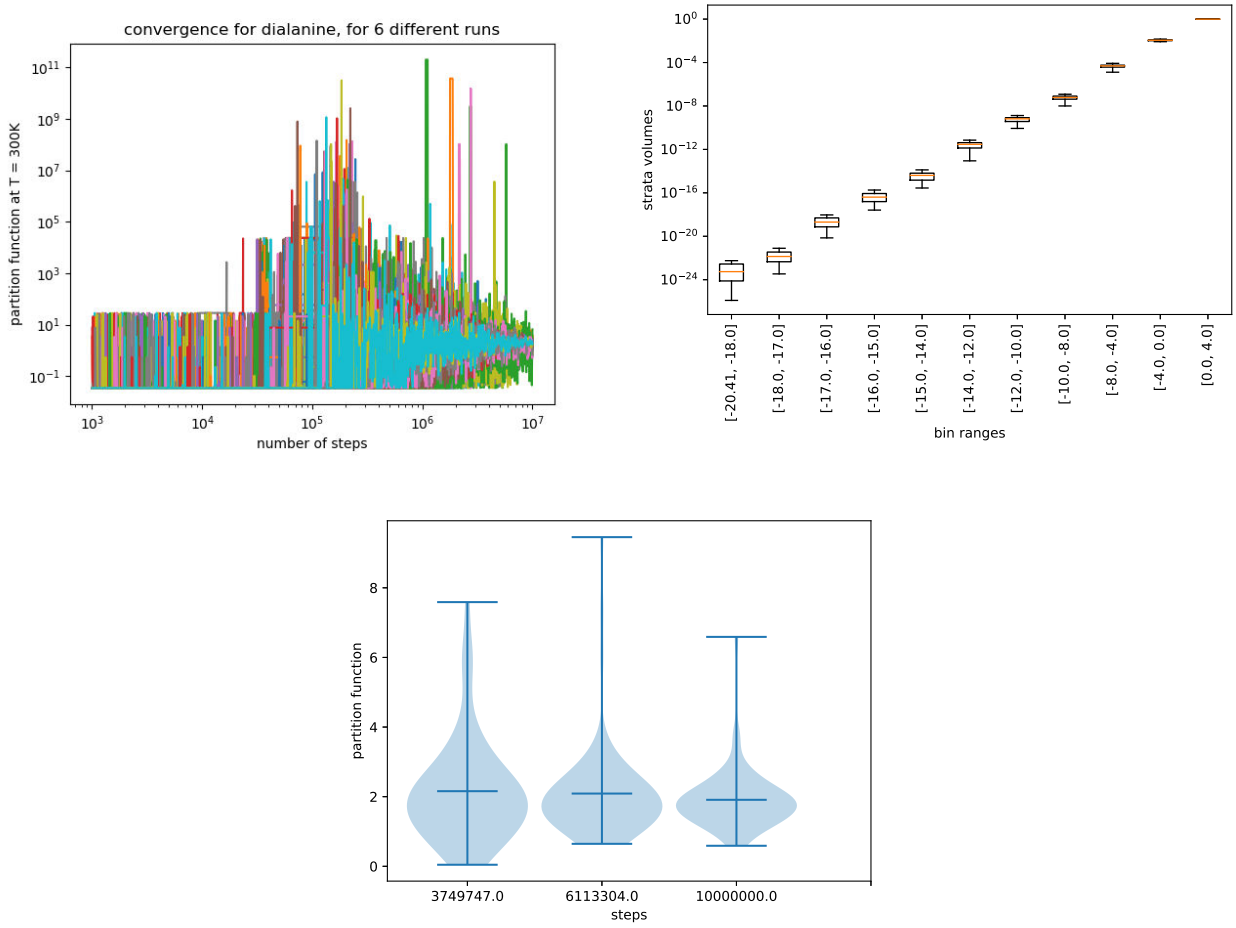
Dialanine

On this system, the results reported thereafter were obtained using the cone improvement, as convergence could not be obtained without it.

To compute the value defined by Eq. (2.17) restricted to the basin of the local minimum with torsion angles, we enforce the simulation to remain within this basin. Checking whether a point is in a given minima basin of attraction requires a minimization of the potential energy. Since this is costly operation, we check this condition every N (=100) steps. If the random walk has escaped, we roll back to the latest point in this basin. (Note that this requires downgrading all statistics and random number generators [CC18a].)

Remark 2.5. *The previous roll back strategy may introduce some bias, as an excursion outside the basin may not be detected. The effectiveness of this strategy relies on a bounded proportion of roll backs, see [CC18a].*

Figure 2.12 Analysis of convergence for dialanine, using amber-69sb forcefield in vacuum. Results averaged over 60 independant simulations. **(Top) evolution of the partition function. (Middle) Box plot of the estimation θ_i for each bin i. (Bottom) Violin plot of the partition function at $T = 300K$, at three different time frames along the course of the simulation.**



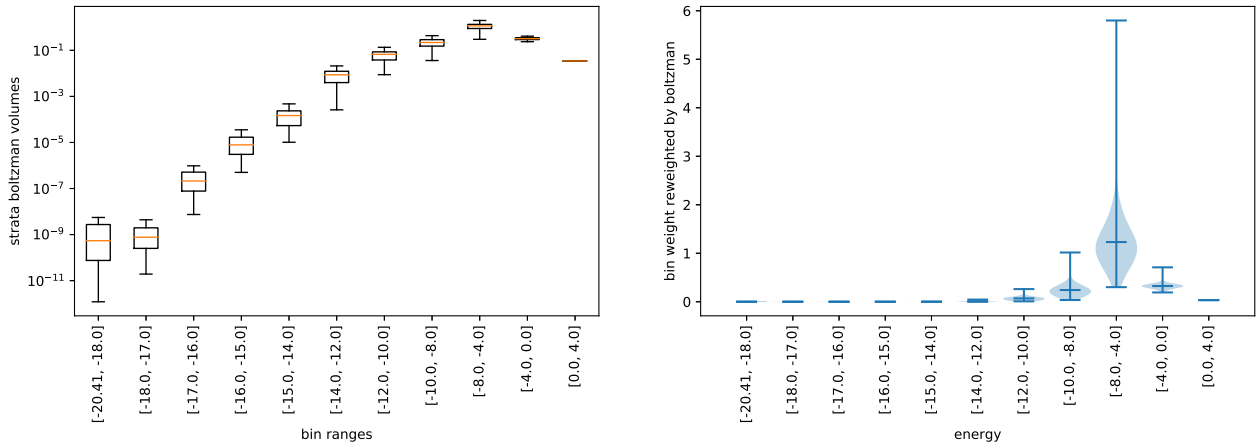
We perform 60 runs, each with 10^7 steps.

To analyze convergence, we plot the time evolution of the observable defined from the partition function at $T = 300K$ (Eq. (2.17), Fig. 2.12(Top)). Since all simulations use the same number of bins, we also provide a box plot for each bin Fig. 2.12(Middle)). Finally, to check whether the observable is unimodal or not, we perform a violin plot at three different time frames along the course of the simulation (Fig. 2.12(Bottom)).

We note that the convergence was reached to a different extent as a function of the volume of strata: the smaller the volume of a stratum, the higher the variance of estimates. This is expected as sampling rare events is always more challenging. It appears, though, that the observable converges (Fig. 2.12(Bottom)), since values concentrate along a single mode.

To get further insights, we study the contribution of individual strata reweighted by Boltzmann's factor (Fig. 2.13). We first note that the energy range used is sufficiently large since contributions of the last stratum is one order of magnitude smaller than the highest one (Fig. 2.13(Top)). The more detailed violin plots–not in log scale, also shows that distributions within strata are unimodal.

Figure 2.13 Analysis of convergence for dialanine, using amber-69sb forcefield in vacuum. Results averaged over 60 independant simulations. **(Top) Box plot of the final bins volume with respect to the Boltzman distribution at $T = 300K$ (Bottom) Violin plot of the final bins volume with respect to the Boltzman distribution at $T = 300K$**



2.5 Outlook

Given a physical system characterized by an energy, the Wang-Landau algorithm is a stochastic method returning an estimation of the density of states in terms of histogram. A core component of the method is the random walk used to navigate between the strata i.e. the preimages in configuration space of the energy slices defining the histogram. In this work, we make an explicit link between the convergence of the Wang-Landau algorithm and mixing properties of the underlying random walk. This analysis prompted the development of a novel, parameter-free random walk. This random walk embarks three components which respectively target the following three difficulties: avoiding overstepping strata, coping with the curse of dimensionality, and accommodating multi-model distributions. The geometry awareness removes the necessity of tuning the variance of the random walk while performing well with strata of varying width. The cone improvement is crucial to deal with concentration phenomena in high dimensional problems. Darting is necessary for multimodals distributions with low transition probabilities between modes. The performances of our random walk are assessed by measuring so-called climbing times which quantify the diffusivity across strata. All in all, the resulting Wang-Landau algorithm is effective is computing observables for small biomolecules, within hours on a laptop computer.

Our work calls for developments on two types of questions. On the design side, while our random work operates in Cartesian coordinates, switching to internal coordinates is an appealing strategy to handle biomolecules whose conformational changes are best described by valence and torsion angles. On the analysis side, our assessment is experimental and prompts challenging analysis issues. On the one hand, a rigorous analysis of the mixing time of our random walk would provided insights on which geometric features of the conformational space / landscape matter. On the other hand, bridging the gap between the mixing time of the random walk and the convergence speed of WL would be of high interest.

2.6 Appendix: uniform sampling in a hypercone

We wish to sample uniformly at random in the intersection of a cone of aperture α intersected with a n-dimensional ball $B^n(R)$ as described in Fig.2.5. The algorithm and the calculations use the notations of Fig. 2.5.

2.6.1 Pre-requisites

Special functions. We shall need the Beta and incomplete Beta functions, defined by

$$\begin{cases} B(a,b) = \int_0^1 t^{x-1}(1-t)^{y-1}dt, \\ B(x;a,b) = \int_0^x t^{x-1}(1-t)^{y-1}dt(\text{ with } 0<x<1). \end{cases} \tag{2.18}$$

Using both, one defines the regularized incomplete Beta factor

$$I_x(a,b) = \frac{B(x;a,b)}{B(a,b)}. \tag{2.19}$$

Spheres and balls: surface and volume. The surface area of a sphere of a $n-1$ sphere $S^{n-1}(R)$ of radius R in $\mathbb{R}^d$

$$\text{Area}_{n-1}(R) = R^{n-1}\frac{2\ \pi^{n/2}}{\Gamma\left(d/2\right)} \equiv R^{n-1}A_n. \tag{2.20}$$

The volume of the corresponding ball $B^n(R)$ satisfies

$$\text{Vol}_n(R) = R\frac{\text{Area}_n(R)}{n} = R^n\frac{2}{n}\frac{\pi^{n/2}}{\Gamma\left(n/2\right)} = R^n\frac{\pi^{n/2}}{\Gamma\left(n/2+1\right)} \equiv R^nV_n. \tag{2.21}$$

To generate a point X uniformly at random on the unit the unit sphere S^n, we generate a point $X = (x_1,\ldots,x_n)^t$ whose coordinates are iid Gaussian with $\mu = 0$ and $\sigma = 1$. The corresponding density is given by

$$f_G(X) = \frac{1}{(2\pi)^{n/2}}e^{-\frac{x_1^2+x_2^2+\cdots+x_n^2}{2}}. \tag{2.22}$$

To obtain a unit vector, we normalize the latter as $\frac{X}{\|X\|}$. (NB: due to normalization the coordinates of this vector are not independent.)

Random generation within a ball. To generate a point uniformly at random inside $B^n(R=1)$, observe that the volume of $B^n(r) = r^nV_n$. Differentiating yields

$$\frac{d}{dr}(r^nV_n) = dr^{n-1}V_n. \tag{2.23}$$

Therefore one generate a random value using the density dr^{n-1} for $r \in [0,1]$.

Spherical caps of the n-dimensional ball. We consider a conic region inside the n-dimensional ball, consisting of the union of a pyramid and that of a spherical cap defined by the cone of aperture α (Fig. 2.5). Surface and volume of such a cap is easily computed [Li11].

To compute the volume of the cap, we integrate the volume of a $n-1$ dimensional sphere or radius $r\sin\beta$ whose height element is $d(r\cos\beta) = r\sin\beta$:

$$V_{n,\alpha}^{\text{Cap}}(r) = \int_0^\alpha \text{Vol}_{n-1}(r\sin\beta)d(r\cos\beta) = \frac{\text{Vol}_n(r)}{2}I_{\sin^2\alpha}(\frac{n+1}{2},\frac{1}{2}). \tag{2.24}$$

Note that the incomplete Beta factor as the probability for a point of the ball to also be inside the spherical cap.

To compute the surface of the cap, we integrate the area of a $n-1$ dimensional sphere or radius $r\sin\beta$ with arc element $rd\beta$:

$$A_{n,\alpha}^{\text{Cap}}(r) = \int_0^\alpha \text{Area}_{n-1}(r\sin\beta)rd\beta = \frac{\text{Area}_{n-1}(r)}{2}I_{\sin^2\alpha}(\frac{n-1}{2},\frac{1}{2}). \tag{2.25}$$

2.6.2 Algorithm to uniformly sample a hypercone

Sampling from f_{cap}

The previous algorithm requires sampling from f_{cap} defined in eq.(2.11). The most straightforward way to sample from a probability density is to compute the inverse of the cumulative distribution function ($F(x) = \int_{-\infty}^{x} f(y)dy$). This requires to compute a primitive of the density. However, see BEFORE, there is no simple analytic expression for the primitive of f_{cap}. Hence, we fall back to rejection sampling with a well chosen base distribution such that the rejection rate do note depend on the dimension n.

Observe that while $(1-l^2)^{\frac{n-1}{2}}$ do not have a simple primitive, the function $l(1-l^2)^{\frac{n-1}{2}}$ do. Therefore we define

$$g_{cap}(l) = MC_{cap}l(1-l^2)^{\frac{n-1}{2}} \tag{2.26}$$

with M such that for all l, $g_{cap}(l) \geq f_{cap}(l)$ which is required for rejection sampling. The optimal choice for M is:

$$M = \frac{1}{L} = \frac{1}{cos\alpha}$$

. L $\tilde{g}_{cap}$ the renormalized version of g_{cap}. Assuming we can sample point from $\tilde{g}_{cap}$, the acceptance rate for each l in the rejection algorithm used with f_{cap} and g_{cap} is

$$\frac{f_{cap}(l)}{g_{cap}(l)} = \frac{1}{lM} \leq \frac{1}{M}$$

as $l \leq 1$. Hence the acceptance rate do not depend on n and only on α the opening of the cone.

Sampling from $\tilde{g}_{cap}$: To sample from $\tilde{g}_cap$ we compute the inverse of it's cumulative distribution.

Let

$$B(x) = \int_L^x l(1-l^2)^{\frac{n-1}{2}}dl$$

using the change of variable $y = 1-l^2$, we deduce:

$$\begin{aligned} B(x) &= \left[-\frac{(1-y^2)^{(1+n)/2}}{1+n}\right]_L^x \\ &= \frac{(1-L^2)^{(1+n)/2}}{1+n} - \frac{(1-x^2)^{(1+n)/2}}{1+n} \end{aligned}$$

The cumulative distribution for $\tilde{g}_{cap}$ is

$$\begin{aligned} F(x) &= 1_{x>L}\frac{B(x)}{B(1)} \\ &= 1_{x>L}\left(1 - \frac{(1-x^2)^{(1+n)/2}}{(1-L^2)^{(1+n)/2}}\right) \end{aligned}$$

And it's inverse:

$$F^{-1}(x) = \sqrt{1-(1-L^2)(1-x)^{2/(n+1)}}$$

Hence we can sample from $\tilde{g}_{cap}$.

Sampling from f_{cone}

The inverse CDF for f_{cone} is straightforward to compute:

$$F_{cone}^{-1}(x) = Lx^{1/n}$$

Therefore we can sample from f_{cone}.

2.6.3 Changing the cone axis

The previous section algorithm generates a point in a cone whose axis is fixed: $e_1 = (1, 0, ...0)$. In practice, the axis of a cone is aligned with the gradient of the potential energy – Section 2.3.3.

To handle arbitrary cones, we apply a linear transformation. We describe here how to apply this transformation with a contained complexity. Let $d \in \mathbb{R}^n \setminus \{e_1\}$ be the desired axis of the cone.
Let H be the hyperplane orthogonal to e_1. In the algorithm, we generates points in H. Suppose we generate $(x_2, ..., x_n)$ in H. For any orthonrmal basis $\epsilon_2, ..., \epsilon_n$ of H, the points $x_2\epsilon_2 + ... + x_n\epsilon_n$ will have the same distribution in H. Hence we try to find a basis $\epsilon_2, ..., \epsilon_n$ adapted to our problem.

We choose $\epsilon_2 = \frac{d - < d, e_1 > e_1}{\|d - < d, e_1 > e_1\|}$.
We complete this base with $\epsilon_3, ..., \epsilon_n$, and we will see that the choice of these $\epsilon_3, ..., \epsilon_n$ do not matter.
Let R the rotation such that $R(e_1) = d$ and $R(\epsilon_i) = \epsilon_i$ for $i > 2$.

Let $H_0 = Vect(e_1)$, $H_1 = Vect(e_1, \epsilon_2)$ and $H_2 = Vect(\epsilon_3, ..., \epsilon_n)$.
Let $x \in \mathbb{R}^n$. Then there exists u_1, u_2 and v such that

$$x = u_1e_1 + u_2\epsilon_2 + u_3v$$

with $v = x - < x, e_1 > e_1 - < x, \epsilon_2 > \epsilon_2 \in H_2$. u_1, u_2 and v are straightforward to compute. We easily get:

$$R(x) = R(u_1e_1 + u_2\epsilon_2) + u_3v$$

Thus the transformation R can be reduced to a simple rotate in $\mathbb{R}^2$. Let $\theta =< e_1, d >$. Then

$$R(u_1e_1 + u_2\epsilon_2) = u_1d + u_2\left(cos(\theta + \pi/2)e_1 + sin(\theta + \pi/2)\epsilon_2\right)$$

Thus we full transform is as follow:

- compute
$$\epsilon_2 = \frac{d - < d, e_1 > e_1}{\|d - < d, e_1 > e_1\|}$$
- compute $u_1 =< x, e_1 >$, $u_2 =< x, \epsilon_2 >$ and $v = x - u_1e_1 - u_2\epsilon_2$
- compute $\theta = (e_1, x)$ and $\tilde{d} = (cos(\theta + \pi/2)e_1 + sin(\theta + \pi/2)\epsilon_2)$
- $R(x) = u_1d + u_2\tilde{d} + v$

2.7 Appendix: transition probability for darting

2.7.1 Notations

We give here a detailed computation of the transition probability for darting given by eq. 2.14. We use the same notations than in section 2.3.4. Let us write P_{dart} the Markov kernel associated to the darting move. Let x be a point of $\mathcal{E}$. The transition kernel has a density, hence we write $P(x, y)$ instead of $P(x, dy)$. For a minimum k, let $H(k)$ the Hessian of U at m_k. Let $\lambda_1, ..., \lambda_n$ it's eigenvalues and $e_1, .., e_n$ it's eigenvectors as an orthonormal basis. Finally let $A_k \subset \mathcal{E}$ be the basin of attraction of minimum k and let k_x the minimum such that $x \in A_{k_x}$. We consider the following rescaling of state space:

$$h_k(y) = m_k + \sum_i \sqrt{\lambda_i}(y - m_k|e_i)e_i. \tag{2.27}$$

Let $\tilde{U}_k(z) = U(h_k^{-1}(z))$ the potential energy in the rescaled space. Let $\tilde{f}_k(u, T_U)$ the application which associates the first intersection between $m_k + \alpha u$ and $\tilde{U} = T_U + U(m_k)$ with $\alpha > 0$. Formally, $\tilde{f}_k$ is an application defined on $S^{n-1} \times \mathbb{R}^+$.

Let $f_k(u, T_U) = h^{-1}(\tilde{f}(u, T_U))$. Also let $f_{k*}(\mu_{k,x})$ be the pushforward measure of $\mu_{k,x}$ by f_k.
Then, the Markov kernel seen as an operator on measures is given by:

$$P_{dart}(x, .) = \frac{1}{K} \sum_k \frac{f_{k*}(\mu_{k,x})}{\int \mu_{k,x}} \tag{2.28}$$

where $\mu_{k,x}$ is the product measure of the Lebesgue measure on S^{n-1} and the Lebesgue measure of

$$I_k(x) = [U(x) - U(k_x) + U(k) - \beta, U(x) - U(k_x) + U(k) + \beta] \tag{2.29}$$

2.7.2 Assumptions

The following assumption ensures that function f_k defines a bijection between the set of directions and the restriction of the target energy level set surface to the basin of a local minimum:

Assumption 1. *For every local minimum $k \leq K$, $u \in S^{n-1}$, $T_U \in [U(m_k), U(m_k) + M]$, the intersection $\{y | y = m_k + \alpha u, \alpha > 0\} \cap \{y | U(y) = T_U\} \cap A_k$ is a single point. See Fig. 2.14 and Fig. 2.15.*

Doing a line search for every minimum is expensive. Using constant M defined in Section 2.3.4(see paragraph *When to jump*), we introduce the following assumption to simplify Eq. (2.28):

Assumption 2. *For every y such that $U(y) - U(m_{k_y}) \leq M$, then for every $k \leq K$,*

$$\|y - m_{k_y}\| \leq \|y - m_k\|$$

The simplified expression for Eq. (2.28) reads as

$$P(x, y) = \frac{1}{K} \frac{f_{k_y *}(\mu_{k_y,x})}{\int \mu_{k_y,x}}$$

where k_y is the closest minimum to y. As a final observation, assumptions 1 and 2 are true if M is small enough (using a second order Taylor expansion for the proof at the bottom of the local minima)

Figure 2.14 Not allowed by assumption 1 as there are multiple intersection point between a direction and the restriction of an energy level set to a basin.

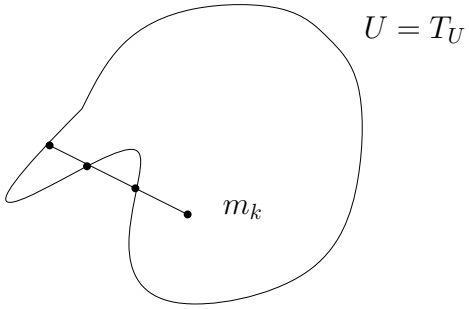
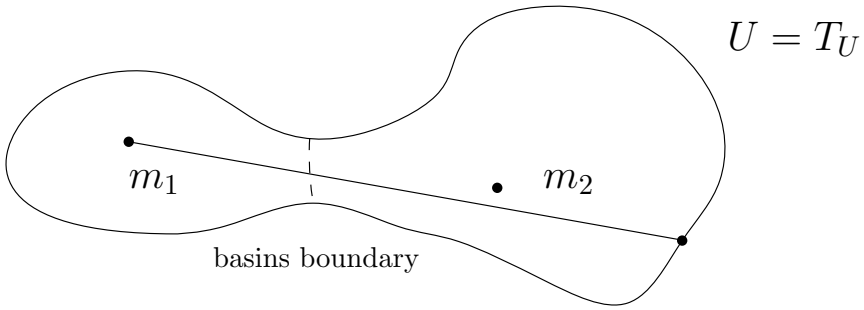


Figure 2.15 Not allowed by assumption 1 as selected directions yield intersection points outside the basin of m_1.



2.7.3 Derivation of the transition probability

Under assumption 1, f_k is a bijection from $S^{n-1} \times [U(m_k), U(m_k) + M]$ to the connected component containing m_k of the set of point $\{y | U(y) \leq U(m_k) + M\}$. Hence it's inverse is well defined. The density of the pushfoward measure can be computed using the usual change of variable formula:

$$f_{k*}(\mu_{k,x})(y) = |J(f_k^{-1})(y)| 1_{I_k}(U(y))$$

For notation simplicity, we consider a fixed k and write $f = f_k$ and $h = h_k$ for the following computation. The inverse of f has the following expression:

$$\tilde{f}^{-1}(z) = \left(\frac{z - m_k}{\|z - m_k\|}, \tilde{U}(z) \right)$$

Let $z = h(y)$ and $u = \frac{z-m_k}{\|z-m_k\|}$, and choose $w_1, ..., w_{n-1}$ in $\mathbb{R}^n$ such that $w_1, ..., w_{n-1}, u$ is an orthonormal basis of $\mathbb{R}^n$. Let $l = \|z - m_k\|$. Then:

$$\frac{\partial \tilde{f}^{-1}}{\partial w_i}(z) = \left(\frac{1}{l} w_i, \frac{\partial \tilde{U}}{\partial w_i}(y) \right)$$

Observe that $w_1, ..., w_{n-1}$ is an orthonormal basis of the tangent space of S^{n-1} at u. Then considering that $\tilde{f}^{-1}$ is an application from an open set of $\mathbb{R}^n$ to $S^{n-1} \times \mathbb{R}^+$, the Jacobian of f^{-1} becomes:

$$J(\tilde{f}^{-1})(z) = \begin{pmatrix} \frac{1}{l} & 0 & ... & 0 & 0 \\ 0 & \frac{1}{l} & ... & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & ... & \frac{1}{l} & 0 \\ (\nabla \tilde{U}(z) | w_1) & (\nabla \tilde{U}(z) | w_2) & ... & (\nabla \tilde{U}(z) | w_{n-1}) & (\nabla \tilde{U}(z) | u) \end{pmatrix}$$

Hence

$$|J(\tilde{f}^{-1})(z)| = \frac{1}{l^{n-1}} (\nabla \tilde{U}(z) | u)$$

And using $\tilde{U}(z) = U(h^{-1}(z))$,

$$\frac{\partial \tilde{U}}{\partial u}(z) = \nabla U(y)^T J(h^{-1})(z) u \tag{2.30}$$

$$= \nabla U(y)^T J(h^{-1})(z) \frac{z - m_k}{l} \tag{2.31}$$

$$= \nabla U(y)^T J(h^{-1})(z)(h(y) - m_k) \frac{1}{l} \tag{2.32}$$

$$= \nabla U(y)^T (y - m_k) \frac{1}{l} \tag{2.33}$$

Where the simplification in equation 2.33 is justified by the fact that $h(y) - m_k = J(h)(y - m_k) = J(h^{-1})^{-1}(y - m_k)$. Combining the two previous equations:

$$|J(\tilde{f}^{-1})(z)| = \frac{1}{l^n}\nabla U(y)^T(y - m_k)$$

We deduce:

$$|J(f^{-1})(y)| = |J(h)|\frac{1}{l^n}\nabla U(y)^T(y - m_k)$$

The Jacobian matrix of h is easy to compute:

$$|J(h)| = \prod_{i \leq n} \sqrt{\lambda_i}$$

Hence we deduce:

$$f_{k*}(\mu_{k,x})(y) = 1_{I_k}(U(y))\frac{1}{l^n}\nabla U(y)^T(y - m_k)\prod_{i \leq n} \sqrt{\lambda_i} \tag{2.34}$$

The rescaling factor for measure $\mu_{k,x}$ is:

$$\int \mu_{k,x} = \frac{2\pi^{\frac{n}{2}}}{\Gamma\left(\frac{n}{2}\right)2\beta} \tag{2.35}$$

Injecting equations 2.34 and 2.35 into equation 2.28 allows us to compute $P_{dart}(x, y)$.

Chapter 3

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

3.1 Introduction

3.1.1 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 [LB14]. In numerical mathematics, a frequent goal is to generate point following a given target distribution [RC13]. 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 [BGJM11], 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 [BGJM11]. A generic method in this realm, both conceptually elegant and remarkably generic is the Metropolis-Hastings methods, 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) [Bet17, Rad12]. HMC aims at sampling a distribution π 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 [Bet17]. 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 [BEF00]. 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 [LB14]. 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 [DF88]. 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 [BF87, Lev97].

A major breakthrough was the development of the polynomial-time randomized algorithm by Dyer et al. [DFK91]. More recently, it has been shown that (ε, δ) approximations of the volume could be computed in $O^\star(n^4)$ [LLV06]. 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 $O^\star(n^3)$ [CV15, CV18], 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 σ^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 [LK99], hit-and-run (HAR) [Lov99, LV03, LV04, HCT$^+$17], billiard walk [GP14].

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 [aVF14, EF18], 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 [LP17]. 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 [PG14].

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 [PS85]. 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 [MBdD$^+$18].

The case geometric algorithms of 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 worse terminates with an erroneous answer.

Such situations occur near degenerate situations, and manifest systematically even on the simplest expressions in 2D space [**?**]. The design of robust geometric algorithms can be done in a general way using the Exact Geometric Computation paradigm [YD95], as it is done for example in the CGAL [cga] 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.

3.1.2 Contributions

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

1. In section 3.2, 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 3.3, 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 3.4 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 [CV16]. In both cases, we show superior performances over HAR, for dimension up to $n = 50$.

3.2 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 π. The traditional HMC algorithm does not handle boundaries, hence we present a modification of HMC with reflections which is drawn from [AD15] and [GP14, PG14].

3.2.1 HMC with reflections

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 π is the restriction to $\overline{Q}$ of positive smooth function defined on $\mathbb{R}^n$ and we use Φ_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 [AD15] and [GP14, PG14], we modify the flow by forcing reflections on the boundary of Q. This flow, illustrated on Fig. 3.2, is denoted as follow:

$$\begin{cases} \tilde{\Phi}_t : \overline{Q} \times \mathbb{R}^n \to Q \times \mathbb{R}^n \\ (q,p) \mapsto \tilde{\Phi}_t(q,p) \equiv (\tilde{\Phi}_t^{(q)}(q,p), \tilde{\Phi}_t^{(p)}(q,p)). \end{cases} \tag{3.1}$$

Note that the latter equation defines position and velocity upon applying the flow. However, as noted in [GP14, PG14], 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 [GP14, PG14], we cull problematic trajectories by not moving in those cases.

Remark 3.1. *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.*

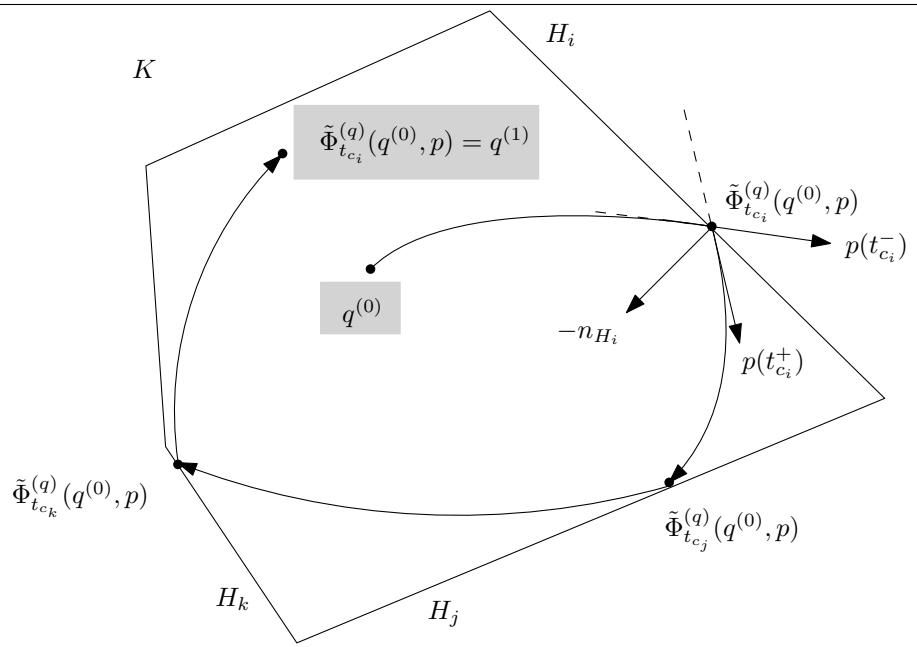
Remark 3.2. *When q is in the boundary of Q and $t > 0$, $\tilde{\Phi}_t(q,p)$ is only defined for the momenta p such that the open half-line with origin q and direction p, is included in Q in a neighborhood of q.*

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. 3.1):

Figure 3.1 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 3.2 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. to 4. define a Markov kernel $P_{\pi,L}$. The full algorithm (steps from 1. to 4.) define a Markov kernel P_π that can be expressed with $P_{\pi,L}$.

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

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

the application from Γ to Γ which corresponds to steps 3 and 4.

Remark 3.3. *For any point $(q,p) \in \Gamma$, if L is small enough, $(q,p) \in \Gamma_L$. However, if L is too large, Γ_L could be empty.*

3.2.2 Measure invariance via detailed balance

We recall the definition of detailed balance:

Definition. 3.1 (detailed balance / reversible)**.** *A Markov chain P is said to satisfy detailed balance (or reversible) with respect to the measure π if for every A and B measurable,*

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

Theorem. 3.1. *Let $A \subset \Gamma$ be a measurable set and let $t \geq 0$ such that $\tilde{\Phi}_t(q,p)$ is defined for every $(q,p) \in \Gamma$. Then*

$$\lambda(\tilde{\Phi}_t^{-1}(A)) = \lambda(A).$$

with λ the Lebesgue measure restricted to Γ.

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

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

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

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

on Γ.

Lemma. 3.1. *The maps $\bar{\Phi}$ and Ψ preserve the Lebesgue measure on Γ_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 [AD15], 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. 3.2.

1. $\Psi \circ \Psi = I$
2. $\Psi(\Gamma_L) \subset \Gamma_L$
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.
4. is clear.
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 Ψ: $\Psi(\Psi(B_A)) \subset \Psi(A_B)$. Using 1., we deduce $B_A \subset \Psi(A_B)$. □

Theorem. 3.2. *P_L and π 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

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

By symmetry:

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

Using lemma 3.2 and then lemma 3.1 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. 3.3. *P_π satisfies detailed balance with respect to π.*

Proof. this is a direct consequence of theorem 3.2 □

3.2.3 Convergence result

Detailed balance ensures that π is invariant by P_π, but it does not imply the convergence of the P^n to π 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 [RR04b]

Definition. 3.2. *A subset $C \subset \mathcal{X}$ is small (or, (n_0, ϵ, ν)-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 $A \subset \mathcal{X}$

Theorem. 3.4. *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*

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

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

Theorem. 3.5. *If the gradient of the potential energy ∇U is bounded on Q and Q is convex then Q is small for P_π.*

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

$$\begin{cases} \tilde{q}(t) & = q(\alpha t) \\ \tilde{p}(t) & = \alpha p(\alpha t). \end{cases} \tag{3.2}$$

The equation of motion becomes:

$$\frac{d\tilde{q}}{dt}(t) = \tilde{p}(t) \tag{3.3}$$

$$\frac{d\tilde{p}}{dt}(t) = \alpha^2 \nabla_q U(\tilde{q}(t)). \tag{3.4}$$

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 α, t, q, p and that ϕ is correctly defined for $\alpha = 0$. In this case, it is easy to see that:

$$\phi(0, t, q, p) = q + pt. \tag{3.5}$$

As π is the restriction of a positive smooth function, ϕ 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 [Bur]), we see that ϕ 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 ν 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 $q \in Q$, $P_\pi(q, .) \geq \epsilon \nu(.)$.

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

$$\gamma(p) = \alpha \exp(-\frac{1}{2}\|\frac{1}{\alpha}p\|^2). \tag{3.6}$$

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 ϕ are bounded on A. Thus there exists $Z > 0$ such that for every $(\alpha, t, q, p) \in A$,

$$\|\phi^q(\alpha, t, q, p) - \phi^q(0, 1, q, p)\| \leq (\alpha + |t-1|)Z \tag{3.7}$$

and

$$\|d_p\phi^q(\alpha,t,q,p) - d_v\phi^q(0,1,q,p)\| \leq (\alpha + |t-1|)Z. \tag{3.8}$$

Using equation 3.5, the two above inequalities are equivalent to

$$\|\phi^q(\alpha,t,q,p) - (q+p)\| \leq (\alpha + |t-1|)Z \tag{3.9}$$

and

$$\|d_p\phi^q(\alpha,t,q,p) - Id_{\mathbb{R}^n}\| \leq (\alpha + |t-1|)Z \tag{3.10}$$

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

$$1/2 < |d_p\phi(\alpha,t,q,p)| < 2. \tag{3.11}$$

Finally, using Lemma 3.3 together with the fact that ∇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 ϕ and Φ 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 3.4 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 3.5 (applied to f translated by $-q$), $B(0,\rho/2) \subset V$. Furthermore, equation 3.11 implies that for every $q' \in V$

$$|df^{-1}(q')| > 1/2. \tag{3.12}$$

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

$$\xi(q') = \nu(f^{-1}(q'))|df^{-1}(q')| \geq \epsilon/2 \tag{3.13}$$

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.

□

Lemma. 3.3. *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. 3.4. *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 φ is a diffeomorphism.*

Proof. The only thing to prove is that φ 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. 3.5. *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 Σ 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 Σ, 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 $B^o(0, r/2) \subset E$. □

3.3 Application: computing the volume of a polytope

In this section, we specialize our generic algorithm (Algorithm 3.1) so as to use it as a building block of the polytope volume calculation from [CV16]. The corresponding pseudo-code is provided in the supplemental section 3.6.

3.3.1 Volume algorithm

The algorithm used in [CV16] compute the volume of a polytope with target relative error ε. 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. $f_i(x) = \exp(-a_i\|x\|^2)$, 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

$$\mathrm{Vol}(K) = \int_K f_0(x)dx \frac{\int_K f_1(x)dx}{\int_K f_0(x)dx} \cdots \frac{\int_K dx}{\int_K f_{m-1}(x)dx} \equiv \int_K f_0(x)dx \prod_{i=1,\ldots,m} R_i \tag{3.14}$$

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

$$\pi_i(x) = \frac{f_i(x)}{\int_K f_i(y)dy} 1_K(x) \tag{3.15}$$

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:

$$R_i^{(k)} = \frac{1}{k} \sum_{j=1}^{k} \frac{f_i(X_j)}{f_{i-1}(X_j)}. \tag{3.16}$$

Instead of using a fixed number of points for the Monte-Carlo integration, a stopping criterion is introduced by [CV16]. 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.

3.3.2 HMC algorithm

This specialization has two advantages: first, trajectories have analytic expressions – see also [PP14]; second, the intersection between a trajectory and n hyperplanes also has a simple analytical expression.

Analytical trajectories. As recalled in section 3.3.1, importance sampling is used to estimate the ratios R_i. Hence, we build an HMC method to sample from $\pi_i(x)$ from eq.3.15.

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:

$$\frac{d^2 q_j}{dt^2}(t) = -2a_i q_j(t) \text{ for } j \leq n. \tag{3.17}$$

Each coordinate is independent and has a solution of the form

$$q_j(t) = C_j \cos(wt + \phi_j) \tag{3.18}$$

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

$$\begin{cases} \cos(\phi_j) & = \frac{q_j(0)}{C_j} \\ \sin(\phi_j) & = \frac{-p_j(0)}{\omega C_j} \end{cases} \tag{3.19}$$

Thus we deduce:

$$\begin{cases} \omega & = \sqrt{2a_i} \\ C_j & = \sqrt{q_j(0)^2 + p_j(0)^2/w^2} \\ \phi_j & = \arctan\left(-\frac{p_j(0)}{q_j(0)\omega}\right) + 1_{\{q_j(0)<0\}}\pi \end{cases} \tag{3.20}$$

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

$$(Ax)_i = b_i. \tag{3.21}$$

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 ω_{n_H},C_{n_H} and ϕ_{n_H} be the parameters of the trajectory along direction n_H. Then finding the intersection times is equivalent to solve the equation for t:

$$C_{n_H} \cos(\omega_{n_H} t + \phi_{n_H}) = b_i \tag{3.22}$$

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

$$\begin{cases} t_1 &= (\arccos(b_i/C_{n_H}) - \phi_{n_H})/\omega_{n_H}. \\ t_2 &= (-\arccos(b_i/C_{n_H}) - \phi_{n_H})/\omega_{n_H}. \end{cases} \tag{3.23}$$

On 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.

3.3.3 HMC implementation based on interval arithmetic

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 3.3.3.

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 [Mül01].

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

`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 that contains 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.

Robust operations

Our robust implementation calls for the following operations (i) Computing the trajectory parameters – Eq. (3.20), (ii) Finding the exit intersection time – Eq. (3.23), (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 3.6; in particular, Algorithm 3 refines Algorithm 3.1 based on these robust primitives.

Trajectory parameters. Following Eq. (3.20), we construct the numbers C_j, w and ϕ_j as `iRRAM` number types. See Algorithm 4.

Exit intersection time t_c with one hyperplane. Intersecting the trajectory with a hyperplane yields two solutions (Eq. 3.23) 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 5.

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^{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.

Performing one HMC step. Equipped with the previous operations, our robust implementation – Algorithm 3, 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. (3.23). 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^{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^{o} since

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

3.3.4 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. 3.6. *Let $P_k^{(n)}$ the distribution after k steps in dimension n and g_n the isotropic Gaussian 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 $k=-\log n\frac{\log(\epsilon)}{\log(\rho)}$*

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

where the norm is the total variation. Furthermore, the average complexity (i.e 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.3.18, 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 3.4 combined with theorem 3.5 for the 1-D Gaussian restricted to $[0,1]$, we deduce that there exists $0<\rho<1$ such that for all $x\in[0,1]$,

$$\|P_k(x,.)-g\|\leq\rho^k.$$

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 $P_k^{(n)}(x,y)=P_k(x_1,y_1)P_k^{(n-1)}((x_2,...,x_n),(y_2,...,y_n))$ and $g_n(x)=g(x_1)g_{n-1}((x_2,...,x_n))$. The total variation distance can be written as

$$\begin{aligned}
\|P_k^{(n)}(x,.)-g_n\| &= \frac{1}{2}\int_{[0,1]^n}\left|P^{(n)}(x,y)-g_n(y)\right|dy\\
&= \frac{1}{2}\int_{[0,1]\times[0,1]^{n-1}}\left|P_k(x,y_1)P_k^{(n-1)}(x,y_2)-g(y_1)g_{n-1}(y_2)\right|dy_1dy_2\\
&= \frac{1}{2}\int_{[0,1]\times[0,1]^{n-1}}\left|\left(P_k(x,y_1)-g(y_1)+g(y_1)\right)P_k^{(n-1)}(x,y_2)-g(y_1)g_{n-1}(y_2)\right|dy_1dy_2\\
&\leq \frac{1}{2}\int_{[0,1]\times[0,1]^{n-1}}\left|\left(P_k(x,y_1)-g(y_1)\right)P_k^{(n-1)}(x,y_2)\right|dy_1dy_2\\
&+ \frac{1}{2}\int_{[0,1]\times[0,1]^{n-1}}\left|g(y_1)P_k^{(n-1)}(x,y_2)-g(y_1)g_{n-1}(y_2)\right|dy_1dy_2\\
&\leq \frac{1}{2}\int_{[0,1]}\left|P_k(x,y_1)-g(y_1)\right|dy_1\\
&+ \frac{1}{2}\int_{[0,1]^{n-1}}\left|P_k^{(n-1)}(x,y_2)-g_{n-1}(y_2)\right|dy_2
\end{aligned}$$

We deduce

$$\|P_k^{(n)}(x,.) - g_n\| \leq n\|P_k(x,.) - g\| \leq n\rho^k \tag{3.24}$$

Hence for a fixed ϵ, if k satisfies $n\rho^k \leq \epsilon$, then for every x, $\|P_k^{(n)}(x,.) - g_n\| \leq \epsilon$. Thus we take $k = -\log n \frac{\log(\epsilon)}{\log(\rho)}$, 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. □

3.4 Experiments

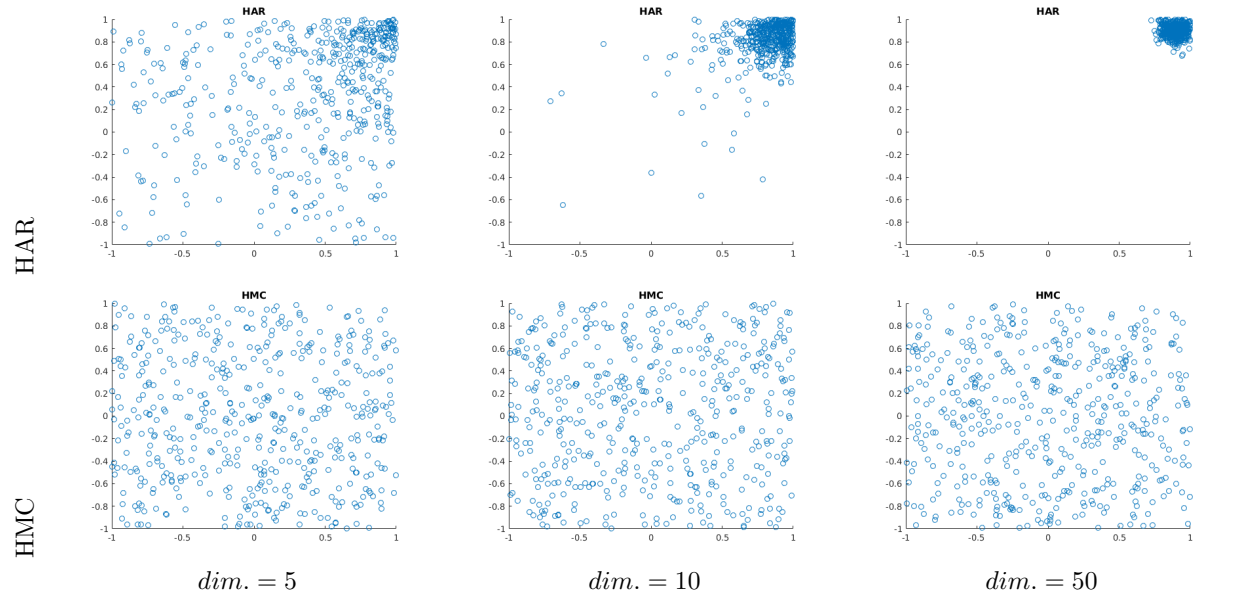
3.4.1 Setup

As described in section 3.3, we embed our random walk in the framework of [CV16]. We reuse the MATLAB code provided by [CV16] and make little modifications to call our HMC random walk instead of the usual HAR random walk.

3.4.2 Illustrations of the HMC random walk

It is well known that HAR struggles to get out of corners. Fig.3.3 shows that the HMC random walk do not suffer from the same drawback: starting in the corner of a cube yields an almost uniform distribution after 10 steps even when the dimension increases, while HAR performances suffers greatly.

Figure 3.3 Projections of the first two coordinates for starting from a corner ($q_i^{(0)} = 0.9$), after 10 steps of HAR or HMC, repeated 500 times for a nearly flat isotropic Gaussian distribution ($\sigma^2 = 500$) restricted to the cube $[-1, 1]^n$.



3.4.3 Analysis

Volume computation

The goal of the experiment is twofold: first compute the complexity (ie the number of call to 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 [CV16] introduced the stop criterion of Eq. (3.16). 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 [CV16] 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 [CV16] 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. To that end 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 oracle. For HAR, this is equal to the number of sampled points. For HMC, it takes the number of reflections into account.
- for HMC, the average number of calls to 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.

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 which is not feasible. Hence we settle for two experiments with one parameter fixed for each of them:

- (1) we fix the maximum travel time at a reasonable value (as we will see in (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.
- (2) we fix the stopping criterion (W) and vary the maximum travel time for several dimensions.

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 do detailed analysis. First, we choose the cube, as it has good theoretical properties while being a difficult case for the HAR. Then we choose the simplex, as it has sharp angles and is a widely used convex. 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.

Running times

Volume calculations presented thereafter were conduced 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.

3.4.4 Tests on volume calculations

Complexity analysis – stopping criterion

We ran the volume computation 50 times for each dimension using different values for W. From section.3.3.4, 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 if 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 ε, 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. 3.4) – a model for which HMC is well understood, and the isotropic simplex (Fig.3.5).

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 $W = 30 + 4n^2$. Similarly, for HMC the error decreases when W grows too fast. We select $W = 10$, $W = 30$ and $W = 10 + 4n^{0.5}$ as plausible values. The error was expected to decrease for $w = 30 + 4n^{0.5}$; however in our dimension range, the $4n^{0.5}$ part does not dominate the constant, hence we did not expect to see a clear decrease.

Overall, the experiments match our intuition for the cube, and do not significantly depart for the simplex. The complexity growth rates are summarized in Fig.3.6. These growth rates were obtained by a linear regression on the complexity curves (Figs. 3.4, 3.5, bottom plots). Remarkably, all correlation coefficients obtained were superior to 0.997.

Figure 3.4 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

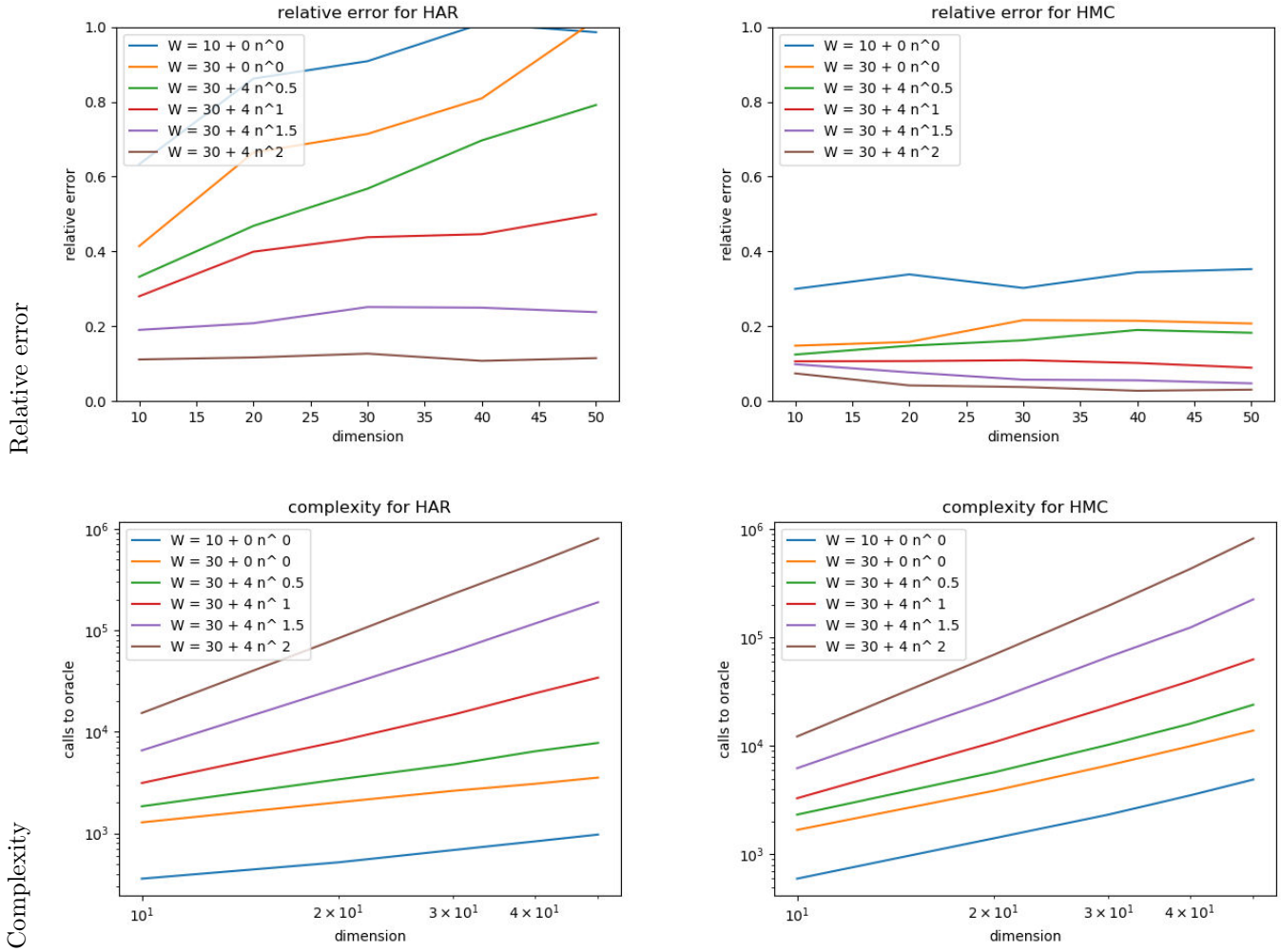


Figure 3.5 Iso simplex: relative errors (top) and complexities i.e. number of calls to the oracle (bottom) for volume computation – HAR (left) vs HMC (right) with the maximum travel time fixed at 1 for HMC.All quantities are averaged over 50 runs

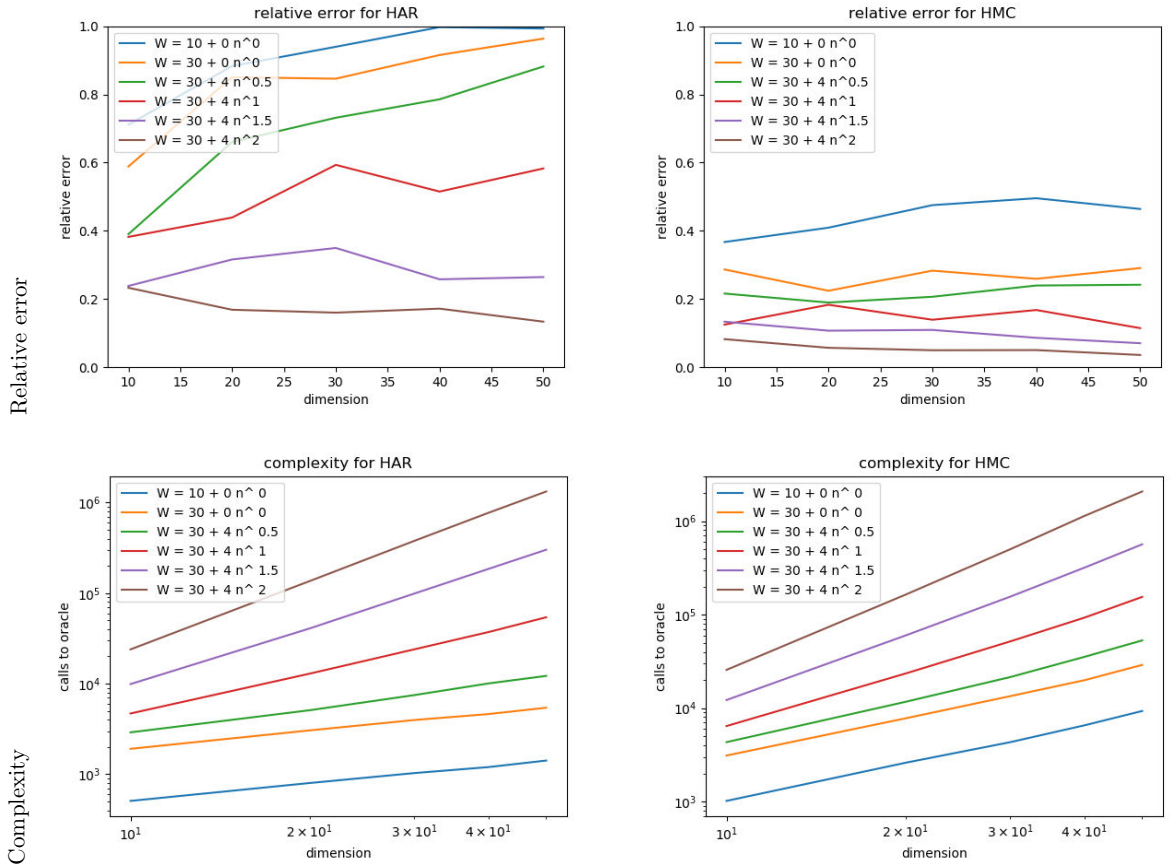


Figure 3.6 Scaling of the complexity with the dimension, computed using the same datas than Fig.3.4 and Fig.3.5. Plausible complexities are highlighted in yellow. Exponents for the complexity growth rates were obtained with a linear regression on the complexity curves of Figs. 3.4, 3.5 – see main text.

Window size	complexity			
	HMC		HAR	
	Cube	Iso Simplex	Cube	Iso Simplex
$W = 10$	$O(n^{1.29})$	$O(n^{1.36})$	$O(n^{0.62})$	$O(n^{0.62})$
$W = 30$	$O(n^{1.30})$	$O(n^{1.37})$	$O(n^{0.63})$	$O(n^{0.64})$
$W = 30 + 4n^{0.5}$	$O(n^{1.43})$	$O(n^{1.54})$	$O(n^{0.89})$	$O(n^{0.89})$
$W = 30 + 4n^{1}$	$O(n^{1.82})$	$O(n^{1.96})$	$O(n^{1.48})$	$O(n^{1.51})$
$W = 30 + 4n^{1.5}$	$O(n^{2.21})$	$O(n^{2.37})$	$O(n^{2.08})$	$O(n^{2.12})$
$W = 30 + 4n^{2}$	$O(n^{2.60})$	$O(n^{2.73})$	$O(n^{2.45})$	$O(n^{2.49})$

Travel time dependency for HMC

Based on the previous analysis, we choose $W = 30+4\sqrt{n}$ to study the standard simplex. Both in dimension 5 and 20, the errors decrease sharply with the travel time, then stabilize (Fig.3.7 and Fig.3.8 Top left). This can be seen in the number of points sampled by the algorithm which increases sharply before stabilizing(Fig.3.7 and Fig.3.8 Top right). Our interpretation is that when the travel time is too short, successive points are highly correlated, resulting in a false detection of convergence by the sliding window. Hence we get an high error and a low number of points. On the contrary, if the travel time is long enough, successive points are essentially decorrelated. As such, there is an optimal travel time between 0.5 and 1 in dimension 5 and between 1 and 1.5 for dimension 20. However, this optimum is neither the diameter, neither the radius of the inscribed sphere. At maximum travel time 1, there is an average of around 1.2 calls to oracle in dimension 5 and 2 in dimension 20.

Figure 3.7 Standard simplex in dimension 5, HMC: relative errors with deciles up to 80 percents of all points(top left), number of points sampled per run with std deviation(top right), number of calls to oracle per run with std deviation(bottom left), average number of calls to oracle per step with std deviation, 500 runs

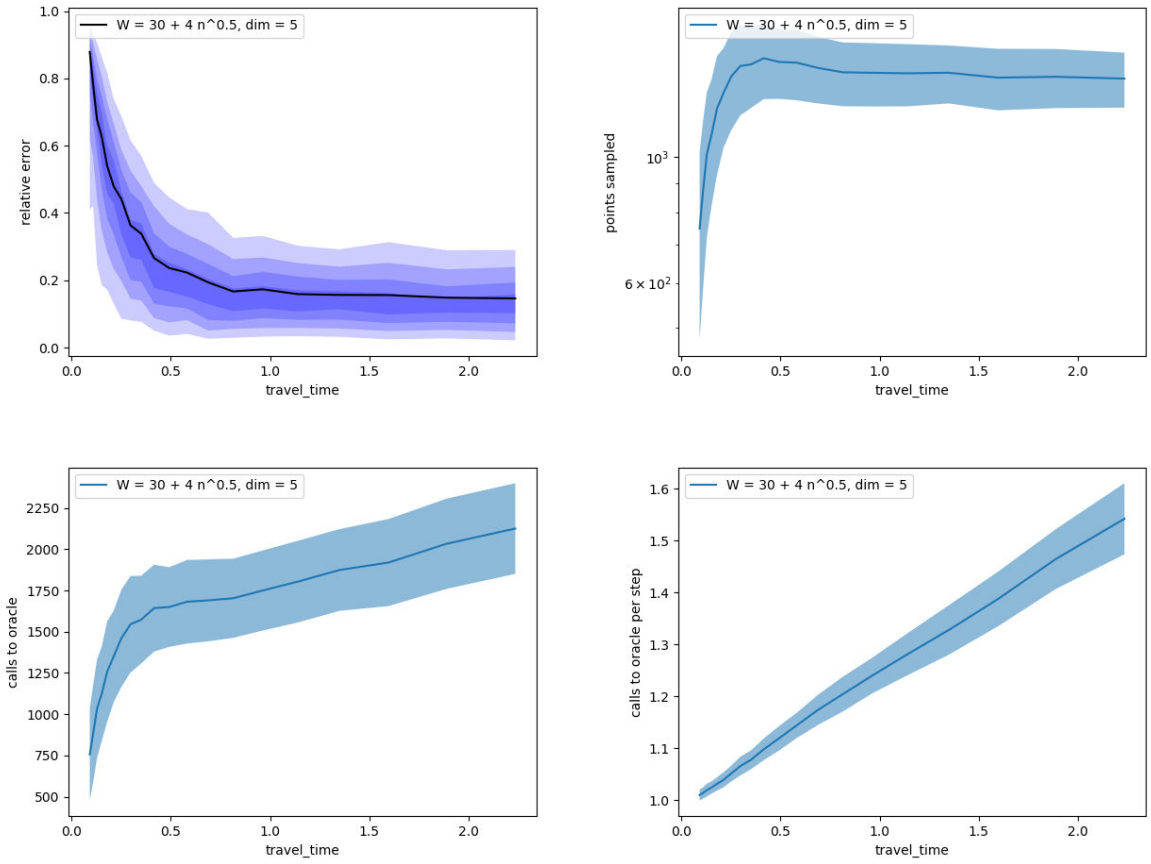
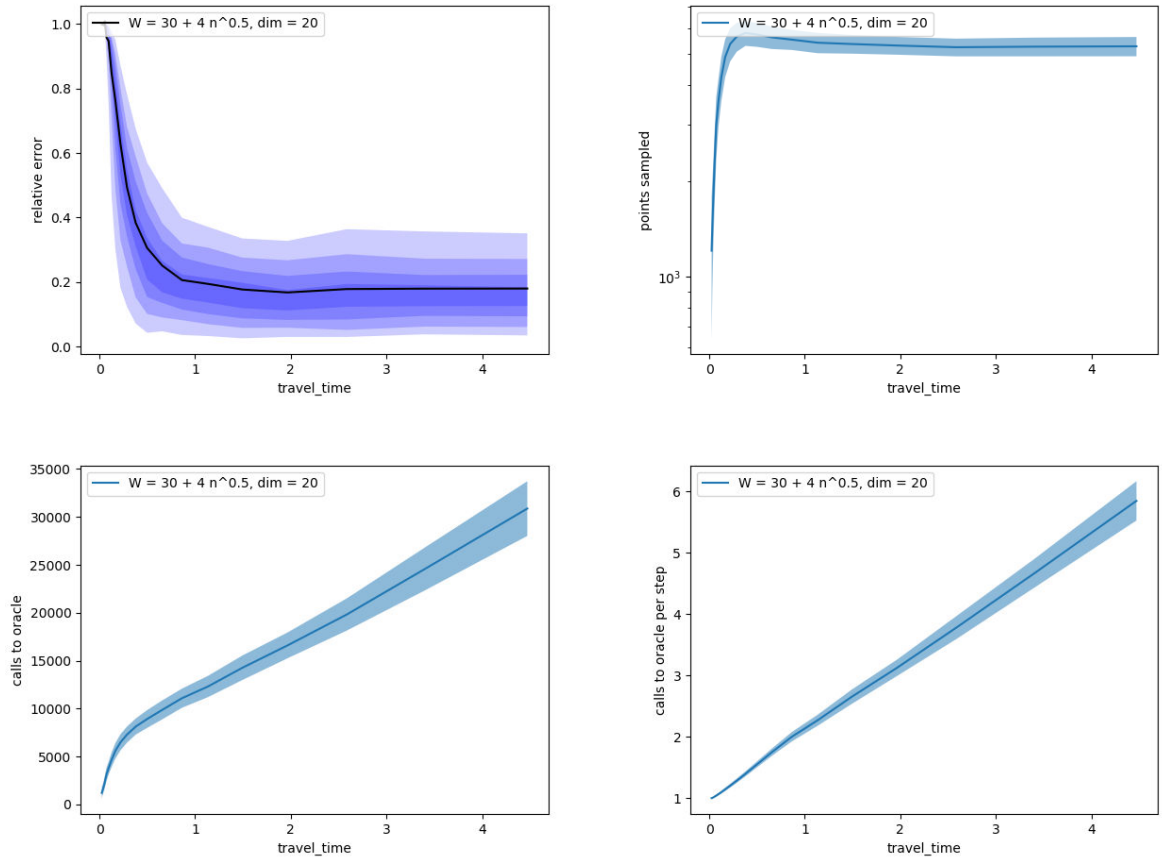


Figure 3.8 Same as Fig.3.7 in dimension 20.



3.5 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 analysis of the optimal travel time required by HMC to sample a target distribution, which should yield an automated choice for the travel time. 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.

3.6 Supporting information: pseudo-code

In the following, we provide the pseudo-code for the high level description of the HMC algorithm (Fig. 3.1).

Algorithm 2 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 3 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
```

3.7 Supporting information: results

To complement the analysis of section 3.4.4, we provide in the following plots with the variance of the statistics of interest (Figs. 3.9, 3.10, 3.11).

Algorithm 4 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 5 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 6 Reflecting the normal

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

Algorithm 7 Update trajectory with distance t

Update_positions_momenta(q, p, t)
for i from 1 to n **do**
  $[\omega, C, \Phi] =$ Compute_traj_params (q_i, p_i)
  Set $q_i = C\cos(\omega t + \phi)$
  Set $p_i = -\omega C \sin(\omega t + \phi)$
Return (q, p)

Figure 3.9 Error analysis: variance 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.

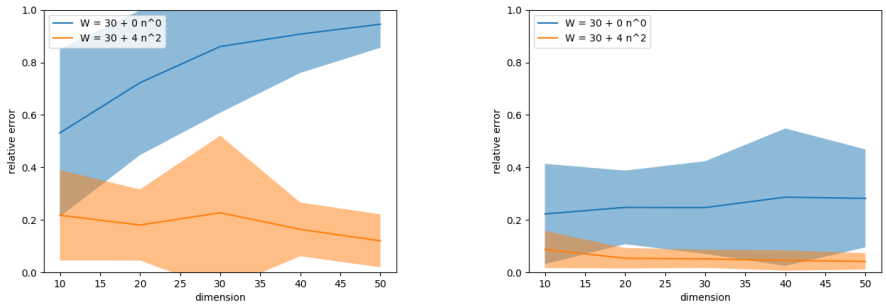


Figure 3.10 Number of generated points: variance. Model: isotropic simplex. (Left) HAR (Right) HMC The log scale hints at a polynomial number of points as a function of the dimension.

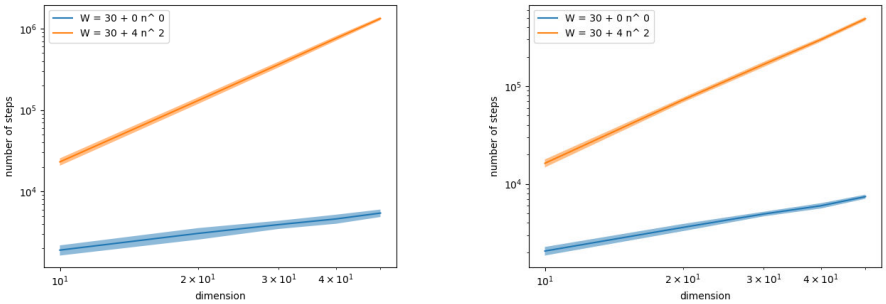
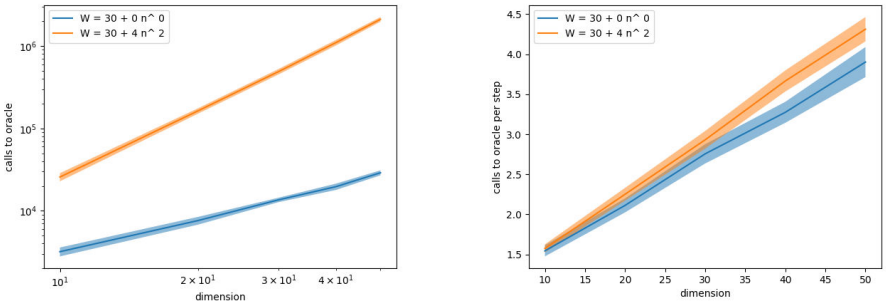


Figure 3.11 (Left) Number of oracle calls for HMC (Right) Ratio between the number of oracle calls and the number of points generated Plots for the isotropic simplex.



Chapter 4

A generic software framework for Wang-Landau type algorithms

4.1 Introduction

4.1.1 Sampling with the Wang-Landau algorithm

The Wang-Landau (WL) algorithm is a recently developed stochastic algorithm computing densities of states of a physical system. From a physics standpoint, WL computes the density of states associated to an energy. From a mathematical standpoint, WL estimates the push forward measure by an energy function of a density in state space, with a control on the relative error. For typical systems, the DoS spans several orders of magnitude and may be close to zero, whence the shear difficulty. WL is a stochastic approximation method falling in the class of adaptive Monte Carlo as it uses an evolving Markov kernel. At each step, WL estimates s the density of states of the system, which requires two main ingredients: the construction of a biased Markov kernel biased using the current estimation of the DoS – this requires a so-called *move set*; the update of the DoS – referred to as the *learning rule*.

This state of affairs prompted a variety of developments: since its inception, the Wang-Landau algorithm has spurred the development of many different variants, and numerous studies have been published about their efficiencies.

The learning rule was extensively studied, comparing the flat histogram criterion with deterministic rules and other proposed rules [SA11, WSTP15]. It was proposed to smooth or dynamically split the histogram [SNY11, BJMD13] or even replace it with a continuous representation [FLE19]. Other suggested merging the Wang-Landau algorithm with parallel tempering [RKIdP03, JH06]. Finally, a wealth of models were studied, from discrete models [LC10, SA11] to complex molecules such as met-enkephalin [JH06, RKIdP03, OMG10, SNY11, PCA$^+$06].

The physical systems studied triggered the development of various move sets. For example, in molecular studies, various move sets were proposed, based on molecular dynamics [RKIdP03, SNY11], on internal coordinates (dihedral angles) [OMG10, PCA$^+$06], or variants [MGCM12, BJMD13]. In a related vein, WL was also used to perform numerical integration. Multidimensional integral may be approximated by a discrete sum of function values multiplies by the measure of points achieving a given value [BMP08]. (Note that the function value plays the role of the density of states of a physical system.) Such calculations are of special interest to study convergence properties, since exact values (for the whole integral or the density of states) make it possible to scrutinize the convergence properties [AM17]. In this context, it was observed that bin width introduce another kind of saturation error, which call for a refined treatment of function values [AM17].

In most if not all the aforementioned studies, the reproducibility of results is jeopardized by the absence of software and/or the lack of implementation details.

4.1.2 Contributions

The previous review of previous work calls for a customizable implementation of WL, capable of accommodating a variety of systems, the variants of the algorithm, and the move sets. All this in a manner not sacrificing performances.

Systems. We isolate the pre-requisite namely the configuration space, the density, and the energy function within the so-called physical system. This accommodates discrete and continuous physical systems, as well as systems derived from numerical integration.

WL and variants. The Wang-Landau algorithm has many variants. The flat histogram rule originally proposed has been enhanced by the 1/t rule. However it has been suggested that the flat histogram criterion is not relevant, and other proposal have been suggested. In addition, several updates rule have been proposed as it was noted in [JR14]. We present here a generic framework which allows to implement all these different variants and combine them.

Move sets. As outline in [CC18b], the choice of the underlying random walk q and the associated mixing time for Wang-Landau is a core ingredients for fast convergence. Furthermore, in specific situation, like a discrete state space (e.g. the Ising model) or a state space which is a manifold, the creation of a custom move set which respect the constrains of these state space is mandatory. In this paper we present a generic move set framework allowing to easily create and combine customs adaptive move sets, with the ability to exploit and/or learn features from the configuration space, reducing the mixing time. In addition, we provide several generic move sets on $\mathbb{R}^n$ that uses geometric information to enhance mixing time (see chapter 2) . Those move sets are readily available for users and extensive experimental results are provided in chapter 2 , including for a small bio-molecule.

This work presents the first generic (C++) implementation providing these features. The versatility of the framework is illustrated with a variety of problems including the computation of density of states of physical systems and biomolecules, and the computation of high dimensional integrals.

The source code will be integrated to the SBL Structural Biology Library [CD17] and `http://sbl.inria.fr`.

4.2 Mathematiccal pre-requisites

4.2.1 Density of states and calculation by WL

First, it should be noted the measure on state space only requires to be defined up to a constant (in other words, it's a finite measure and not a probability measure). Therefore in this chapter, we consider a distribution with density $\pi(x)$ defined on a subset $\mathcal{E} \subset \mathbb{R}^D$. Also consider a partition of $\mathcal{E}$ into so-called strata $\{\mathcal{E}_1, \ldots, \mathcal{E}_d\}$. Denoting λ the Lebesgue measure, our problem is to estimate

$$\theta_i^* \stackrel{Def}{=} \frac{\pi(\mathcal{E}_i)}{\pi(\mathcal{E})} = \frac{1}{\pi(\mathcal{E})} \int_{\mathcal{E}_i} \pi(x)\lambda(dx). \tag{4.1}$$

This problem arises in many areas of science and engineering, two of them being of particular interest in the sequel.

At time t, the WL algorithm computes a sequence of estimates $\theta_i(t)$. Points sampled in a given stratum $\mathcal{E}_i$ by WL at time t follows the probability density

$$\frac{\pi(x)}{\pi(\mathcal{E}_i)}. \tag{4.2}$$

which do not depend on t.

4.2.2 Numerical integration

The Wang-Landau algorithm has been used for numerical integration [BMP08, LWLL07, AM16]. In the early works, the bin sizes leaded to a saturation of the error. To prevent the error saturation, estimations of the average energy were added in [AM16]. In practice, the method proposed by [AM16] to compute an integral $I = \int_{\mathcal{E}} U(x)\pi(x)dx$ relies on the following observation:

$$I = \sum_i \int_{\mathcal{E}_i} U(x)\pi(x)dx \tag{4.3}$$

$$= \pi(\mathcal{E}) \sum_i \frac{\pi(\mathcal{E}_i)}{\pi(\mathcal{E})} \int_{\mathcal{E}_i} U(x) \frac{\pi(x)dx}{\pi(\mathcal{E}_i)} \tag{4.4}$$

$$= \pi(\mathcal{E}) \sum_i \theta_i^* \int_{\mathcal{E}_i} U(x) \frac{\pi(x)dx}{\pi(\mathcal{E}_i)}. \tag{4.5}$$

Following the points distribution in each stratum given by Eq. (4.2), the average value in a bin is defined by the integral

$$\int_{\mathcal{E}_i} U(x) \frac{\pi(x)dx}{\pi(\mathcal{E}_i)}. \tag{4.6}$$

This integral can be computed during WL runtime by computing the average energy value in a given bin. Therefore if the total volume $\pi(\mathcal{E})$ is known, the integral I can be computed with WL.

It should be noted that our formulation slightly differs from [AM16] because we consider the general case of a finite measure π on $\mathcal{E}$ while they only consider the Lebesgue measure.

4.2.3 Incidence of the choice of the measure

For two measure π and μ, we define:

$$\theta_i^{\pi*} = \frac{\int_{\mathcal{E}_i} \pi(x)dx}{\int_{\mathcal{E}} \pi(x)dx} = \frac{\pi(\mathcal{E}_i)}{\pi(\mathcal{E})}$$

$$\theta_i^{\mu*} = \frac{\int_{\mathcal{E}_i} \mu(x)dx}{\int_{\mathcal{E}} \mu(x)dx} = \frac{\mu(\mathcal{E}_i)}{\mu(\mathcal{E})}$$

The choice between π and μ has two direct implications. First, $\theta^{\pi*}$ is different than $\theta^{\mu*}$. Second, as seen from Eq. (4.2), the density sampled by WL in a given stratum are different.

In certain cases changing the measure on the state can improve the convergence speed. Consider the potential energy $U(x) = \|x\|^2$. In high dimension, because of the concentration of measure, most of the points of each stratum will be on the outer boundary with the Lebesgue measure, making it difficult to go from one stratum to a lower one. If the measure π is changed to counteract this effect and bias toward the interior boundary, it might improve the mixing time of the random walk. On the other hand we will see that for discrete systems, or when the bin size goes to 0, the choice of measure do not matter.

Density of state: change of measure Therefore we describe here a method to deduce the density of state for a measure μ while the Wang-Landau algorithm was used with a measure π on state space. To get from one measure to another, we will need the numerical integration capabilities of the software.

For each $\mathcal{E}_i$, WL for the measure π samples points with respect to the density of Eq. (4.2) in $\mathcal{E}_i$. Therefore we can compute the following integral for each stratum during the run-time of the Wang-Landau algorithm:

$$I_{\mathcal{E}_i} = \int_{\mathcal{E}_i} \frac{\mu(x)}{\pi(x)} \frac{\pi(x)}{\int_{\mathcal{E}_i} \pi(y)dy} dx = \frac{\mu(\mathcal{E}_i)}{\pi(\mathcal{E}_i)} \tag{4.7}$$

We rewrite $\theta^*_\pi(i)$:

$$\begin{aligned}\theta^*_\pi(i) &= \frac{\mu(\mathcal{E}_i)}{I_{\mathcal{E}_i}} \frac{1}{\pi(\mathcal{E})} \\ &= \frac{\mu(\mathcal{E}_i)}{I_{\mathcal{E}_i}} \frac{1}{\sum_j \pi(\mathcal{E}_j)} \\ &= \frac{\mu(\mathcal{E}_i)}{I_{\mathcal{E}_i}} \frac{1}{\sum_j \frac{\mu(\mathcal{E}_j)}{I_{\mathcal{E}_j}}}\end{aligned}$$

Hence:

$$\theta_i^{\pi*} \sum_j \frac{\mu(\mathcal{E}_j)}{I_{\mathcal{E}_j}} = \frac{\mu(\mathcal{E}_i)}{I_{\mathcal{E}_i}} \qquad \forall i \tag{4.8}$$

and diving by $\mu(\mathcal{E})$,

$$\theta_i^{\pi*} \sum_j \frac{\theta_j^{\mu*}}{I_{\mathcal{E}_j}} = \frac{\theta_i^{\mu*}}{I_{\mathcal{E}_i}} \qquad \forall i \tag{4.9}$$

Finally, using that $\theta^*_\mu(i) = 1 - \sum_{j \neq i} \theta_j^{\mu*}$ we deduce:

$$\theta_i^{\pi*} \frac{\theta_i^{\mu*}}{I_{\mathcal{E}_i}} + \sum_{j \neq i} \theta^*_\mu(j) \left(\frac{\theta_i^{\pi*}}{I_{\mathcal{E}_j}} + \frac{1}{I_{\mathcal{E}_i}} \right) = \frac{1}{I_{\mathcal{E}_i}} \qquad \forall i. \tag{4.10}$$

Since $\theta_i^{\pi*}$ and $I_{\mathcal{E}_i}$ are estimated during runtime, the system of equation given in eq 4.10 yields a linear system for the unknowns $\theta_i^{\mu*}$, which can therefore be estimated.

Remark 4.1. *the custom Move sets will not take the geometry of π into account.*

Discrete energy space. We assume a discrete energy space $\{U_1, ..., U_d\}$. The state space $\mathcal{E}$ is not necessarily discrete, but we assume that a stratum covers exactly one energy level. We analyse the special case where π and μ can be written as $\pi(x) = f_\pi(U(x))$ and $\mu(x) = f_\mu(U(x))$. This specific setting includes the Boltzmann distribution which is of crucial importance for statistical physics.

Theorem. 4.1. *For any $\theta_i^\mu(t_0)$, if $\theta_i^\pi(t_0)$ verifies*

$$\theta_i^\pi(t_0) = \frac{f_\pi(U_i)}{f_\mu(U_i)} \theta_i^\mu(t_0) \tag{4.11}$$

then the Wang-Landau algorithm will generate exactly the same points with densities μ and π on state space and for all $t \geq t_0$,

$$\theta_i^\pi(t) = \frac{f_\pi(U_i)}{f_\mu(U_i)} \theta_i^\mu(t) \tag{4.12}$$

Proof. We have:

$$\begin{aligned}\theta_i^{\pi*} &= \frac{\int_{\mathcal{E}_i} \pi(x)dx}{\int_{\mathcal{E}} \pi(x)dx} = \frac{f_\pi(U_i) Vol(\mathcal{E}_i)}{\sum_j f_\pi(U_i) Vol(\mathcal{E}_j)} \\ \theta_i^{\mu*} &= \frac{\int_{\mathcal{E}_i} \mu(x)dx}{\int_{\mathcal{E}} \mu(x)dx} = \frac{f_\mu(U_i) Vol(\mathcal{E}_i)}{\sum_j f_\mu(U_i) Vol(\mathcal{E}_j)}\end{aligned}$$

Let $\pi_{\theta^\pi(t_0)}$ the biased density sampled by WL at time t_0 when using π as the measure on state space.

$$\begin{aligned}
\pi_{\theta^\pi(t_0)}(x) &= \left(\sum_{i=1}^{d} \frac{\theta_i^{\pi *}}{\theta_i^\pi(t_0)}\right)^{-1} \frac{\pi(x)}{\theta^\pi_{J(x)}(t_0)} \\
&= \left(\sum_{i=1}^{d} \frac{\theta_i^{\pi *}}{\theta_i^\pi(t_0)}\right)^{-1} \frac{f_\pi(U_{J(x)})}{\frac{f_\pi(U_{J(x)})}{f_\mu(U_{J(x)})}\theta^\mu_{J(x)}(t_0)} \\
&= \left(\sum_{i=1}^{d} \frac{\theta_i^{\pi *}}{\theta_i^\pi(t_0)}\right)^{-1} \frac{f_\mu(U_{J(x)})}{\theta^\mu_{J(x)}(t_0)} \\
&= \left(\sum_{i=1}^{d} \frac{\theta_i^{\mu *}}{\theta_i^\mu(t_0)}\right)^{-1} \frac{f_\mu(U_{J(x)})}{\theta^\mu_{J(x)}(t_0)}
\end{aligned}$$

Where the last line is trivial when considering $\left(\sum_{i=1}^{d} \frac{\theta_i^{\mu *}}{\theta_i^\mu(t_0)}\right)^{-1}$ as the renormalisation factor. Finally we deduce

$$\pi_{\theta^\pi(t_0)}(x) = \left(\sum_{i=1}^{d} \frac{\theta_i^{\mu *}}{\theta_i^\mu(t_0)}\right)^{-1} \frac{\mu(x)}{\theta^\mu_{J(x)}(t_0)} \tag{4.13}$$

Which is the biased density that would be used by WL at time t_0 when using μ as the measure on state space. Let x_{t_0} be the sampled point. Then the updated estimations using γ as the correction factor are:

$$\begin{aligned}
\theta^\pi_{J(x_{t_0})}(t_0+1) &= \theta^\pi_{J(x_{t_0})}(t_0) * \gamma \\
\theta^\mu_{J(x_{t_0})}(t_0+1) &= \theta^\mu_{J(x_{t_0})}(t_0) * \gamma \\
\theta^\pi_i(t_0+1) &= \theta^\pi_i(t_0) \qquad \forall i \neq J(x_{t_0}) \\
\theta^\mu_i(t_0+1) &= \theta^\mu_i(t_0) \qquad \forall i \neq J(x_{t_0})
\end{aligned}$$

which still respects eq.4.12. The final result is obtained by recurrence. □

We conclude that in this setting, the choice of measure on the state space do not matter.

4.2.4 Boundary condition

In order to restrict the state space to a subset A, two ways are possible. The first is to design a custom move set q such that for every x in A, the probability $q(x, A^c)$ of going out of A is 0. The second is to reject samples by evaluating the characteristic function 1_A of A. For complex boundary conditions, the first method is usually impossible, and the second can be computationally expensive. However in some cases it is possible to write $1_A = 1_B 1_C$ with 1_B being cheap to evaluate and 1_C expensive, see Example 1. Assume most of the boundary is determined by B, in the sense that the probability that P_θ leaves C without leaving B is small compared to the probability of leaving A. A simple strategy is to only evaluate 1_B at every step and evaluate 1_C every k steps, with k of the order of magnitude of the complexity of 1_C. Then if $x_{k(t+1)}$ is not in C, the algorithms rolls back to x_{kt}, and rerun the random walk while checking 1_C for each point to find the exit point.

This method main flow is that it introduces a bias in the solution: the random walk is allowed to leave C for $k-1$ point as long as it comes back to A for the k-th point. However, under the previously stated assumption, this bias is small.

Example 1. *A common use case is a filter ensuring that a given conformation remains in a prescribed basin below a potential energy threshold C. Checking the former condition requires quenching i.e. minimizing U*

– *a costly operation. We therefore use the indicator functions* $1_A(x) = 1_B(x)1_C(x)$*, with* $1_B(x) = 1$ *iff* $U(x) < C$*, and* $1_C(x) = 1$ *iff* x *is in the prescribed basin.*

Remark 4.2. *This method requires to go back in time and reproduce the same steps of the random to find the exit point. Hence it requires to save the pseudo-random number generators states in addition of the full state of the algorithm every* k *steps.*

4.3 Code design

4.3.1 Overview

The programm is divided into 4 major components (Fig. 4.1):

- (Must be provided) The *physical* system provided by the user which describes the state space and the energy function. And possibly the energy gradient.
- (Can be provided) The *move set* defines the random walk and the associated move probability.
- (Can be provided) The *WL data structure* storing bin ranges. Various pieces of information on a per bin basis:
  - The estimation θ, and can be extended to store any statistics.
  - Custom information including acceptance rate, etc.

  To perform specific processing, like numerical integration, the user must defined a custom WL DS, by inheritance from the default (provided) one.
- (Provided by the library) The *Wang Landau* class itself which is responsible for gluing the two other components together, run the algorithm and feeding the *move set* every information it needs. In our case, the bin range of the current and neighbouring energy bin, but it can be anything the user needs it to be.

Figure 4.1 Using the WL framework: the different software components. Blue: library provided; Green: default provided, can be replaced; Yellow: defined by user.

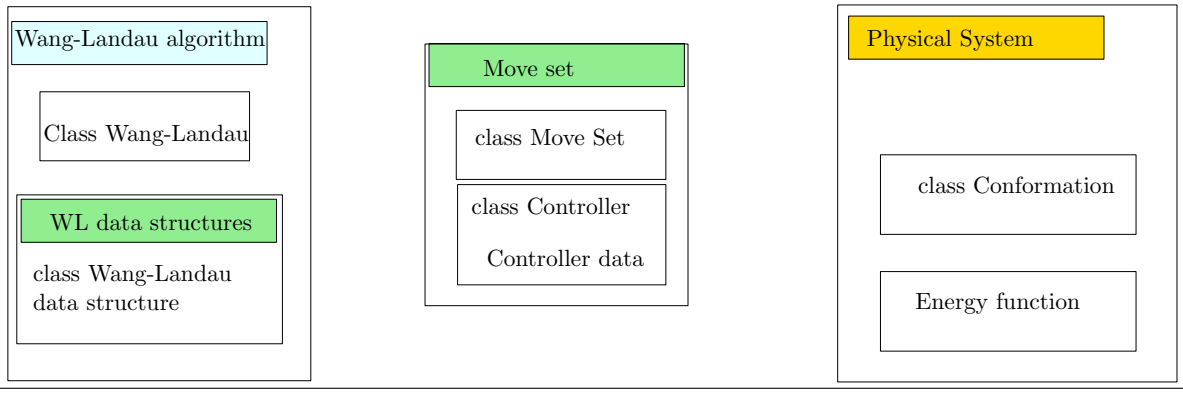
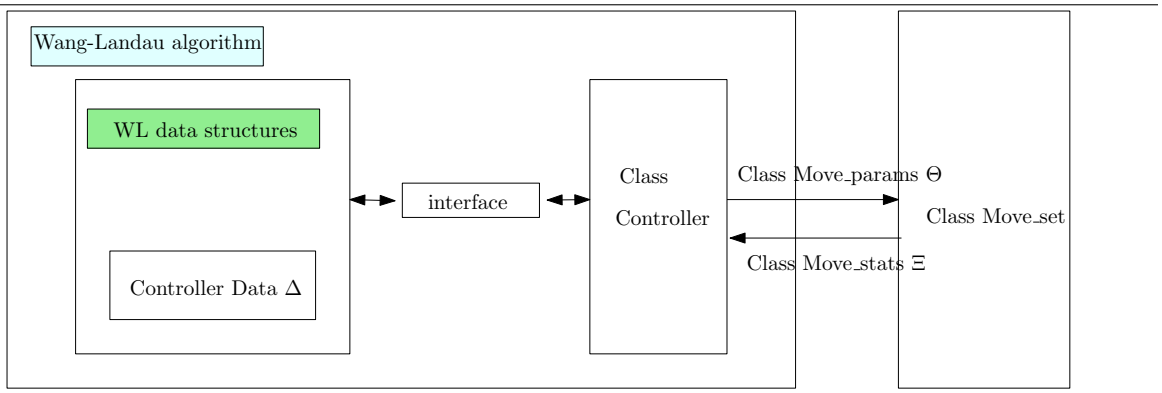


Figure 4.2 Move set and calling algorithm: software design. Color codes as in Fig. 4.1, that is: Blue: library provided; Green: default provided, can be replaced; Yellow: defined by user.



4.3.2 Physical system

The physical system role is to defining the state space and energy function.

Conformation

By *conformation*, we refer to the C++ class storing the point in state space and the corresponding energy $U(x)$. For move sets requiring the gradient, the conformation also stores $\nabla U(x)$.

In practice, for continuous systems, conformations are stored as a vector of doubles. For systems whose state space is discrete, conformations are typically stored as an integer.

Physical System

A physical system is a C++ class which defines the conformation type, an energy function, the boundary condition and if required, the density π on state space (if not specified, $\pi(x) = 1$) –see section 4.2.3.

4.3.3 Move sets in the context of a calling algorithm

Move set

Move sets play a pivotal role in various stochastic algorithms, including (energy) landscape exploration algorithms in the lineage of basin hopping, and density of state calculations in the lineage of Wang-Landau. We therefore outline a general move set design not restricted to the Wang-Landau algorithm, while still indicating specificities for WL.

Strictly speaking, a **Move Set** (MS) provides 2 operations:

- generates a conformation y given a conformation x and parameters Θ (such as the variance for a Gaussian move set) that we call **Move Parameter**
- also provides the probability $p(x, y, \Theta)$ to move from x to y, used by algorithm to satisfy details balance if required. While detailed balance is not required for every single algorithm, it is so for Wang-Landau in order to guarantee correctness of the sampling.

Furthermore, after a conformation generation, the move set return informations Ξ that we call **Move Stats** that can be used later for adaptivity.

Provided move sets

When the conformational space is $\mathbb{R}^n$, the following move sets are provided:

- Isotropic Gaussian move set.
- The cone-based move set defined in [CC18b],
- The darting move set also defined in [CC18b].

Combined move set

Classical move sets includes the ball walk, Gaussian moves, etc. A mixture a move sets can also be used to accommodate different features of –see chapter 2.

A *combined move sets* is composed of a collection of $k > 0$ move sets $\{MS_1, ..., MS_k\}$, and a function

$$\begin{aligned} w : \mathcal{E} &\longrightarrow \mathbb{R}^k \\ x &\longrightarrow (w_1(x), ..., w_k(x)) \end{aligned}$$

such that for all $x \in \mathcal{E}$, $\sum_l w_l(x) = 1$. For a given $x \in \mathcal{E}$, $w_l(x)$ represents the probability of choosing move set MS_l to generate the next point. The parameter for such a move set is a vector Θ with each coordinates Θ_l being the parameter for move set MS_l. A combined move set transition kernel is given by

$$p(x, y, \Theta) = \sum_{l=1}^{k} w_l(x) p_l(x, y, \Theta_l) \tag{4.14}$$

with p_l the transition kernel of move set MS_l.

Such move sets, called *combined move sets*, are implemented as a generic C++ class.

Remark 4.3. *Practically, a combined move set is represented as a tree. Internal nodes are the combined move sets, and samples are generated by move sets found within leaves.*

Adaptivity via controller

The choice of parameter Θ is central for rapid convergence of stochastic algorithms. This choice depends on the geometric features of the landscape, but also the goal of the calling stochastic algorithm.

In the following, we introduce a software component called **Controller**, which adapts the move set to the landscape as a function of the objectives of the calling algorithm (Fig. 4.2) by generating the parameter Θ for the move set.

The **Controller** is typically embedded a in an algorithm specific context, hence generating an optimal Θ imposes interactions with other data structures. Moreover, the **Controller** may learn features of the landscape. Such controllers interact with the underlying algorithm to store relevant data (**Move_Set_Controller_Data**, noted Δ). To that end, the underlying algorithms provides the controller a **Data_Interface** (algorithm dependent). This overall design is summed up inFig 4.2.

Remark 4.4. *In the case of Wang-Landau, the class* ***Move_Set_Controller_Data*** *is instantiated once per bin, therefore, the controller can store data Δ_i on a per bin basis. The move parameters Θ generation by the controller defines a mapping from the vector Δ to Θ, see Eq. (4.14).*

Adaptivity for combined move sets. For the *combined move set*, the controller will be automatically generated. The data stored (**Move_Set_Controller_Data**) is the reunion of the data used by each controller, and the controllers of the move sets composing the combined move set are automatically called when relevant. In the WL case, it means that each **Move_Set_Controller_Data** is instantiated once per bin for all the move set composing the *combined move set*.

4.3.4 WL data structure

The Wang-Landau data structure responsibilities are to handle the bin data structure including the estimation of the density θ, update the densities when a new point is sampled, record useful statistics for move stats, and in general store any runtime statistics that the user needs to record. We will not describe the Wang-Landau Data Structure in it's entirety, but only it's main functions and customization points. The implementation is on the class **T_WL_Data_Structure_MS**, which should be inherited from for customization.

Bins data structure

Default. The WL data structure handle bin creation and splitting. The user can specify the behaviour of the data structure when a new bin is required by giving a default bin size and whether the energy levels are continuous or not (e.g., the Ising model has discrete energies). We leave out the implementation details of this behaviour. A type **Bin_properties_MS** defines the default properties to store for each bin: $\theta(i), \nu(i)$ and an instance of **Move_Set_Controller_Data** Δ_i.

Customization. To store additional statistics, one must define a class **Custom_Bin_Properties** that derives from **Bin_properties_MS** and pass this type as a template parameter to **T_WL_Data_Structure_MS**.

Update after point generation

Default. The WL data structure updates relevant information stored in the data structure each time a point is generated by the algorithm. To that end, the WL data structure performs the following operations:

- check the energy of a generated conformation can be accommodated, and creates a new bin if not.
- update the estimated density i.e. θ
- check the flat histogram criterion and update the learning rate γ
- reset the learning rate γ upon addition of a new bin

Customization. This default behaviour can be customize to perform the following:

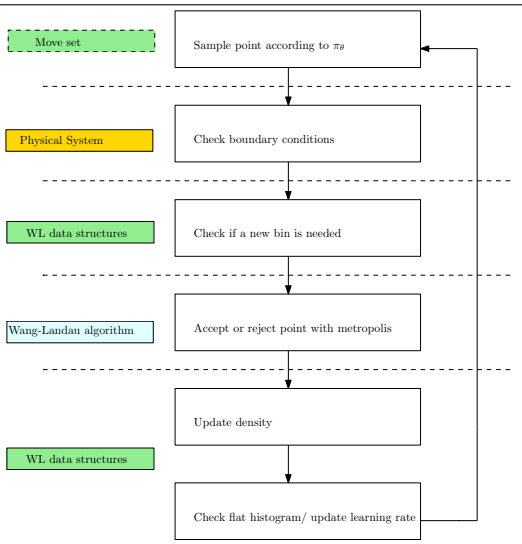
- use a different learning strategy, possibly introducing variants of the flat histogram rule
- record additional statistic. For example, the average energy value per bin (Eq. 4.6) can be recorded to perform numerical integration, see section 4.2.2.

Practically, a new class inheriting from **T_WL_Data_Structure_MS** must be provided.

4.3.5 Main algorithm

The main class implements the WL algorithm using the previous ingredients. The corresponding C++ template class, **T_Wang_Landau**, requires the physical system, the move set (and its controller), the WL DS as template parameters. The execution flow is represented on Fig. 4.3.

Figure 4.3 Flow of the algorithm as it is controlled by the T_Wang_Landau class. Color codes as in Fig. 4.1.



4.3.6 Output

Overview. In the output, we distinguish two types of data:

- Exploitation data: data providing the result of the calculation.
- Monitoring data: pieces of information providing information on the decisions made by the algorithm.

These data are reported as serialized files.

We also introduce two reporting modes:

- constant lag mode: data are dumped every T steps, with T a user defined parameter.
- log lag mode: the lag time between two dumps increases exponentially. (Useful to perform log plots.)

Exploitation data. One output file containing, for each recorded step, the histogram θ.

Monitoring data.

- One output file containing for each recorded step the value of ν_i per bin. Note that the length of this vector is exactly the number of bins, which may vary along the simulation.
- One output file containing for each recorded step the information Δ_i per bin. Following Rmk. 4.3, Δ_i has a tree structure; this tree is dumped at every recorded step.

4.4 Experiments

4.4.1 Setup

Software setup. The software design presented was developed within the Structural Bioinformatics Library (`http://sbl.inria.fr`), and will be made available upon publication.

All experiments were conducted as follows. The energy range is discovered at run time, with new bins created when required. The $1/t$ WL algorithm is used. Move sets parameters are discussed on a per example basis.

Tests. We report three types of experiments. To show the ability of our setup to handle both discrete and continuous physical systems, we report results for the Ising model and a small biomolecules (alanine dipeptide). To show the versatility and the efficiency of the implementation, we present numerical integration results, solving one case left open in [AM17]. Finally, we provide insights on an issue typically overlooked, namely the choice of the measure used in the calculation/integral. We show that improved convergence speed can be obtained upon integrating against a suitable measure (Boltzmann versus Lebesgue).

4.4.2 Handling discrete and continuous systems

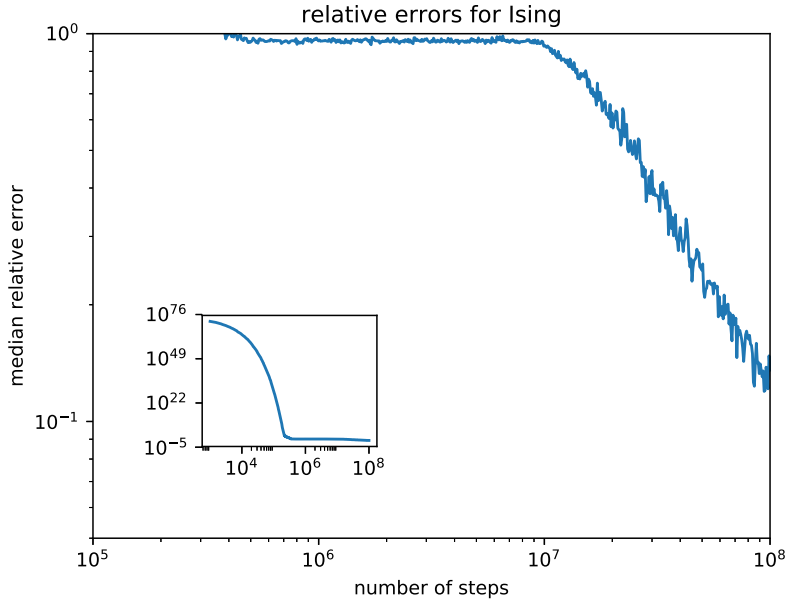
In the following, we illustrate the ability of our framework to handle discrete and continuous systems, on two classical examples.

Ising model

Ising models are simple yet challenging systems since the number of states is exponential in the size. Such models were first used to illustrate the strength of the WL algorithm [WL01]. In the sequel, we consider the 16×16 Ising model, each vertex on the grid having the $+1$ or -1 label. The total number of states is $2^{16\times 16} = 0.115792089210^{78}$. The energy of a given conformation is given by the sum over each vertex of the product of the labels with the neighbors. The move set flips 1 spin at random in the grid. An analytical formula of the number of configuration for each energy is available [], which we use to compute the relative error.

The relative error of the estimation computed by the WL algorithm first decreases exponentially, then with the $\frac{1}{\sqrt{t}}$ rate(Fig. 4.4).

Figure 4.4 Relative error for 16x16 Ising model. Median taken over 40 runs.



Alanine dipeptide

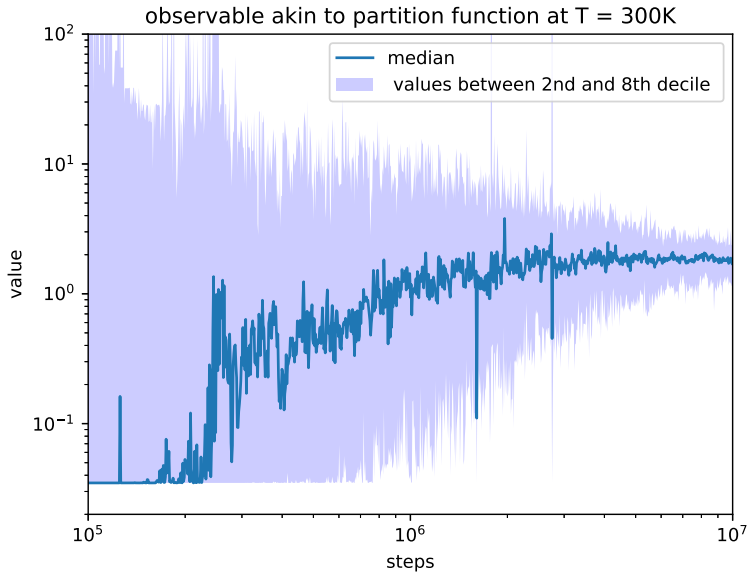
Dialanine is a small (bio)-system with 22 atoms that is commonly used for benchmarking algorithms. The amber99-sb force field in vacuum is used to compute the density of state between -21 kcal/mol and 4 kcal/-mol, associated to a single local minima of the potential energy function (torsion angles: $(\phi = 59.8862, \psi =$

-35.5193)). As described in section 4.2.4, the boundary condition for a local minim requires energy minimisation, which is too expensive to execute every step. Therefore, we use the method described in section 4.2.4 and minimize the energy only every 100 steps. We recall the observable akin to the partition function (Eq. 2.17 that we study to show convergence:

$$\frac{Z}{\lambda(\mathcal{E})} = \frac{1}{\lambda(\mathcal{E})} \int_{\mathcal{E}} \exp\left(-U(x)/kT\right) \approx \sum_{\text{Energy levels U}} \theta_i^* \exp\left(-U/kT\right). \tag{4.15}$$

We performed 60 runs, each with 10^7 steps, and plotted the median of the estimates, as well as the 2nd and 8th deciles (Fig. 4.5).

Figure 4.5 median and 1rst to 9nth decile of observable akin to the partition function (Eq. 4.15) for Dialanine, data obtained for 60 runs.



4.4.3 Numerical integration

In cases where direct Monte-Carlo integration is impractical, in particular due to concentration of the measure, numerical integration schemes based on Wang-Landau have been proposed [LWLL07, AM17].

In the sequel, we study a simple 2D example–as a sanity check, and proceed with one one ill-behaved Gaussian integral.

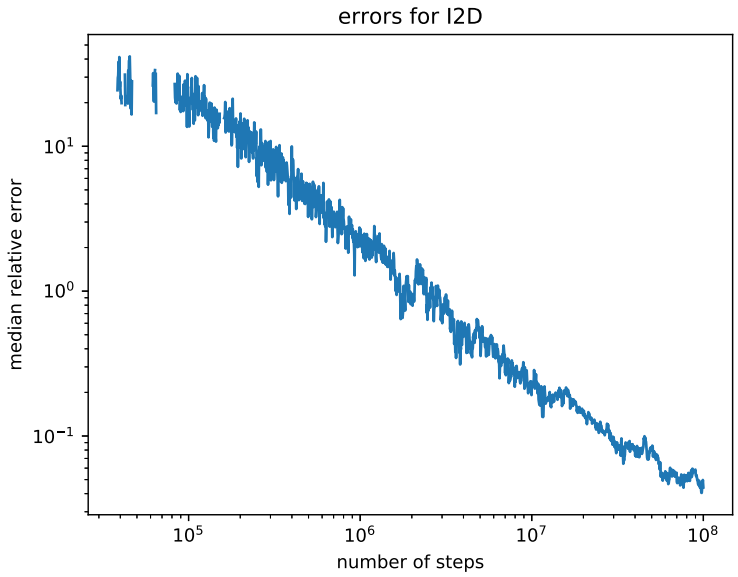
2D integral

We start with the following test integral from [BMP08, LWLL07, AM16]:

$$I_{2D} = \int_{-1}^{1} \int_{-1}^{1} (x_1^6 - x_1 x_2^3 + x_1^2 x_2 + 2x_1) \sin(4x_1 + 1) \cos(4x_2) dx_1 dx_2 \tag{4.16}$$

As expected, the decay rate is $1/\sqrt{t}$ (Fig. 4.6).

Figure 4.6 median relative error for I_{2D} (Eq. 4.16), data obtained for 40 runs.



Integration of Gaussian density

More interesting is the following integral, studied up to dimension 6 in [AM17]:

$$I_n = \int f_n(x)dx \text{ with } f_n(x) = \frac{1}{(2\pi\sigma^2)^{n/2}} e^{-\frac{\|x\|^2}{2\sigma^2}}. \tag{4.17}$$

In the sequel, we use $\sigma^2 = 0.4$. In [AM17], they make the choice of restricting the integration domain to $[-10, 10]^n$ to get most of the mass. However this choice adds artificial difficulties for random walks, as they tend to get stuck in corners of the cube. Hence we change the integration domain to the ball $B(0, 10)$ of center 0 and radius 10. In this setting, I_n is very close to 1 for the dimension range we are interested in.

While this integral is studied up to dimension 6 in [AM17]. For dimensions 5 and 6 though, the algorithm does not converge, leading the authors to introduce the so-called *two state* strategy which amounts to splitting the integration domain by the median of function values.

In the sequel, we present an effective strategy up to dimension 15 (Fig. 4.7). When the dimension increases, the volume of the outermost strata increases exponentially; also, since the gradient of the function gets small, so are the components of the gradient, preventing the cone-based move set to act as an effective guide to diffuse across WL bins. Overall, the random walk has difficulties to escape the stratum, jeopardizing convergence.

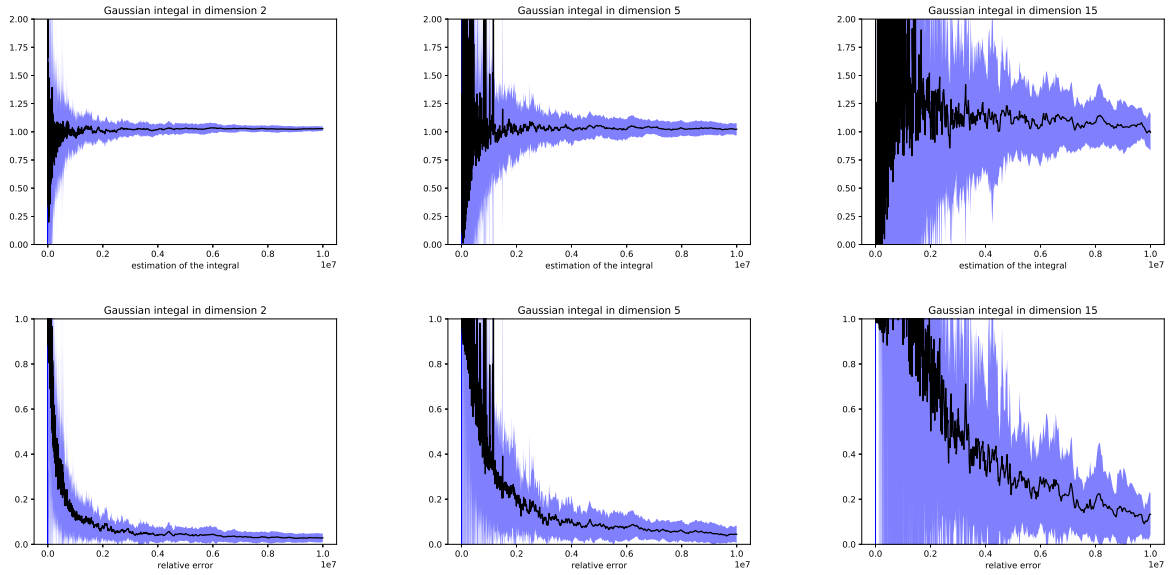
To enhance the ability to deal with high dimensions, we introduce a general strategy by taking as energy the logarithm of our integration function:

$$E_n(x) = \log(f_n(x)). \tag{4.18}$$

This boosts the diffusive behaviour of the random walk. To get the correct value for the integral, instead of taking the average in each strata, we take the average of the exponential:

$$I_n(x) = \sum_i V(\mathcal{E}_i) \int_{\mathcal{E}_i} \frac{e^{E_n(x)}}{V(\mathcal{E}_i)} dx. \tag{4.19}$$

Figure 4.7 Value (Top row) and relative error (Bottom row) for the Gaussian integral (Eq. 4.17) in dimensions 2, 5 and 15, respectively. A total of 40 runs were performed. The black plot display the median; the purple plot display the 1st and 9th decile.



4.4.4 Change of measure

In section 4.2.3, we have shown how to obtain results for a measure μ on state space while sampling from the base measure π. Practically, this raises the question of choosing the measure to integrate against. we illustrate this by comparing integration results obtained respectively using Lebesgue measure and Boltzmann measure. In both cases, the random walk used relies on Gaussian moves. We make this choice since cone based random walks (see section 4.3.3) attenuate concentration effects, making the experiment irrelevant.

Consider the following potential $U(x) = \|x\|^2$, and an energy discretisation of $[0, 1]$. The aim is to compute the volume of each stratum with respect to the Lebesgue measure (we make this choice because the exact volume is trivial to compute in the case of the Lebesgue measure). When the dimension increases, inside each stratum the volume is concentrated near the outer ring. Hence points sampled by the Wang-Landau algorithm with respect to the Lebesgue measure would be concentrated near the outer ring of the strata. This behaviour hinders the convergence speed of Wang-Landau, and a solution consists in splitting the bins if the volume near the outer ring is too imbalanced compared to the interior ring [BJMD13].

We consider a different approach and introduce a Boltzmann-like density

$$\pi_T(x) = e^{-U(x)/T}.$$

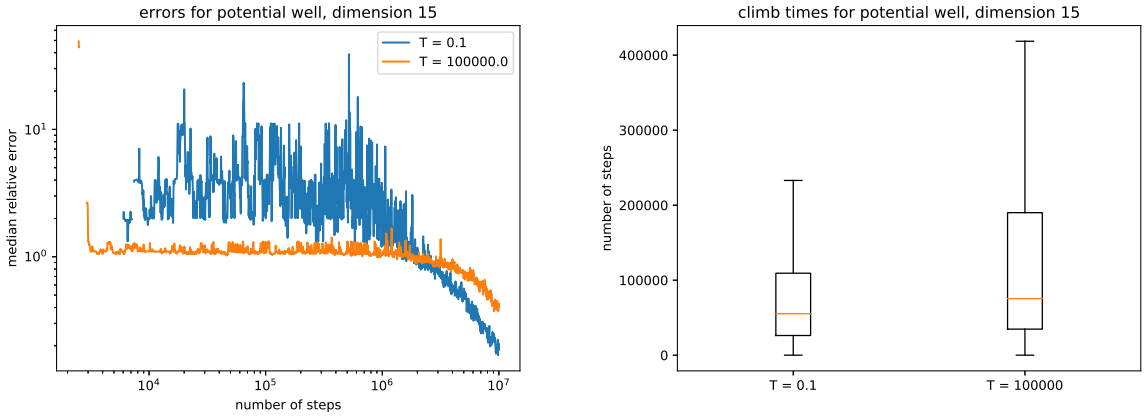
Using the Wang-Landau with this density means that points sampled in a stratum are distributed according to π_T, which is larger near the interior sphere than the exterior one. In other words, a suitable temperature T is expected to balance out the measure concentration effects. In the sequel, we compare $T = 0.1$ versus $T = 10^5$, this latter value defining an almost flat distribution close to the Lebesgue measure.

Of course to go back to the Lebesgue measure we need a good estimations of the integrals of Eq. (4.7), which might degrade the overall accuracy. However, we show that in dimension 15, for the Gaussian random walk, we get a noticeable improvement in convergence speed and relative error (Fig.4.8(Left)) using the Boltzmann distribution with low temperature than using it with a temperature so high that it is essentially the same than the Lebesgue measure. For another take on this, recall that the *climbing time* is the number of steps required to go from the bottommost stratum to the topmost stratum. Formally, the *climb* across d strata is defined by two times t_0 and t_1 such that

- $x_{t_0} \in \mathcal{E}_0$ and $x_{t_0-1} \notin \mathcal{E}_0$.
- $x_{t_1} \in \mathcal{E}_{d-1}$ and $\forall t \in [t_0, t_1[$, $x_t \notin \mathcal{E}_{d-1}$.

The *climbing time* is then $t_1 - t_0$. (see section 2.4.1) The shorter climb time for $T = 0.1$ illustrate the improved mixing (Fig. 4.8(Right)).

Figure 4.8 Relative errors with respect to Lebesgue measure (Left) and climb times (Right) using sampling done at different temperature (with Boltzmann measure) for the single well potential. A Gaussian random walk of variance 0.02 was used; a total of 40 runs were performed. The temperature $T = 10^5$ is essentially the Lebesgue measure.



4.5 Conclusion

The Wang-Landau (WL) algorithm is a recently developed stochastic algorithm initially designed to compute densities of states for physical systems, and also amenable to numerical integration.

To the best of our knowledge, this work presents the first versatile implementation of WL. Our architecture, based on template C++ classes, makes it possible to target two user profiles.

End-users can easily perform calculations of discrete and continuous systems, which merely requires plugging the specification of the system into our design.

For developers, our setup allows testing various building blocks and combinations thereof, and select those best suited for a system. The most crucial aspects are the energy discretisation strategy and the random walk, which, if poorly chosen, prevent convergence. We note in passing that the default implementations provided for move sets are state-of-the-art and should prove useful for future benchmarking.

Overall, our framework allows great versatility, while retaining reasonable default implementations. We anticipate that it will enjoy applications in physics (partition function), numerical integration, as well as machine learning (computation of maxima a posteriori), on systems involving from tens to hundreds of degrees of freedom.

Chapter 5

Outlook

This work lies at the crossroads of computational physics and randomized algorithms design.

More specifically, the thesis makes three contributions. First we introduce an efficient random walk for the Wang-Landau algorithm. Second, we modify Hamiltonian Monte Carlo to sample a bounded domain, providing theoretical results for the modified HMC and an analysis of the quality of the random walk for the computation of the volume of convex bodies. Finally, a robust implementation of the HMC random walk and a generic software design for Wang-Landau are provided.

Our work calls for further investigations, both for convex volume computation and density of state estimation.

For Wang-Landau and density of states computation in general, several areas call for further work. A first step could be to study Wang-Landau in a context similar to the computation of convex volumes. I believe such a setting would allow for a detailed proof of convergence and a much sharper theoretical convergence speed. To that end, proofs techniques developed to analyse the complexity of convex volume computation algorithms might prove useful. This might allow to isolate crucial features for fast convergence and increase both our theoretical and practical understanding of these algorithms.

Second, it seems clear to me after the work on HMC for convex volumes that the histogram approximation, while convenient, really hampers the ability to design efficient random walks. Indeed, the regularity of the sampled biased depends on the regularity of the approximation used for the Density of State. Using a histogram to estimate the DoS leads to a discontinuous biased density. However, efficient sampling strategies such as HMC or Langevin dynamics rely on the derivatives of the density of the sampled measure, and therefore cannot be used in Wang-Landau. Hence, while theoretical developments could use the simplicity of the histogram estimation, practical development should focus on continuous approximation of the density whenever possible (for example, discrete states cannot have a continuous representation).

Finally, I believe the learning rate needs to be better understood. The learning rate γ requires additional parameters (using the Flat Histogram criterion or not, but also the initial value for γ) with considerable impact on convergence speed, but with no sound theoretical basis to choose them. Besides, the question of the optimality of the current rules is not answered. This problem is not limited to Wang-Landau but seems common to this class of methods, including Umbrella sampling and Adaptive potential methods. For this analysis, I believe Wang-Landau is a good starting point since it uses a simple histogram approximation instead of a more complex kernel density estimation (or any complex method). To that end, I have developed some preliminaries ideas with encouraging results which might help bridge the gap between the exponential convergence sought after by the original authors of Wang-Landau and the practical $1/t$ rule.

For convex volumes computation, the proof and speed of convergence for HMC with reflections in convex do not yet take into account reflections, however they are necessary to escape corners and drastically reduces mixing time. More theoretical work is required in this area to find sharp mixing time bounds and their scaling with the dimension, as well as to understand the potential limitations and problematic cases for such a random walk. This in turn would allow theoretical proof of complexity for the complete convex volume

algorithm using HMC with reflections.

Last but not least, I believe it is worth investigating applying density of state estimation algorithm to convex volume estimation. Indeed, as stated in remark 1.10, the concentric ball algorithm for convex volume computation is estimating a density. Conversely, one might use the Wang-Landau algorithm with a suitable energy function ($x \mapsto ||x||$), in which case each bin stores the push forward measure i.e. the volume of the region sandwiched between two concentric spheres. Preliminary tests were performed during the PhD, with complexity seemingly of the same order of magnitude than the popular convex volume algorithms. However, the lack of target precision for Wang-Landau makes the comparison difficult and the potential gains of HMC seemed greater, hence this research direction was not fully explored. Besides, I hope that replacing the Wang-Landau histogram by a continuous approximation of the density of state will yield a good complexity.

Bibliography

[AD15] H. Afshar and J. Domke. Reflection, refraction, and Hamiltonian Monte Carlo. In *Advances in Neural Information Processing Systems*, pages 3007–3015, 2015.

[Ada06] S. Adams. Lectures on mathematical statistical mechanics, 2006.

[AM16] W. Atisattapong and P. Maruphanton. Obviating the bin width effect of the 1/t algorithm for multidimensional numerical integration. *Applied Numerical Mathematics*, 104:133–140, 2016.

[AM17] W. Atisattaponga and P. Marupanthornb. A 1/t algorithm with the density of two states for estimating multidimensional integrals. *Computer Physics Communications*, 220(122–128), 2017.

[ASV01] I. Andricioaei, J.E. Straub, and A.F. Voter. Smart darting Monte Carlo. *The Journal of Chemical Physics*, 114(16):6994–7000, 2001.

[aVF14] I. Emiris an V. Fisikopoulos. Efficient random-walk methods for approximating polytope volume. In *Proceedings of the thirtieth annual symposium on Computational geometry*, page 318. ACM, 2014.

[BBKG18] A. Barp, F-X. Briol, A. Kennedy, and M. Girolami. Geometry and dynamics for markov chain monte carlo. *Annual Review of Statistics and Its Application*, 5:451–471, 2018.

[BEF00] 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.

[Bet17] M. Betancourt. A conceptual introduction to Hamiltonian Monte Carlo. *arXiv preprint arXiv:1701.02434*, 2017.

[BF87] I. Bárány and Z. Füredi. Computing the volume is difficult. *Discrete & Computational Geometry*, 2(4):319–326, 1987.

[BGJM11] Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng. *Handbook of Markov Chain Monte Carlo*. CRC press, 2011.

[BJMD13] L. Bornn, P. Jacob, P. Del Moral, and A. Doucet. An adaptive interacting Wang–Landau algorithm for automatic density exploration. *Journal of Computational and Graphical Statistics*, 22(3):749–773, 2013.

[BMP08] R. Belardinelli, S. Manzi, and V. Pereyra. Analysis of the convergence of the 1/ t and Wang-Landau algorithms in the calculation of multidimensional integrals. *Physical Review E*, 78(6):067701, 2008.

[BP07a] R.E. Belardinelli and V.D. Pereyra. Fast algorithm to calculate density of states. *Physical Review E*, 75(4):046701, 2007.

[BP07b] R.E. Belardinelli and V.D. Pereyra. Wang–Landau algorithm: A theoretical analysis of the saturation of the error. *The Journal of chemical physics*, 127(18):184105, 2007.

[BP16] R. Belardinelli and V. Pereyra. Nonconvergence of the wang-landau algorithms with multiple random walkers. *Physical Review E*, 93(5):053306, 2016.

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

[Cai11] W. Cai. Me346a introduction to statistical mechanics, 2011.

[CC18a] A. Chevallier and F. Cazals. A generic framework for Wang-Landau type algorithms. *NA*, 2018.

[CC18b] A. Chevallier and F. Cazals. Wang-landau algorithm: an adapted random walk to boost convergence. *NA*, 2018.

[CD17] F. Cazals and T. Dreyfus. The Structural Bioinformatics Library: modeling in biomolecular science and beyond. *Bioinformatics*, 7(33):1–8, 2017.

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

[CV15] B. Cousins and S. Vempala. Bypassing KLS: Gaussian cooling and an $O^*(n^3)$ volume algorithm. In *ACM STOC*, pages 539–548. ACM, 2015.

[CV16] B. Cousins and S. Vempala. A practical volume algorithm. *Mathematical Programming Computation*, 8(2):133–160, 2016.

[CV18] 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.

[DF88] M. Dyer and A. Frieze. On the complexity of computing the volume of a polyhedron. *SIAM Journal on Computing*, 17(5):967–974, 1988.

[DFK91] 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.

[EF18] I. Emiris and V. Fisikopoulos. Practical polytope volume approximation. *ACM Trans. on Math. Software*, 44(4), 2018.

[FJK+15] G. Fort, B. Jourdain, E. Kuhn, T. Lelièvre, and G. Stoltz. Convergence of the wang-landau algorithm. *Mathematics of Computation*, 84(295):2297–2327, 2015.

[FLE19] A. Farris, Y-W. Li, and M. Eisenbach. Histogram-free multicanonical monte carlo sampling to calculate the density of states. *Computer Physics Communications*, 235:297–304, 2019.

[GP14] E. Gryazinaa and B. Polyak. Random sampling: Billiard walk algorithm. *arXiv*, 2014.

[HCT+17] 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.

[Jan12] W. Janke. Monte carlo simulations in statistical physics: From basic principles to advanced applications. *Order, Disorder and Criticality: Advanced Problems of Phase Transition Theory*, 3:93–166, 2012.

[JH06] C. Junghans and U.H. Hansmann. Numerical comparison of WANG–LANDAU sampling and parallel tempering for met-enkephalin. *International Journal of Modern Physics C*, 17(06):817–824, 2006.

[JP16] W. Janke and W. Paul. Thermodynamics and structure of macromolecules from flat-histogram monte carlo simulations. *Soft matter*, 12(3):642–657, 2016.

[JR14] P. Jacob and R. Ryder. The Wang–Landau algorithm reaches the flat histogram criterion in finite time. *The Annals of Applied Probability*, 24(1):34–53, 2014.

[KLS97] Ravi Kannan, László Lovász, and Miklós Simonovits. Random walks and an o*(n5) volume algorithm for convex bodies. *Random Structures & Algorithms*, 11(1):1–50, 1997.

[LB14] D. Landau and K. Binder. *A guide to Monte Carlo simulations in statistical physics*. Cambridge university press, 2014.

[LC10] F. Lou and P. Clote. Thermodynamics of RNA structures by wang–landau sampling. *Bioinformatics*, 26(12):i278–i286, 2010.

[Lev97] S. Levy. *Flavors of Geometry*. Cambridge University Press, 1997.

[Li11] S. Li. Concise formulas for the area and volume of a hyperspherical cap. *Asian Journal of Mathematics and Statistics*, 4(1):66–70, 2011.

[LK99] 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.

[LLV06] 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.

[Lov99] L. Lovász. Hit-and-run mixes fast. *Mathematical Programming, Series B*, 86:443–461, 12 1999.

[LP17] D. Levin and Y. Peres. *Markov chains and mixing times*, volume 107. American Mathematical Soc., 2017.

[LS87] Z. Li and H.A. Scheraga. Monte Carlo-minimization approach to the multiple-minima problem in protein folding. *PNAS*, 84(19):6611–6615, 1987.

[LSR10] T. Lelièvre, G. Stoltz, and M. Rousset. *Free energy computations: A mathematical perspective*. World Scientific, 2010.

[LTE04] D.P Landau, S-H. Tsai, and M. Exler. A new approach to monte carlo simulations in statistical physics: Wang-landau sampling. *American Journal of Physics*, 72(10):1294–1302, 2004.

[LV03] L. Lovász and S. Vempala. Hit-and-run is fast and fun. *preprint, Microsoft Research*, 2003.

[LV04] 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.

[LWLL07] Ying Wai Li, Thomas Wüst, David P Landau, and HQ Lin. Numerical integration using wang–landau sampling. *Computer physics communications*, 177(6):524–529, 2007.

[MBdD+18] 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.

[MGCM12] Katie A Maerzke, Lili Gai, Peter T Cummings, and Clare McCabe. Incorporating configurational-bias monte carlo into the wang-landau algorithm for continuous molecular systems. *The Journal of Chemical Physics*, 137(20):204105, 2012.

[Mül01] N. Müller. The iRRAM: Exact arithmetic in C++. In *Computability and Complexity in Analysis*, pages 222–252. Springer, 2001.

[OMG10] Pedro Ojeda-May and Martin E Garcia. Electric field-driven disruption of a native β-sheet protein conformation and generation of a helix-structure. *Biophysical journal*, 99(2):595–599, 2010.

[Pal82] RG Palmer. Broken ergodicity. *Advances in Physics*, 31(6):669–735, 1982.

[PCA+06] P. Poulain, F. Calvo, R. Antoine, M. Broyer, and P. Dugourd. Performances of wang-landau algorithms for continuous systems. *Physical Review E*, 73(5):056704, 2006.

[PG14] B. Polyak and E.N. Gryazina. Billiard walk-a new sampling algorithm for control and optimization. *IFAC Proceedings Volumes*, 47(3):6123–6128, 2014.

[PP14] A. Pakman and L. Paninski. Exact hamiltonian monte carlo for truncated multivariate gaussians. *Journal of Computational and Graphical Statistics*, 23(2):518–542, 2014.

[PS85] F. Preparata and M. Shamos. *Computational geometry: an introduction*. Springer Science & Business Media, 1985.

[Rad12] 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.

[RC13] C. Robert and G. Casella. *Monte Carlo statistical methods*. Springer Science & Business Media, 2013.

[RDRC16] A. Roth, T. Dreyfus, C.H. Robert, and F. Cazals. Hybridizing rapidly growing random trees and basin hopping yields an improved exploration of energy landscapes. *J. Comp. Chem.*, 37(8):739–752, 2016.

[RKIdP03] Nitin Rathore, Thomas A Knotts IV, and Juan J de Pablo. Density of states simulations of proteins. *The Journal of chemical physics*, 118(9):4285–4290, 2003.

[RR01] G. Roberts and J. Rosenthal. Optimal scaling for various metropolis-hastings algorithms. *Statistical science*, 16(4):351–367, 2001.

[RR04a] Gareth O. Roberts and Jeffrey S. Rosenthal. General state space markov chains and mcmc algorithms. *Probability Surveys*, 2004.

[RR04b] G.O. Roberts and J.S. Rosenthal. General state space markov chains and MCMC algorithms. *Probability Surveys*, 2004.

[RR07] J.S. Rosenthal and G.O. Roberts. Coupling and ergodicity of adaptive MCMC. *Journal of Applied Probablity*, 44:458–475, 2007.

[SA11] A. Swetnam and M. Allen. Improving the wang–landau algorithm for polymers and proteins. *Journal of computational chemistry*, 32(5):816–821, 2011.

[SCK10] H. Stamati, C. Clementi, and L. Kavraki. Application of nonlinear dimensionality reduction to characterize the conformational landscape of small peptides. *Proteins: Structure, Function, and Bioinformatics*, 78(2):223–235, 2010.

[Smi99] P. Smith. The alanine dipeptide free energy surface in solution. *The Journal of chemical physics*, 111(12):5568–5579, 1999.

[SNY11] Hiromitsu Shimoyama, Haruki Nakamura, and Yasushige Yonezawa. Simple and effective application of the wang–landau method for multicanonical molecular dynamics simulation. *The Journal of Chemical Physics*, 134(2):024109, 2011.

[SW11] C. Sminchisescu and M. Welling. Generalized darting Monte Carlo. *Pattern Recognition*, 44(10):2738–2748, 2011.

[SW13] S. Somani and D. J. Wales. Energy landscapes and global thermodynamics for alanine peptides. *The Journal of Chemical Physics*, 139(12), 2013.

[Tie98] L. Tierney. A note on Metropolis-Hastings kernels for general state spaces. *Annals of applied probability*, pages 1–9, 1998.

[Wal03] D. J. Wales. *Energy Landscapes*. Cambridge University Press, 2003.

[WL01] F. Wang and D.P. Landau. Efficient, multiple-range random walk algorithm to calculate the density of states. *Physical review letters*, 86(10):2050, 2001.

[WS14] D. Wales and P. Salamon. Observation time scale, free-energy landscapes, and molecular symmetry. *Proceedings of the National Academy of Sciences*, 111(2):617–622, 2014.

[WSTP15] B. Werlich, T. Shakirov, M. Taylor, and W. Paul. Stochastic approximation monte carlo and wang–landau monte carlo applied to a continuum polymer model. *Computer Physics Communications*, 186:65–70, 2015.

[YD95] C. Yap and T. Dubé. The exact computation paradigm. In *Computing in Euclidean Geometry*, pages 452–492. World Scientific, 1995.

Appendix A

Appendix

A.1 Appendix: force field and Hessian of force field

The SBL incorporates force field computations for molecules in vacuum. During my PhD, I worked on two parts:

- optimizing the force field computation code,
- and enhancing the symbolic differentiation procedure to factorize common terms in the computation of the Hessian matrix of a force field.

Practically, I contributed to the following packages molecular potential energy (`https://sbl.inria.fr/doc/group__Molecular__potential__energy-package.html`), and covalent structure (`https://sbl.inria.fr/doc/group__Molecular__covalent__structure-package.ht`).ml

Optimizing force field computation: data locality. Potential energy computation for bio-molecules using common force fields such as AMBER or CHARMM requires traversing the covalent structure to iterate over all bounds, bounds angles and torsion angles. The SBL is using a graph structure to store and manipulate the covalent structure. However, this structure, while very convenient for manipulating/editing the covalent structure, has poor performances when it comes to iterating over bounds, bounds angles and torsion angles. The reason is that it is not what is called *cache friendly*.

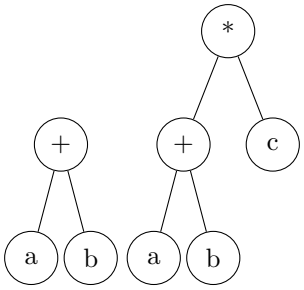
To understand what this means and how to improve performances, a brief summary of modern computers architectures is needed. In modern CPUs, fetching data from the memory is a very costly operation. Typically, it takes around 100ns to fetch a variable stored in the RAM, which results in hundreds of unused CPU cycles. For comparison, multiplying two doubles takes 1 or 2 cycles (or slightly more) depending on the CPU architecture, and maybe even less with advanced vector instructions. Therefore, memory access is a major bottleneck of modern architectures. To mitigate this issue, CPUs have a very fast *on-chip* memory called *cache*. This memory, sometimes divided into multiple levels, is much smaller than the RAM, with a few MB. When the CPU operates, it moves working variables from the RAM to the cache, to benefit from the very fast access time to the cache. When the CPU requires a variable that is not stored in the cache, but in the main memory, a high cost must be paid, and the CPU will sit idle while waiting for the variable. This event is called a *cache miss*.

To minimize cache misses, the CPU tries to guess which variables will be used in advance to store them in the cache before they are needed. The main mechanism for that is to rely on *data locality*. Data locality is an assumption made by the CPU that data used together are stored contiguously in memory. Consider for instance, an array A of 50 double stored contiguously in memory. If one wishes to iterate over the array when the first value $A[0]$ is requested , the CPU will fetch not only $A[0]$, but also all values up to $A[16]$ (the number of values depends on the architecture of your CPU). Then the next values already sits in the cache when requested, and thus do not generate *cache misses*. A *cache friendly* program is a program that strives to reduce the number of *cache misses*, by using data structures providing good *data locality*.

Therefore, we spent considerable efforts to provide a new data structure for covalent structure in the SBL tailored to minimize cache misses for potential energy computation. At the same time, we changed data structure storing force fields parameters to more efficient data structure, be it for their locality but also for their access time (vectors vs maps). These efforts were required to run any Wang-Landau simulation using force fields, and brought a x100 to x1000 speed-up over the existing force field implementation of the SBL.

Hessian of Force Field: simplifying symbolic differentiation results Symbolic differentiation was used in the SBL to generate the gradient of the potential energy function for force fields. We used a similar strategy to compute second derivative of the potential energy function in order to compute the Hessian without using finite differences–an extremely costly strategy in high dimension. However, straightforward symbolic differentiation yields a very unoptimized program, especially for complex terms such as those associated with torsion angles. Each coefficient of the Hessian is evaluated independently from the others. And therefore, each common factors is recomputed several times.

Thus, we have written a python script that looks for common factors in the Hessian coefficients. We recall that a mathematical expression can be expressed as a tree. The label of each node is the operations (for example, '+'), the children represent the operands, and the leaf can be numerical values or symbolic variables. For example, $a + b$ and $(a + b) * c$ reads as



Therefore, finding common factors is equivalent to finding common subtrees in different graphs. For cost and simplicity reasons, we do not consider operation reordering (i.e. two different trees representing the same expression). The algorithm is as follow:

1. set h the maximum height of all trees in the union of trees given by each coefficient of the Hessian matrix
2. find all common sub-trees of size h in the union of trees given by each coefficient of the Hessian matrix
3. set $h = h - 1$
4. if $h > 2$, go back to step 2

The python script uses the package *sympy* for all symbolic computation: the actual differentiation and the expression tree manipulation. The end result is a vastly shorter output C++ file (the script generates C++) since common factors are factored out, especially for torsion angles, since they involve 4 atoms, hence 12 coordinates, hence 144 terms in the Hessian matrix. This brings down compilation time, but the main advantage of thee factorization is the run time speed up.