Sampling with Riemannian Hamiltonian Monte Carlo in a Constrained Space

Yunbum Kook[*], Yin Tat Lee[†], Ruoqi Shen[‡], Santosh S. Vempala[§]

February 7, 2022

Abstract

We demonstrate for the first time that ill-conditioned, non-smooth, constrained distributions in very high dimension, upwards of 100,000, can be sampled efficiently *in practice*. Our algorithm incorporates constraints into the Riemannian version of Hamiltonian Monte Carlo and maintains sparsity. This allows us to achieve a mixing rate independent of smoothness and condition numbers.

On benchmark data sets in systems biology and linear programming, our algorithm outperforms existing packages by orders of magnitude. In particular, we achieve a 1,000-fold speed-up for sampling from the largest published human metabolic network (RECON3D). Our package has been incorporated into the COBRA toolbox.

1 Introduction

Sampling is Fundamental. Sampling algorithms arise naturally in models of statistical physics, e.g., Ising, Potts models for magnetism, Gibbs model for gases, etc. These models directly suggest Markov chain algorithms for sampling the corresponding configurations. In the Ising model where the vertices of a graph are assigned a spin, i.e., ± 1, in each step, we pick a vertex at random and flip its spin with some probability. The probability is chosen so that the distribution of the vector of all spins approaches a target distribution where the probability exponentially decays with the number of agreements in spin for pairs corresponding to edges of the graph. In the Gibbs model, particles move randomly with collisions and their motion is often modeled as reflecting Brownian motion. Sampling with Markov chains is today the primary algorithmic approach for high-dimensional sampling. For some fundamental problems, sampling with Markov chains is the only known efficient approach or the only approach to have guarantees of efficiency. Two notable examples are sampling perfect matchings of a bipartite graph and sampling points from a convex body. These are the core subroutines for estimating the permanent of a nonnegative matrix and estimating the volume of a convex body, respectively. The solution space for these problems scales exponentially with the dimension. In spite of this, polynomial-time algorithms have been discovered for both problems. The current best permanent algorithm scales as n^7 (time) [1, 20], while the current best volume algorithm scales as $n^{3+o(1)}$ (number of membership tests) [21]. For the latter, the first polynomial-time algorithm had a complexity of n^{27} [14], and the current best complexity is the result of many breakthrough discoveries, including general-purpose algorithms and analysis tools.

Sampling is Ubiquitous. The need for efficient high-dimensional sampling arises in many fields. A notable setting is *metabolic networks* in systems biology. A constraint-based model of a metabolic network consists of m metabolites and n reactions, and a set of equalities and inequalities that define a set of feasible steady state reaction rates (fluxes):

$$\Omega = \left\{ v \in \mathbb{R}^n \mid Sv = 0,\, l \le v \le u,\, c^T v = \alpha \right\}$$

[*]Georgia Tech, yb.kook@gatech.edu
[†]University of Washington and Microsoft Research, yintat@uw.edu
[‡]University of Washington, shenr3@cs.washington.edu
[§]Georgia Tech, vempala@gatech.edu

where S is a stoichiometric matrix with coefficients for each metabolite and reaction. The linear equalities ensure that the fluxes into and out of every node are balanced. The inequalities arise from thermodynamical and environmental constraints. Sampling constraint-based models is a powerful tool for evaluating the metabolic capabilities of biochemical networks [30, 43]. While the most common distribution used is uniform over the feasible region, researchers have also argued for sampling from the Gaussian density restricted to the feasible region; the latter has the advantage that the feasible set does not have to be bounded. A previous approach to sampling, using hit-and-run with rounding [17], has been incorporated into the COBRA package [18] for metabolic systems analysis (Bioinformatics).

A second example of mathematical interest is the problem of computing the volume of the Birkhoff polytope. For a given dimension n, the Birkhoff polytope is the set of all doubly stochastic $n \times n$ matrices (or the convex hull of all permutation matrices). This object plays a prominent role in algebraic geometry, probability, and other fields. Computing its volume has been pursued using algebraic representations; however exact computations become intractable even for $n = 11$, requiring years of computation time. Hit-and-run has been used to show that sampling-based volume computation can go to higher dimension [9], with small error of estimation. However, with existing sampling implementations, going beyond $n = 20$ seems prohibitively expensive.

A third example is from Machine Learning, a field that is increasingly turning to *sampling* models of data according to their performance in some objective. One such commonly used criterion is the logistic regression function. The popularity of logistic regression has led to sampling being incorporated into widely used packages such as STAN [41], PyMC3 [38], and Pyro [2].

Problem Description. In this paper, we consider the problem of sampling from distributions whose densities are of the form

$$e^{-f(x)} \text{ subject to } Ax = b, x \in K \tag{1.1}$$

where f is a convex function and K is a convex set. We assume that a self-concordant barrier ϕ for K is given. Note that any convex set has a self-concordant barrier [29] and there are explicit barriers for the majority of convex sets we use in practice [36], so this is a mild assumption. We introduce an algorithm to handle this problem efficiently when K is a product set of K_i, each with small dimension. Many practical applications can be written in this form with small dimensional K_i. As a special case, the algorithm can handle K in the form of $\{x \in \mathbb{R}^n : l_i \leq x_i \leq u_i \text{ for all } i \in [n]\}$ with $l_i \in \mathbb{R} \cup \{-\infty\}$ and $u_i \in \mathbb{R} \cup \{+\infty\}$, which is the common model structure in systems biology. Moreover, any generalized linear model $\exp(-\sum f_i(a_i^\top x - b_i))$, e.g., the logistic model, can be rewritten in the form

$$\exp(-\sum t_i) \text{ subject to } Ax = b + s, (s,t) \in K \tag{1.2}$$

where $K = \Pi K_i$ and $K_i = \{(s_i, t_i) : f_i(s_i) \leq t_i\}$ are 2-dimensional convex sets.

The Challenges of Practical Sampling. High dimensional sampling has been widely studied in both the theoretical computer science and the statistics community. Many popular samplers are first-order methods, such as MALA [37], basic HMC [34, 12] and NUTS [19], which update the Markov chain based on the gradient information of f. The runtime of such methods can depend on the condition number of the function f [13, 26, 6, 7, 39]. However, the condition number of real-world applications can be very large. For example, Recon 1 [24], a reconstruction of the human metabolic network, can have condition number as large as 10^6 due to the dramatically different orders of different chemicals' concentration. Motivated by sampling from ill-conditioned distributions, another class of samplers use higher-order information such as Hessian of f to take into account the local structure of the problems [40, 8]. However, such samplers cannot handle non-smooth distributions, such as hinge-loss, lasso, and uniform densities over polytopes.

For non-smooth distributions, the best polytime methods are based on discretizations of Brownian motion, e.g., the Ball walk [22] (and its affine-invariant cousin, the Dikin walk [23]), which takes a random step in a ball of a fixed size around the current point. Hit-and-Run [31] builds on these by avoiding an explicit step size and going to a random point along a random line through the current point. Both approaches hit the same bottleneck — in a polytope that contains a unit ball, the step size should be $O(1/\sqrt{n})$ to avoid stepping out of the body with large probability. This leads to quadratic bounds (in dimension) on the number of steps to "mix".

Due to the reduction mentioned in (1.2), non-smooth distributions can be translated to the form in (1.1) with constraint K. Both the first and higher-order sampler and the polytime non-smooth samplers have their limitations in handling distributions with non-smooth objective function or constraint K. Given the limitations of all previous samplers, a natural question we want to ask is the following.

Question. *Can we develop a sampler that can handle the constrained problem in (1.1) and preserve sparsity with mixing time independent of the condition number?*

In some applications, smoothness and condition number can be controlled with tailor-made models. Our goal here is to propose a general solver that can sample from any non-smooth distributions as given. For traditional samplers such as the Ball walk and Hit-and-Run, as mentioned earlier, the step size needs to be small so that the process does not step out. An approach that gets around this bottleneck is Hamiltonian Monte Carlo (HMC), where the next step is given by a point along a Hamiltonian-preserving curve according to a suitably chosen Hamiltonian. It has two advantages. First, the steps are no longer straight lines in Euclidean space, and we no longer have the concern of "stepping out". Second, the process is *symplectic* and measure-preserving, and hence the filtering step is easy to compute. It was shown in [28] that significantly longer steps can be taken and the process with a convergence analysis in the setting of Hessian manifolds, leading to subquadratic convergence for uniformly sampling polytopes.

To make this practical, however, remains a challenge. There are two high-level difficulties. One is that many real-world instances are highly skewed (far from isotropic) and hence it is important to use the local geometry of the density function. This means efficiently computing or maintaining second-order information such as a Hessian of the logarithm of the density. This can be done in the Riemannian HMC (RHMC) framework [15, 28], but the computation of the next step requires solving the Hamiltonian ODE to high accuracy, which in turn needs the computation of leverage scores, a procedure that takes at least matrix multiplication time in the worst case. Another important difficulty is maintaining hard linear constraints. Existing high-dimensional packages do allow for constraints (they must be somehow incorporated into the target density), and RHMC is usually considered with a full-dimensional feasible region such as a full-dimensional polytope. This can also be done in the presence of linear equalities by working in the affine subspace defined by the equalities, but this has the effect of losing any sparsity inherent in the problem and turning all coefficient matrices and objective coefficients into dense objects, thereby potentially incurring a quadratic blow-up.

Our Solution: Constrained Riemannian Hamiltonian Monte Carlo (CRHMC). We develop a constrained version of RHMC, maintaining both sparsity and constraints. Our refinement of RHMC ensures that the process satisfies the given constraints throughout, without incurring a significant overhead in time or sparsity. It works even if the resulting feasible region is poorly conditioned. Since many instances in practice are ill-conditioned and have degeneracies, we believe this is a crucial aspect. Our algorithm outperforms existing packages by orders of magnitude.

In Section 2, we describe the algorithm in detail starting with its main ingredients, and concluding with a proof that it converges to the desired distribution. We also bound the error of the simple ODE solver we use (midpoint Euler). Following that, in Section 3, we present empirical results on several benchmark datasets, showing that CRHMC successfully samples much larger models than previously known to be possible, and is significantly faster in terms of rate of convergence ("number of steps") and total sampling time. Our complete package is available on Github.

2 Algorithm: Constrained RHMC

In this section, we propose a constrained Riemannian Hamiltonian Monte Carlo (CRHMC[1]) algorithm to sample from a distributions of the form

$$e^{-f(x)} \text{ subject to } c(x) = 0$$

[1] pronounced "crumch".

where the constraint function $c : \mathbb{R}^n \to \mathbb{R}^m$ satisfies the property that the Jacobian $Dc(x)$ has full rank for all x such that $c(x) = 0$. It is useful to keep in mind the case when $c(x) = 0$ is an affine subspace $Ax = b$, in which case $Dc(x) = A$, and the full-rank condition simply says that the rows of A are independent.

We note that the constrained Hamiltonian Monte Carlo (CHMC) [3] provided the same framework that can be extended to CRHMC, and in fact they mention CRHMC as a possible variant. However, their algorithm for CRHMC requires eigenvalue decomposition and is not efficient for large problems. To overcome their limitations, we start by going over basics of RHMC and CRHMC with more explicit mathematical description in Section 2.1 and 2.2 respectively, including a proof of stationarity in Section 2.3. In Section 2.4, we claim that the introduction of a self-concordant barrier on a Hessian manifold leads to CRHMC being independent of the condition number of distribution under a proper choice of a mass matrix. Then in Section 2.5 we illustrate how we avoid costly steps to make CRHMC more efficient. Lastly in Section 2.6 we demonstrate how we discretize CRHMC using the implicit midpoint method, together with proofs for stationarity of the discretized CRHMC and for fast convergence of the numerical integrator in our setting. For missing definitions and details, we refer readers to Appendix A.

2.1 Basics of RHMC

To introduce our algorithm, we first recall the RHMC algorithm (Algorithm 1). In RHMC, we extend the space x to the pair (x, v), where v denotes the *velocity*. Instead of sampling from $e^{-f(x)}$, RHMC samples from the distribution $e^{-H(x,v)}$, where $H(x, v)$ is the Hamiltonian, and then outputs x. To make sure the distribution is correct, we choose the Hamiltonian such that the marginal of $e^{-H(x,v)}$ along v is proportional to $e^{-f(x)}$. One common choice of $H(x, v)$ is

$$H(x, v) = f(x) + \frac{1}{2} v^\top M(x)^{-1} v + \frac{1}{2} \log\det M(x) \tag{2.1}$$

where $M(x)$ is a position-dependent positive definite matrix defined on $\mathbb{R}^n$.

RHMC contains two steps. Step 1 samples v. For H of the form (2.1), it involves simply sampling from a multivariate normal distribution. Step 2 rotates the (x, v)-space via the Hamiltonian dynamics

$$\frac{dx}{dt} = \frac{\partial H(x, v)}{\partial v}, \quad \frac{dv}{dt} = -\frac{\partial H(x, v)}{\partial x}. \tag{2.2}$$

Note that the target distribution is invariant under this ODE. Hence, one can think of this step as moving the randomness of Step 1 from v to x.

Algorithm 1: `Riemannian Hamiltonian Monte Carlo` (RHMC)

```
Input: Initial point $x^{(0)}$, step size $h$
for $k = 1, 2, \cdots$ do
    // Step 1: resample $v$
    Sample $v^{(k-\frac{1}{2})} \sim \mathcal{N}(0, M(x^{(k-1)}))$ and set $x^{(k-\frac{1}{2})} \leftarrow x^{(k-1)}$.

    // Step 2: Hamiltonian dynamics
    Solve the ODE
        $\frac{dx}{dt} = \frac{\partial H(x,v)}{\partial v}, \quad \frac{dv}{dt} = -\frac{\partial H(x,v)}{\partial x}$
     with $H$ defined in (2.1) and the initial point given by $(x^{(k-\frac{1}{2})}, v^{(k-\frac{1}{2})})$.
    Set $x^{(k)} \leftarrow x(h)$ and $v^{(k)} \leftarrow v(h)$.
end
Output: $x^{(k)}$
```

2.2 Basics of CRHMC

To extend RHMC to the constrained case, we need to make sure both Step 1 and Step 2 satisfy the constraints, so the Hamiltonian dynamic has to maintain $c(x) = 0$ throughout Step 2. Note that

$$\frac{d}{dt}c(x_t) = Dc(x_t) \cdot \frac{dx_t}{dt} = Dc(x_t) \cdot \frac{\partial H(x_t, v_t)}{\partial v_t} \tag{2.3}$$

where $Dc(x)$ is the Jacobian of c at x. With H defined in (2.1), Condition (2.3) becomes $Dc(x)M(x)^{-1}v = 0$. However, for full rank $Dc(x)$, if $M(x)$ is invertible, then $\mathrm{Range}(v) = \mathrm{Range}(\mathcal{N}(0, M(x))) = \mathbb{R}^n$ immediately violates this condition due to $\dim(\mathrm{Null}(Dc(x)M^{-1}(x))) = n - m$. To get around this issue, we use a non-invertible matrix $M(x)$ with its pseudo-inverse $M(x)^\dagger$ to satisfy $Dc(x)M(x)^\dagger v = 0$ for any $v \in \mathrm{Range}(M(x))$. Since we want the step to be able to move in all directions satisfying $c(x) = 0$, we impose the following condition with $\mathrm{Range}(M(x)) = \mathrm{Range}(M(x)^\dagger)$ in mind:

$$\mathrm{Range}(M(x)) = \mathrm{Null}(Dc(x)) \text{ for all } x \in \mathbb{R}^n, \tag{2.4}$$

which can be achieved by $M(x)$ proposed in Section 2.4.

Under the condition (2.4), we sample v from $\mathcal{N}(0, M(x))$ in Step 1, which is equivalent to sampling from $e^{-H(x,v)}$ subject to $v \in \mathrm{Range}(M(x)) = \mathrm{Null}(Dc(x))$. Also, the stationary distribution of CRHMC should be proportional to

$$e^{-H(x,v)} \text{ subject to } c(x) = 0 \text{ and } v \in \mathrm{Null}(Dc(x)).$$

Here, to maintain $v \in \mathrm{Null}(Dc(x))$ during Step 2 we add a Lagrangian term to H. Without the Lagrangian term, v_t would escape from $\mathrm{Null}(Dc(x_t)) = \mathrm{Range}(M(x_t))$ in Step 2 as seen in Lemma 1 below, which contradicts $\mathrm{Range}(v_t) = \mathrm{Range}(\mathcal{N}(0, M(x_t))) = \mathrm{Range}(M(x_t))$. Therefore, the constrained Hamiltonian we propose is

$$H(x, v) = \overline{H}(x, v) + \lambda(x, v)^\top c(x) \quad \text{with} \quad \overline{H}(x, v) = f(x) + \frac{1}{2}v^\top M(x)^\dagger v + \log \mathrm{pdet}(M(x)) \tag{2.5}$$

where pdet denotes pseudo-determinant and $\lambda(x, v)$ is picked so that $v \in \mathrm{Null}(Dc(x))$ (see Lemma 1). Later in Section 2.5, we present alternative formulas for $M(x)^\dagger$ and $\log \mathrm{pdet}(M(x))$ for efficient computation, and for our case $c(x) = Ax - b$ we prove that the constrained Hamiltonian H can be simplified without affecting the dynamic on x.

Lemma 1. *Consider the constrained Hamiltonian defined by (2.5) with* $\mathrm{Range}(M(x)) = \mathrm{Null}(Dc(x))$ *and*

$$\lambda(x_t, v_t) = (Dc(x_t)Dc(x_t)^\top)^{-1}\left(D^2c(x_t)[v_t, \frac{dx_t}{dt}] - Dc(x_t)\frac{\partial \overline{H}(x_t, v_t)}{\partial x}\right).$$

When the initial point satisfies $c(x_0) = 0$, the ODE solution of (2.2) satisfies $c(x_t) = 0$ and $Dc(x_t)v_t = Dc(x_0)v_0$ for all t.

Proof. First we compute

$$\begin{aligned}\frac{d}{dt}c(x_t) &= Dc(x_t) \cdot \frac{dx_t}{dt} = Dc(x_t) \cdot \frac{\partial H(x_t, v_t)}{\partial v_t}\\ &= Dc(x_t)M(x_t)^\dagger v + Dc(x_t)D_v\lambda(x_t, v_t)^\top c(x_t)\\ &= Dc(x_t)D_v\lambda(x_t, v_t)^\top c(x_t)\end{aligned}$$

where we used $\mathrm{Range}(M(x)^\dagger) = \mathrm{Range}(M(x)) = \mathrm{Null}(Dc(x))$. Since $c(x_0) = 0$, by the uniqueness of the ODE solution, we have that $c(x_t) = 0$ for all t. Next we compute

$$\begin{aligned}\frac{dv_t}{dt} &= -\frac{\partial H(x_t, v_t)}{\partial x}\\ &= -\frac{\partial \overline{H}(x_t, v_t)}{\partial x} - Dc(x_t)^\top \lambda(x_t, v_t) - D_x\lambda(x_t, v_t)^\top c(x_t)\\ &= -\frac{\partial \overline{H}(x_t, v_t)}{\partial x} - Dc(x_t)^\top \lambda(x_t, v_t)\end{aligned}$$

where we used $c(x_t) = 0$. Hence, we have

$$\begin{aligned}\frac{d}{dt}Dc(x_t)v_t &= D^2c(x_t)[v_t, \frac{dx_t}{dt}] + Dc(x_t)\frac{dv_t}{dt} \\ &= D^2c(x_t)[v_t, \frac{dx_t}{dt}] - Dc(x_t)\frac{\partial \overline{H}(x_t, v_t)}{\partial x} - Dc(x_t)Dc(x_t)^\top \lambda(x_t, v_t).\end{aligned}$$

Setting $\lambda(x_t, v_t) = (Dc(x_t)Dc(x_t)^\top)^{-1}(D^2c(x_t)[v_t, \frac{dx_t}{dt}] - Dc(x_t)\frac{\partial \overline{H}(x_t,v_t)}{\partial x})$, we have that $\frac{d}{dt}Dc(x_t)v_t = 0$ and thus $Dc(x_t)v_t = Dc(x_0)v_0$ for all t (i.e., $v_t \in \mathrm{Null}(Dc(x_t))$ during Step 2). ☐

Remark. Rather than resampling the velocity at every step, we may change the velocity more gradually, using the following update:

$$v' \leftarrow \sqrt{\beta}v + \sqrt{1-\beta}z$$

where $z \sim \mathcal{N}(0, M(x))$ and β is a parameter. We note that this step is time-reversible, i.e., $\mathbf{P}(v|x)\mathbf{P}(v \to v') = \mathbf{P}(v'|x)\mathbf{P}(v' \to v)$ (see Theorem 8).

2.3 Stationarity of CRHMC

An algorithmic description of CRHMC is the same as Algorithm 1 with the constrained H in place of the unconstrained $\overline{H}$. In this section, we prove that the Markov chain defined by CRHMC projected to x satisfies detailed balance with respect to its target distribution proportional to $e^{-f(x)}$ subject to $c(x) = 0$, leading to the target distribution being stationary.

To formalize this, we introduce a few notations here. Let $\mathcal{M} = \{x \in \mathbb{R}^n : c(x) = 0\}$ be a manifold in $\mathbb{R}^n$ and $\pi(x)$ be a desired distribution on $\mathcal{M}$ proportional to $e^{-f(x)}$ satisfying $\int_{\mathcal{M}} \pi(x)dx = 1$. We denote the set of velocity v at $x \in \mathcal{M}$ (i.e., cotangent space) by $\mathcal{T}_x\mathcal{M} = \mathrm{Null}(Dc(x)) = \{v \in \mathbb{R}^n : Dc(x)M(x)^\dagger v = 0\}$. Let T_h be the map sending (x, v) to $(x', v') = (x(h), y(h))$ in Step 2 and define $F_{x,h}(v) := (\pi_1 \circ T_h)(x, v) = x'$, where $\pi_1(x, v) := x$ is the projection to the position space x. For a matrix A, we denote by $|A|$ the absolute value of its determinant $|\det(A)|$.

Theorem 2. *For $x, x' \in \mathcal{M}$, let $\mathbf{P}_x(x')$ be the probability density of the one-step distribution to x' starting at x in CRHMC (i.e., transition kernel from x to x'). It satisfies detailed balance with respect to the desired distribution π (i.e., $\pi(x)\mathbf{P}_x(x') = \pi(x')\mathbf{P}_{x'}(x)$).*

Proof. Fix x and x' in $\mathcal{M}$. Let C_1 be the normalization constant of $e^{-f(x)}$ (i.e., $\pi(x) = C_1e^{-f(x)}$). The transition kernel $\mathbf{P}_x(x')$ is characterized as the pushforward by $F_{x,h}$ of the probability measure $v \sim \mathcal{N}(0, M(x))$ on $\mathcal{T}_x\mathcal{M}$, so it follows that

$$\mathbf{P}_x(x') = C_2 \int_{V_x} \frac{e^{-\frac{1}{2}\log \mathrm{pdet}(M(x)) - \frac{1}{2}v^\top M(x)^\dagger v}}{|DF_{x,h}(v)|}dv,$$

where C_2 is the normalization constant of $e^{-\frac{1}{2}\log \mathrm{pdet}(M(x)) - \frac{1}{2}v^\top M(x)^\dagger v}$ and $V_x = \{v \in \mathcal{T}_x\mathcal{M} : F_{x,h}(v) = x'\}$ is the set of velocity in cotangent space at x such that the Hamiltonian ODE with step size h sends (x, v) to (x', v'). As $c(x) = 0$ for $x \in \mathcal{M}$, it follows that

$$\pi(x)\mathbf{P}_x(x') = C_1C_2 \int_{V_x} \frac{e^{-f(x) - \frac{1}{2}\log \mathrm{pdet}(M(x)) - \frac{1}{2}v^\top M(x)^\dagger v - \lambda(x,v)^\top c(x)}}{|DF_{x,h}(v)|}dv = C_1C_2 \int_{V_x} \frac{e^{-H(x,v)}}{|DF_{x,h}(v)|}dv.$$

Going forward, we use three important properties of the Hamiltonian dynamic including reversibility, Hamiltonian preservation, and volume preservation, which still hold for the constrained Hamiltonian H. Due to reversibility $T_{-h}(x', v') = (x, v)$, we can write

$$\pi(x')\mathbf{P}_{x'}(x) = C_1C_2 \int_{V_{x'}} \frac{e^{-H(x',v')}}{|DF_{x',-h}(v')|}dv',$$

where $V_{x'} = \{v' \in \mathcal{T}_{x'}\mathcal{M} : F_{x',-h}(v') = x\}$ is the counterpart of V_x. From reversibility $T_{-h} \circ T_h = I$, the inverse function theorem implies $DT_{-h} = (DT_h)^{-1}$. Now let us denote

$$DT_h(x,v) = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \quad \& \quad DT_{-h}(x',v') = \begin{bmatrix} A' & B' \\ C' & D' \end{bmatrix},$$

where each entry is a block matrix with the same size. Note that $DF_{x,h}(v) = B$ and $DF_{x',-h}(v') = B'$ hold by the definition of Jacobian. Together with $DT_{-h} = (DT_h)^{-1}$, a formula for the inverse of a block matrix results in

$$|DF_{x',-h}(v')| = |B'| = \frac{|B|}{|D||A - BD^{-1}C|} = \frac{|B|}{|DT_h(x,v)|} = |B| = |DF_{x,h}(v)|,$$

where we use the property of volume preservation in the fourth equality (i.e., $|DT_h(x,v)| = 1$). Finally, the property of Hamiltonian preservation implies $H(x,v) = H(x',v')$ and thus

$$\int_{V_{x'}} \frac{e^{-H(x',v')}}{|DF_{x',-h}(v')|} dv' = \int_{V_x} \frac{e^{-H(x,v)}}{|DF_{x,h}(v)|} dv.$$

Therefore, $\pi(x)\mathbf{P}_x(x') = \pi(x')\mathbf{P}_{x'}(x)$ holds. □

Similar reasoning in Theorem 3 in [3] leads to π-irreducibility of this Markov chain, so CRHMC converges to the unique stationary distribution $\pi \propto e^{-f(x)}$.

2.4 Condition Number Independence via Self-concordant Barrier

Choice of $M(x)$. In the previous section, we promised a Hamiltonian for sampling $e^{-f(x)}$ subjects to the constraint $c(x) = 0$. The construction relies on having a family of positive semi-definite matrix $M(x)$ satisfying the condition (2.4) (i.e., $\mathrm{Range}(M(x)) = \mathrm{Null}(Dc(x))$). One natural choice is the orthogonal projection to $\mathrm{Null}(Dc(x))$:

$$Q(x) = I - Dc(x)^\top (Dc(x)Dc(x)^\top)^{-1} Dc(x). \tag{2.6}$$

In this case, our algorithm is similar to [3].

For the problem we care about, there are often extra constraints on x. For concreteness, let us discuss a simple constraint $K = \{x \geq 0\}$. In the standard HMC algorithm, we note that $\frac{dx}{dt} \sim \mathcal{N}(0, M(x)^{-1})$. To ensure every direction is moving towards/away from $x = 0$ multiplicative, a natural choice of M is $M(x) = \mathrm{diag}(x^{-2})$. To generalize this to any convex set K, we make use of some convex function $\phi(x)$, called a *barrier function*, defined on K such that $\phi(x) \to +\infty$ as $x \to \partial K$. Using the barrier function ϕ, we can define the corresponding family of positive semi-definite matrices $g(x) = \nabla^2 \phi(x)$. For the constrained case, we define $M(x)$ to be some matrix such that its range matches the null space of $Dc(x)$ and $M(x)$ agrees with $g(x)$ on the range. Formally, $M(x)$ should be set to fulfill

$$\begin{aligned} u^\top M(x) u &= u^\top g(x) u \text{ for all } u \in \mathrm{Range}(M(x)), \\ \mathrm{Range}(M(x)) &= \mathrm{Null}(Dc(x)). \end{aligned}$$

Solving this equation, we have

$$M(x) = Q(x)^\top \cdot g(x) \cdot Q(x) \tag{2.7}$$

where $Q(x)$ is the symmetric matrix defined in (2.6).

Independence of Condition Number. We are able to show that sampling using M defined in (2.7) is independent of the condition number of the distribution. First, [27] shows that for a Hessian manifold, the distribution satisfies a Poincaré constant given by the ratio between the Hilbert distance and geodesic distance. Then, for a self-concordance barrier, [32] (see Lemma 3) shows that the ratio between the Hilbert distance and geodesic distance can be bounded by the self-concordance parameter. Finally, by [8], our process converges in chi-square distance with an exponential convergence rate that only depends on the Poincaré constant.

2.5 Efficient Computation of $\partial H/\partial x$ and $\partial H/\partial v$

With $M(x) = Q(x) \cdot g(x) \cdot Q(x)$, now we have the complete description of the algorithm and should be able to efficiently compute $\partial H/\partial x$ and $\partial H/\partial v$ to solve the Hamiltonian ODE. However, when using the algorithm as it is, we face several challenges. First of all, it involves computing the pseudo-inverse and its derivatives, which takes $O(n^3)$ except for very special matrices. Next, the Lagrangian term in the constrained Hamiltonian dynamic requires additional computation such as the second-order derivative of $c(x)$. Lastly, a naive approach to computing leverage scores in $\partial H/\partial x$ results in a very dense matrix.

We address each of the challenges as follows. In Section 2.5.1, we present alternative formulas for pseudo-inverse and pseudo-determinant that can leverage existing sparse linear system solvers. Then we show in Section 2.5.2 that the Lagrangian term can be dropped without affecting the dynamic on x at least when $c(x) = Ax - b$. Finally in Section 2.5.3, we discuss how to obtain leverage scores in an efficient manner by computing only small portion of an inverse matrix and tracking the sparsity pattern of matrices.

2.5.1 Avoiding pseudo-inverse and pseudo-determinant

We give equivalent formulas for $M(x)^\dagger$ and $\log \text{pdet} M(x)$ that can take advantage of sparse linear system solvers. We note that existing sparse linear system solvers can solve large classes of sparse linear system much faster than $O(n^3)$ time [11]. The equivalent formulas we provide avoid the expensive pseudo-inverse and pseudo-determinant computations, and are one crucial reason of our improved practical performance.

We start with a formula for $M(x)^\dagger$.

Lemma 3. *Let $M(x) = Q(x) \cdot g(x) \cdot Q(x)$ where $Q(x) = I - Dc(x)^\top (Dc(x) \cdot Dc(x)^\top)^{-1} Dc(x)$ is the orthogonal projection to the null space of $Dc(x)$. Then, $Dc(x) \cdot M(x)^\dagger = 0$ and $M(x)^\dagger = g(x)^{-\frac{1}{2}} \cdot P(x) \cdot g(x)^{-\frac{1}{2}}$ with*

$$P(x) = I - g(x)^{-\frac{1}{2}} \cdot Dc(x)^\top (Dc(x) \cdot g(x)^{-1} \cdot Dc(x)^\top)^{-1} Dc(x) \cdot g(x)^{-\frac{1}{2}}.$$

Remark. As mentioned earlier, a majority of convex sets appearing in practice are of the form $K = \prod_i K_i$, where K_i are constant dimensional convex sets. In this case, we will choose $g(x)$ to be a block diagonal matrix with each block of size $O(1)$. Hence, the bottleneck of applying $P(x)$ to a vector is simply solving a linear system of the form $(Dc \cdot g^{-1} \cdot Dc^\top)u = b$ for some b.

Proof. Recall that $\text{Range}(M(x)^\dagger) = \text{Range}(M(x))$. Hence, for any $u \in \mathbb{R}^n$, we have that $M(x)^\dagger u \in \text{Range}(M(x))$. Since $\text{Range}(M(x)) \subseteq \text{Range}(Q(x))$ and $\text{Range}(Q(x)) = \text{Null}(Dc(x))$ due to the definition of the orthogonal projection $Q(x)$, it follows that $Dc(x) \cdot M(x)^\dagger u = 0$ for all u.

For the formula of $M(x)^\dagger$, we simplify the notation by ignoring the parameter x. Let $N = g^{-\frac{1}{2}} P g^{-\frac{1}{2}}$ and $J = Dc(x)$. The goal is to prove that $M^\dagger = N$. First, we show some basic identities about Q and N:

$$\begin{aligned} QN =& Qg^{-\frac{1}{2}}(I - g^{-\frac{1}{2}} J^\top (Jg^{-1}J^\top)^{-1} J g^{-\frac{1}{2}}) g^{-\frac{1}{2}} \\ =& (I - J^\top (JJ^\top)^{-1} J)(g^{-1} - g^{-1} J^\top (Jg^{-1}J^\top)^{-1} J g^{-1}) \\ =& g^{-1} - J^\top (JJ^\top)^{-1} J g^{-1} \\ & - (g^{-1} J^\top (Jg^{-1}J^\top)^{-1} J g^{-1} - J^\top (JJ^\top)^{-1} J g^{-1} J^\top (Jg^{-1}J^\top)^{-1} J g^{-1}) \\ =& N \end{aligned} \tag{2.8}$$

Similarly, we have $NQ = N$, $QgN = Q$, and $NgQ = Q$. To prove that $M^\dagger = N$, we need to check that MN and NM are symmetric, $MNM = N$, and $NMN = N$.

For symmetry of MN and NM, we note that $MN = QgQN = QgN = Q$ and $NM = NQgQ = NgQ = Q$. For the formula of MNM and NMN, we note that that Q is a projection matrix and hence

$$\begin{aligned} MNM &= QM = QQgQ = QgQ = M, \\ NMN &= QN = N. \end{aligned}$$

Therefore, we have $M^\dagger = N$. ◻

Another bottleneck of the algorithm is to compute $\log \text{pdet} M(x)$. The next lemma shows a simpler formula that can take advantage of sparse Cholesky decomposition.

Lemma 4. *We have that*

$$\log \operatorname{pdet}(M(x)) = \log\det g(x) + \log\det\left(Dc(x)\cdot g(x)^{-1}\cdot Dc(x)^\top\right) - \log\det\left(Dc(x)\cdot Dc(x)^\top\right).$$

Proof. We simplify the notation by ignoring the parameter x and letting $J = Dc(x)$. Let

$$\begin{aligned} f_1(g) &= \log\operatorname{pdet}(Q\cdot g\cdot Q)\\ f_2(g) &= \log\det g + \log\det Jg^{-1}J^\top - \log\det JJ^\top \end{aligned}$$

Clearly, $f_1(I) = f_2(I) = 0$, and hence it suffices to prove that their derivatives are the same.

Note that $\text{Range}(Q\cdot g\cdot Q) = \text{Null}(J)$ and $\text{Range}(J^\top)$ is the orthogonal complement of $\text{Null}(J)$. Since $J^\top(JJ^\top)^{-1}J$ is the orthogonal projection to $\text{Range}(J^\top)$, all of its eigenvectors in $\text{Range}(J^\top)$ have eigenvalue 1 and all the rest in $\text{Null}(J)$ have eigenvalue 0. Therefore, by padding eigenvalue 1 on $\text{Range}(J^\top) = \text{Null}(J)^\perp = \text{Range}(QgQ)^\perp$, we have

$$\begin{aligned} \operatorname{pdet}(Q\cdot g\cdot Q) &= \det(Q\cdot g\cdot Q + J^\top(JJ^\top)^{-1}J)\\ &= \det(Q\cdot g\cdot Q + (I-Q)) \end{aligned}$$

Using $D\log\det A(g)[u] = \text{Tr}(A(g)^{-1}DA(g)[u])$, the directional derivative of f_1 on direction u is

$$Df_1(g)[u] = \text{Tr}\big((Q\cdot g\cdot Q + (I-Q))^{-1}Q\cdot u\cdot Q\big)$$

Let $N = (Q\cdot g\cdot Q)^\dagger$. As shown in the proof of Lemma 3, we have $NQ = QN = N$ and $QgN = Q$. By using these identities, we can manually check that $(Q\cdot g\cdot Q + (I-Q))^{-1} = N + (I-Q)$. Hence,

$$\begin{aligned} Df_1(g)[u] &= \text{Tr}\left((N + (I-Q))Q\cdot u\cdot Q\right) = \text{Tr}(NuQ)\\ &= \text{Tr}(QNu) = \text{Tr}(Nu) \end{aligned}$$

where we used idempotence of the projection matrix Q (i.e., $Q^2 = Q$).

On the other hand, we have

$$\begin{aligned} Df_2(g)[u] &= \text{Tr}(g^{-1}u) - \text{Tr}\left((Jg^{-1}J^\top)^{-1}(Jg^{-1}ug^{-1}J^\top)\right)\\ &= \text{Tr}\left((g^{-1} - g^{-1}J^\top(Jg^{-1}J^\top)^{-1}Jg^{-1})u\right)\\ &= \text{Tr}(Nu) \end{aligned}$$

where we used the alternative formula of N in Lemma 3. This shows that the derivative of f_1 equals to that of f_2 at any point $g \succ 0$. Since the set of positive definite matrices is connected and $f_1(I) = f_2(I)$, this implies that $f_1(g) = f_2(g)$ for all $g \succ 0$. □

Combining Lemma 3 and Lemma 4, we have the following formula of the Hamiltonian.

$$\begin{aligned} H(x,v) =& \overline{H}(x,v) + \lambda(x,v)^\top c(x)\\ \overline{H}(x,v) =& f(x) + \frac{1}{2}v^\top g(x)^{-\frac{1}{2}}\left(I - g(x)^{-\frac{1}{2}}\cdot Dc(x)^\top(Dc(x)\cdot g(x)^{-1}\cdot Dc(x)^\top)^{-1}Dc(x)\cdot g(x)^{-\frac{1}{2}}\right)g(x)^{-\frac{1}{2}}v\\ &+ \frac{1}{2}\left(\log\det g(x) + \log\det\left(Dc(x)\cdot g(x)^{-1}\cdot Dc(x)^\top\right) - \log\det\left(Dc(x)\cdot Dc(x)^\top\right)\right) \end{aligned}$$

2.5.2 Simplification for subspace constraints

For the case $c(x) = Ax - b$, the constrained Hamiltonian is

$$H(x,v) = f(x) + \frac{1}{2}v^\top g^{-\frac{1}{2}}(I-P)g^{-\frac{1}{2}}v + \frac{1}{2}\left(\log\det g + \log\det Ag^{-1}A^\top - \log\det AA^\top\right) + \lambda^\top c \qquad (2.9)$$

where $P = g^{-\frac{1}{2}}A^\top(Ag^{-1}A^\top)^{-1}Ag^{-\frac{1}{2}}$. The following lemma shows that the dynamic corresponding to H above is equivalent to a simpler H. The key observation is that the algorithm only needs to know $x(h)$ in the HMC dynamic, and not $v(h)$. Thus we can replace H by any other H that produces the same $x(h)$.

Lemma 5. *The Hamiltonian dynamic on x corresponding to (2.9) is same as the dynamic on x corresponding to*

$$H(x,v) = f(x) + \frac{1}{2}v^\top g^{-\frac{1}{2}}\left(I-P\right)g^{-\frac{1}{2}}v + \frac{1}{2}\left(\log\det g + \log\det Ag^{-1}A^\top\right) \tag{2.10}$$

where $P = g^{-\frac{1}{2}}A^\top(Ag^{-1}A^\top)^{-1}Ag^{-\frac{1}{2}}$. Furthermore, we have

$$\frac{dx}{dt} = g^{-\frac{1}{2}}\left(I-P\right)g^{-\frac{1}{2}}v, \tag{2.11}$$

$$\frac{dv}{dt} = -\nabla f(x) + \frac{1}{2}Dg\left[\frac{dx}{dt},\frac{dx}{dt}\right] - \frac{1}{2}\mathrm{Tr}(g^{-\frac{1}{2}}\left(I-P\right)g^{-\frac{1}{2}}Dg). \tag{2.12}$$

Proof. Note that the dynamic on x corresponding to (2.9) is given by

$$\begin{aligned}\frac{dx}{dt} = \frac{\partial H}{\partial v} &= g^{-\frac{1}{2}}\left(I-P\right)g^{-\frac{1}{2}}v + (D_v\lambda)^\top c\\ &= g^{-\frac{1}{2}}\left(I-P\right)g^{-\frac{1}{2}}v\end{aligned} \tag{2.13}$$

where we used that $c(x)=0$ (Lemma 1).

Now let us compute the dynamic on v. Note that

$$v^\top g^{-\frac{1}{2}}\left(I-P\right)g^{-\frac{1}{2}}v = v^\top g^{-1}v - v^\top g^{-1}A^\top(A\cdot g^{-1}\cdot A^\top)^{-1}Ag^{-1}v.$$

Hence, we have

$$\begin{aligned}D_x\left(\frac{1}{2}v^\top g^{-\frac{1}{2}}\left(I-P\right)g^{-\frac{1}{2}}v\right) =& -\frac{1}{2}v^\top g^{-1}\cdot Dg\cdot g^{-1}v + v^\top g^{-1}\cdot Dg\cdot g^{-1}A^\top(A\cdot g^{-1}\cdot A^\top)^{-1}Ag^{-1}v\\ &-\frac{1}{2}v^\top g^{-1}A^\top(A\cdot g^{-1}\cdot A^\top)^{-1}A\cdot g^{-1}\cdot Dg\cdot g^{-1}\cdot A^\top(A\cdot g^{-1}\cdot A^\top)^{-1}Ag^{-1}v\\ =& -\frac{1}{2}v^\top g^{-\frac{1}{2}}(I-P)g^{-\frac{1}{2}}\cdot Dg\cdot g^{-\frac{1}{2}}(I-P)g^{-\frac{1}{2}}v\\ =& -\frac{1}{2}Dg\left[\frac{dx}{dt},\frac{dx}{dt}\right],\end{aligned}$$

where we used $\frac{dx}{dt} = g^{-\frac{1}{2}}(I-P)g^{-\frac{1}{2}}v$ in (2.13). Therefore, it follows that

$$\begin{aligned}\frac{dv}{dt} &= -\frac{\partial\overline{H}}{\partial x} - (D_v\lambda)^\top c - A^\top\lambda \\ &= -\nabla f(x) + \frac{1}{2}Dg\left[\frac{dx}{dt},\frac{dx}{dt}\right] - \frac{1}{2}\mathrm{Tr}(g^{-1}Dg) + \frac{1}{2}\mathrm{Tr}\left((Ag(x)^{-1}A^\top)^{-1}Ag(x)^{-1}\cdot Dg\cdot g(x)^{-1}A^\top\right) - A^\top\lambda\\ &= -\nabla f(x) + \frac{1}{2}Dg\left[\frac{dx}{dt},\frac{dx}{dt}\right] - \frac{1}{2}\mathrm{Tr}(g^{-\frac{1}{2}}\left(I-P\right)g^{-\frac{1}{2}}Dg) - A^\top\lambda\end{aligned} \tag{2.14}$$

where we used that $c=0$ again in the second equality.

Recall that $\frac{dx}{dt} = g^{-\frac{1}{2}}\left(I-P\right)g^{-\frac{1}{2}}v$. In this formula, let us perturb v by $A^\top y$ for any y as follows.

$$\begin{aligned}\left(I-P\right)g^{-\frac{1}{2}}(v+A^\top y) &= \left(I-P\right)g^{-\frac{1}{2}}v + \left(I - g^{-\frac{1}{2}}A^\top(Ag^{-1}A^\top)^{-1}Ag^{-\frac{1}{2}}\right)g^{-\frac{1}{2}}A^\top y\\ &= \left(I-P\right)g^{-\frac{1}{2}}v + (g^{-\frac{1}{2}}A^\top y - g^{-\frac{1}{2}}A^\top(Ag^{-1}A^\top)^{-1}(Ag^{-1}A^\top)y)\\ &= \left(I-P\right)g^{-\frac{1}{2}}v + (g^{-\frac{1}{2}}A^\top y - g^{-\frac{1}{2}}A^\top y)\\ &= \left(I-P\right)g^{-\frac{1}{2}}v.\end{aligned}$$

Hence, removing $A^\top\lambda$ from $\frac{dv}{dt}$ in (2.14) does not change the dynamic on x, and thus we have the new dynamic given simply by (2.11) and (2.12). By repeating this proof, one can check that the simplified Hamiltonian (2.10) also yields (2.11) and (2.12). $\square$

2.5.3 Efficient Computation of Leverage Score

As mentioned earlier, each step for solving the Hamiltonian ODE requires computation of leverage scores. Even after simplifying the Hamiltonian in Section 2.5.2, we still have a term for the leverage scores, $\mathrm{Tr}(g^{-\frac{1}{2}}(I-P)g^{-\frac{1}{2}}Dg)$ in $\frac{dv}{dt}$, and the diagonal entries of $P = g^{-\frac{1}{2}}A^\top(Ag^{-1}A^\top)^{-1}Ag^{-\frac{1}{2}}$ should be realized to compute $\frac{dv}{dt}$. Since $(Ag^{-1}A^\top)^{-1}$ can be extremely dense even when A is very sparse, a naive approach such as direct computation of the inverse leads to a dense matrix and dense-matrix multiplication.

In this section, we discuss how we efficiently compute the diagonal entries of $A^\top(Ag^{-1}A^\top)^{-1}A$. Our idea is based on the fact that certain entries of $(Ag^{-1}A^\top)^{-1}$ can be computed as fast as computing sparse Cholesky decomposition of $Ag^{-1}A^\top$ [42, 4], which can be $O(n)$ time faster than computing $(Ag^{-1}A^\top)^{-1}$ in many settings.

For simplicity, we focus on the case $g(x)$ as a diagonal matrix, since we use the log-barrier $\phi(x) = -\sum_{i=1}^m(\log(x_i - l_i) + \log(u_i - x_i))$ in implementation. We first note that we maintain a "sparsity pattern" $\mathrm{sp}(M)$ of a sparse matrix M so that we handle only these entries in downstream tasks. The sparsity pattern indicates "candidates" of nonzero entries of a matrix (i.e., $\mathrm{sp}(M) \supseteq nnz(M) = \{(i,j) : M_{ij} \neq 0\}$). For instance, it is obvious that $\mathrm{sp}(cc^\top) = \{(i,j) : c_ic_j \neq 0\} = nnz(cc^\top)$ for a column vector c and that $\mathrm{sp}(Ag^{-1}A^\top) = \bigcup_{i\in[n]} \mathrm{sp}(A_iA_i^\top)$ follows from the equality $Ag^{-1}A^\top = \sum_{i=1}^n (Ag^{-\frac{1}{2}})_i(Ag^{-\frac{1}{2}})_i^\top$, where M_i denote the i^{th} column of M (See Theorem 2.1 in [10]). Then we compute the Cholesky decomposition to obtain a sparse triangular matrix L such that $LL^\top = Ag^{-1}A^\top$ with a property $\mathrm{sp}(Ag^{-1}A^\top) \subseteq \mathrm{sp}(L^\top) \cup \mathrm{sp}(L)$ (See Theorem 4.2 in [10]).

Once the sparsity pattern of L is identified, we compute $S := (Ag^{-1}A^\top)^{-1}|_{\mathrm{sp}(L)}$, the restriction of S to $\mathrm{sp}(L)$, that is, the inverse matrix S is computed only for entries in $\mathrm{sp}(L)$. [42, 4] showed that this matrix S can be computed as fast as the Cholesky decomposition of $Ag^{-1}A^\top$.

For completeness, we explain how they compute S efficiently. Let $L_0DL_0^\top$ be the LDL decomposition of $Ag^{-1}A^\top$ such that the diagonals of L_0 is one and so $L = L_0D^{\frac{1}{2}}$, and it easily follows that

$$S = D^{-1}L_0^{-1} + (I - L_0^\top)S = D^{-\frac{1}{2}}L^{-1} + (I - L^\top D^{-\frac{1}{2}})^{-1}S.$$

Since $D^{-1}L_0^{-1}$ is lower triangular and $I - L_0^\top$ is strictly upper triangular, symmetry of S implies that S can be computed from the bottom row to the top row one by one. We note that the computation of S on any entry in $\mathrm{sp}(L)$ only requires previously computed S on entries in $\mathrm{sp}(L)$, due to the sparsity pattern of $I - L^\top D^{-\frac{1}{2}}$. [42, 4] showed that the total cost of computing S is $O(\sum_{i=1}^n n_i^2)$ for backward substitution, where n_i is the number of nonzeros in the i^{th} column of L. This exactly matches the cost of computing L. In our experiments, for many sparse matrices A, we found that $O(\sum_{i=1}^n n_i^2)$ is roughly $O(n^{1.5})$ and it is much faster than dense matrix inverse.

We have presented methods to save computational cost, avoiding full computation of the inverse $(Ag^{-1}A^\top)^{-1}$. This attempt is justified by the fact that only entries of $Ag^{-1}A^\top$ in $\mathrm{sp}(L) \cup \mathrm{sp}(L^\top)$ matter in computing $\mathrm{diag}(A^\top SA) = \mathrm{diag}(A^\top(Ag^{-1}A^\top)^{-1}A)$.

Lemma 6. *Computation of* $\mathrm{diag}(A^\top(Ag^{-1}A^\top)^{-1}A)$ *involves accessing only entries of* $(Ag^{-1}A^\top)^{-1}$ *in* $sp(Ag^{-1}A^\top)$.

Proof. Let $M := (Ag^{-1}A^\top)^{-1} \in \mathbb{R}^{m\times m}$, $\sigma_i := (A^\top(Ag^{-1}A^\top)^{-1}A)_{ii}$ for $i \in [n]$, and a_i be the i^{th} column of A. Observe that

$$\sigma_i = a_i^\top(Ag^{-1}A^\top)^{-1}a_i = \mathrm{Tr}(a_i^\top Ma_i) = \mathrm{Tr}(Ma_ia_i^\top).$$

As the entries of M only in $\mathrm{sp}(a_ia_i^\top)$ matter when computing the trace, we have that all the entries of M used for computing σ_i for all $i \in [n]$ are included in $\bigcup_{i=1}^n \mathrm{sp}(a_ia_i^\top) = \mathrm{sp}(Ag^{-1}A^\top)$. $\square$

Now let us divide the diagonals of S by 2. Then we have $(Ag^{-1}A^\top)^{-1}|_{\mathrm{sp}(L)\cup\mathrm{sp}(L^\top)} = S + S^\top$ and thus

$$\mathrm{diag}(A^\top(Ag^{-1}A^\top)^{-1}A) = \mathrm{diag}(A^\top(Ag^{-1}A^\top)^{-1}|_{\mathrm{sp}(L)\cup\mathrm{sp}(L^\top)}A) = \mathrm{diag}(A^\top SA + A^\top S^\top A) = 2\cdot\mathrm{diag}(A^\top SA)$$

and the last term can be computed efficiently using S. In our experiment, the cost of computing leverage score is roughly twice the cost of computing Cholesky decomposition in all datasets.

Finally, we discuss another approach to compute leverage score with the same asymptotic complexity. We consider the function

$$V(g) = \log\det A g^{-1} A^\top$$

where g is a sparse matrix $g \in \mathbb{R}^{\mathrm{sp}(g)}$ and V is defined only on $\mathbb{R}^{\mathrm{sp}(g)}$. Note that $V(g)$ can be computed using Cholesky decomposition of $A^\top g^{-1} A^\top$ and multiplying the diagonal of the decomposition. Next, we note that

$$\nabla V(g) = -(g^{-1}A^\top(Ag^{-1}A^\top)^{-1}Ag^{-1})|_{\mathrm{sp}(g)}.$$

Hence, we can compute leverage score by first computing $\nabla V(g)$ via automatic differentiation, and the time complexity of computing ∇V is only a small constant factor more than the time complexity of computing V [16]. The only problem with this approach is that the Cholesky decomposition algorithm is an algorithm involving a large loop and sparse operations and existing automatic differentiation packages are not efficient to differentiate such functions.

2.6 Discretization

We discuss how to implement our Hamiltonian dynamic using the implicit midpoint method in Section 2.6.1 and present theoretical guarantees of correctness and efficiency of the discretized CRHMC in Section 2.6.2.

2.6.1 Discretized CRHMC based on Implicit Midpoint Integrator

In our algorithm, we discretize the Hamiltonian process into steps of step size h and run the process for T iterations (see Algorithm 2). Starting from $(x^{(0)}, v^{(0)})$, let $(x^{(t)}, v^{(t)})$ be the point obtained after iteration t. In the beginning of each iteration, we compute the Cholesky decomposition of $Ag(x)^{-1}A^\top$ for later use and resample the velocity with momentum. As noted previously in Lemma 5, for $c(x) = Ax - b$ we can just use the simplified Hamiltonian in (2.10)

$$H(x,v) = f(x) + \frac{1}{2}v^\top g(x)^{-\frac{1}{2}}\left(I - P(x)\right)g(x)^{-\frac{1}{2}}v + \frac{1}{2}\left(\log\det g(x) + \log\det Ag(x)^{-1}A^\top\right)$$

instead of the constrained Hamiltonian $\overline{H} + \lambda^\top c$. We solve the Hamiltonian dynamic for H by the implicit midpoint method, which we will discuss below, and then use a Metropolis filter on H to ensure the distribution is correct.

Implicit Midpoint Method. We elaborate on how the implicit midpoint integrator works (see Algorithm 3), which is known to be symplectic (so measure-preserving) and reversible. Let us write $H(x,v) = \overline{H}_1(x,v) + \overline{H}_2(x,v)$, where

$$\overline{H}_1(x,v) = f(x) + \frac{1}{2}\left(\log\det g(x) + \log\det Ag(x)^{-1}A^\top\right)$$
$$\overline{H}_2(x,v) = \frac{1}{2}v^\top g(x)^{-\frac{1}{2}}\left(I - P(x)\right)g(x)^{-\frac{1}{2}}v$$

Starting from (x_0, v_0), in the first step of the integrator, we run the process on the Hamiltonian $\overline{H}_1$ with step size $\frac{h}{2}$ to get $(x_{1/3}, v_{1/3})$, and this discretization leads to $x_{1/3} = x_0 + \frac{h}{2}\frac{\partial \overline{H}_1}{\partial v}(x_0, v_0)$ and $v_{1/3} = v_0 - \frac{h}{2}\frac{\partial \overline{H}_1}{\partial x}(x_0, v_0)$. Note that $x_{1/3} = x_0$ due to $\frac{\partial \overline{H}_1}{\partial v} = 0$. In the second step of the integrator, we run the process on $\overline{H}_2$ with step size h by solving

$$x_{\frac{2}{3}} = x_{\frac{1}{3}} + h\frac{\partial \overline{H}_2}{\partial v}\left(\frac{x_{\frac{1}{3}} + x_{\frac{2}{3}}}{2}, \frac{v_{\frac{1}{3}} + v_{\frac{2}{3}}}{2}\right)$$
$$v_{\frac{2}{3}} = v_{\frac{1}{3}} - h\frac{\partial \overline{H}_2}{\partial x}\left(\frac{x_{\frac{1}{3}} + x_{\frac{2}{3}}}{2}, \frac{v_{\frac{1}{3}} + v_{\frac{2}{3}}}{2}\right)$$

To this end, starting from $x_{2/3} = x_{1/3}$ and $v_{2/3} = v_{1/3}$, we apply $x_{2/3} \leftarrow x_{1/3} + h\frac{\partial \overline{H}_2}{\partial v}\left(\frac{x_{1/3}+x_{2/3}}{2}, \frac{v_{1/3}+v_{2/3}}{2}\right)$ and $v_{2/3} \leftarrow v_{1/3} - h\frac{\partial \overline{H}_2}{\partial x}\left(\frac{x_{1/3}+x_{2/3}}{2}, \frac{v_{1/3}+v_{2/3}}{2}\right)$ iteratively with the following subroutine for computing $\frac{\partial \overline{H}_2}{\partial v}$

and $\frac{\partial \overline{H}_2}{\partial x}$. According to Lemma 5, this computation involves solving $g(x)^{-1}A^\top \left(Ag(x)^{-1}A^\top\right)^{-1} Ag(x)^{-1}v$ for some v and x. To compute $\left(Ag(x)^{-1}A^\top\right)^{-1} Ag(x)^{-1}v$, we use the Newton's method, which iteratively computes $\nu \leftarrow \nu + M^{-1}Ag(x)^{-1}\left(v - A^\top \nu\right)$ for some M. Note that the Newton's method guarantees that ν converges to $M^{-1}Ag(x)^{-1}v$ if M is invertible. Here, we choose $M = Ag(x^{(t)})^{-1}A^\top$ to ensure fast convergence. Since we have already computed the Cholesky decomposition of M in the beginning, $M^{-1}Ag(x)^{-1}\left(v - A^\top \nu\right)$ can be computed efficiently by backward and forward substitution. In the third step of the integrator, we run the process on the Hamiltonian $\overline{H}_1$ with step size $\frac{h}{2}$ again to get (x_1, v_1), which results in $x_1 = x_{2/3}$ and $v_1 = v_{2/3} - \frac{h}{2}\frac{\partial \overline{H}_1}{\partial x}(x_1)$.

Putting Algorithm 2 and Algorithm 3 together, we obtain discretization of constrained Riemannian Hamiltonian Monte Carlo algorithm.

Algorithm 2: `Discretized Constrained Riemannian Hamiltonian Monte Carlo with Momentum`

```
Input: Initial point $x^{(0)}$, velocity $v^{(0)}$, record frequency $T$, step size $h$, ODE steps $K$
for $t = 1, 2, \cdots, T$ do
    Let $\overline{v} = v^{(t-1)}$ and $x = x^{(t-1)}$.
    // Step 1: Resample $v$ with momentum
    Let $z \sim \mathcal{N}(0, M(x))$. Update $\overline{v}$:
        $$v \leftarrow \sqrt{\beta}\overline{v} + \sqrt{1-\beta}z$$
    // Step 2: Solve $\frac{dx}{dt} = \frac{\partial H(x,v)}{\partial v}, \frac{dv}{dt} = -\frac{\partial H(x,v)}{\partial x}$ via the implicit midpoint method
    Use Implicit Midpoint Method$(x, v, h, K)$ to find $(x', v')$ such that
        $$\begin{aligned} v_{\frac{1}{3}} &= v - \frac{h}{2}\frac{\partial \overline{H}_1(x,v)}{\partial x} \\ x' &= x + h\frac{\partial \overline{H}_2(\frac{x+x'}{2}, \frac{v_{1/3}+v_{2/3}}{2})}{\partial v}, \quad v' = v - h\frac{\partial \overline{H}_2(\frac{x+x'}{2}, \frac{v_{1/3}+v_{2/3}}{2})}{\partial x} \\ v' &= v_{\frac{2}{3}} - \frac{h}{2}\frac{\partial \overline{H}_1(x',v')}{\partial x} \end{aligned} \quad (2.15)$$
    // Step 3: Filter
    With probability $\min\left\{1, \frac{e^{-H(x',v')}}{e^{-H(x,v)}}\right\}$, set $x^{(t)} \leftarrow x'$ and $v^{(t)} \leftarrow v'$.
    Otherwise, set $x^{(t)} \leftarrow x$ and $v^{(t)} \leftarrow -v$.
end
Output: $x^{(T)}$
```

2.6.2 Theoretical Guarantees

We first show that one iteration of Algorithm 2 incurs the cost of solving a few Cholesky decomposition and $O(K)$ sparse triangular systems. Then we show in Theorem 8 that when Algorithm 3 converges, Algorithm 2 has a stationary density proportional to $\exp(-f(x))$. In Lemma 11, we show Algorithm 3 converges fast.

Theorem 7. *The cost of each iteration of Algorithm 2 is solving $O(1)$ Cholesky decomposition and $O(K)$ triangular systems, where K is the number of iterations in Algorithm 3.*

Proof. We first solve the Cholesky decomposition to get $L_{t-1}L_{t-1}^\top = Ag(x^{(t-1)})^{-1}A^\top$ at the beginning of

```
Algorithm 3: Implicit Midpoint Method
  Input: Initial point $x$, velocity $v$, step size $h$, ODE steps $K$
  // Step 1:  Solve $\frac{dx}{dt}=\frac{\partial\overline{H}_1(x,v)}{\partial v}$, $\frac{dv}{dt}=-\frac{\partial\overline{H}_1(x,v)}{\partial x}$
  Set $x_{\frac{1}{3}}\leftarrow x$ and $v_{\frac{1}{3}}\leftarrow v-\frac{h}{2}\frac{\partial\overline{H}_1(x,v)}{\partial x}$.

  // Step 2:  Solve $\frac{dx}{dt}=\frac{\partial\overline{H}_2(x,v)}{\partial v}$, $\frac{dv}{dt}=-\frac{\partial\overline{H}_2(x,v)}{\partial x}$ via implicit midpoint
  Set $\nu\leftarrow 0$.
  for $k=1,2,\cdots,K$ do
      Let $x_{\mathrm{mid}}\leftarrow\frac{1}{2}\left(x_{\frac{1}{3}}+x_{\frac{2}{3}}\right)$ and $v_{\mathrm{mid}}\leftarrow\frac{1}{2}\left(v_{\frac{1}{3}}+v_{\frac{2}{3}}\right)$
      Set $\nu\leftarrow\nu+\left(LL^\top\right)^{-1}Ag(x_{\mathrm{mid}})^{-1}\left(v_{\mathrm{mid}}-A^\top\nu\right)$
      Set $x_{\frac{2}{3}}\leftarrow x_{\frac{1}{3}}+hg(x_{\mathrm{mid}})^{-1}\left(v_{\mathrm{mid}}-A^\top\nu\right)$
      and $v_{\frac{2}{3}}\leftarrow v_{\frac{1}{3}}-\frac{h}{2}Dg(x_{\mathrm{mid}})\left[g(x_{\mathrm{mid}})^{-1}\left(v_{\mathrm{mid}}-A^\top\nu\right),g(x_{\mathrm{mid}})^{-1}\left(v_{\mathrm{mid}}-A^\top\nu\right)\right]$
  end

  // Step 3:  Solve $\frac{dx}{dt}=\frac{\partial\overline{H}_1(x,v)}{\partial v}$, $\frac{dv}{dt}=-\frac{\partial\overline{H}_1(x,v)}{\partial x}$
  Set $x_1\leftarrow x_{\frac{2}{3}}$ and $v_1\leftarrow v_{\frac{2}{3}}-\frac{h}{2}\frac{\partial\overline{H}_1}{\partial x}(x_{\frac{2}{3}},v_{\frac{2}{3}})$.
  Output: $x_1$,$v_1$
```

iteration. Recall that

$$\begin{aligned}H(x,v)&=\overline{H}_1(x,v)+\overline{H}_2(x,v)\\&=\left(f(x)+\frac{1}{2}(\log\det g(x)+\log\det Ag(x)^{-1}A^\top)\right)\\&\quad+\left(\frac{1}{2}v^\top g(x)^{-\frac{1}{2}}\left(I-g(x)^{-\frac{1}{2}}A^\top(Ag(x)^{-1}A^\top)^{-1}Ag(x)^{-\frac{1}{2}}\right)g(x)^{-\frac{1}{2}}v\right).\end{aligned}$$

The value of $H(x^{(t-1)},v^{(t-1)})$ should be computed later for the filter step and can be efficiently computed by the given $L_{t-1}L_{t-1}^\top=Ag(x^{(t-1)})^{-1}A^\top$and solving two sparse triangular systems (i.e., $L_{t-1}^{-\top}(L_{t-1}^{-1}(Ag(x)^{-\frac{1}{2}})))$. We need the same cost (i.e., Cholesky decomposition and solving two triangular systems) for the value of $H(x',v')$, where (x',v') is the output of Algorithm 3. We note that L inherits sparsity of A and thus each triangular system can be solved efficiently by backward and forward substitution.

In the implicit midpoint method, one main component is computation of $\frac{\partial\overline{H}_1(x,v)}{\partial x}$ in Step 1 and $\frac{\partial\overline{H}_1}{\partial x}(x_{\frac{2}{3}},v_{\frac{2}{3}})$ in Step 3 due to leverage scores. As seen in Section 2.5.3, the cost for these computations is within a constant factor of solving the Cholesky decomposition for $Ag(x^{(t-1)})^{-1}A^\top$ and $Ag(x_{\frac{2}{3}})^{-1}A^\top$. Another component is solving $O(K)$ triangular systems to update ν in Step 2.

Adding up all these costs, each iteration of Algorithm 2 only requires solving $O(1)$ Cholesky decomposition and $O(K)$ sparse triangular systems. □

Theorem 8. *The Markov chain defined by Algorithm 2 projected to x has a stationary density proportional to* $\exp(-f(x))$.

Proof. Each iteration consists of two stages: resampling velocity with momentum in Step 1 (i.e., $(x,\overline{v})$ to (x,v)) and solving ODE followed by the filter in Step 2 and 3 (i.e., (x,v) to (x',v')). To prove the claim, we show that Step 1 is time-reversible with respect to the conditional distribution $\pi(v|x)$ and that Step 2 followed by Step 3 is also time-reversible with respect to $\pi(x,v)$.

We begin with the first part. We have $\pi(\overline{v}|x)=\mathcal{N}(0,M(x))$ due to the definition of H. Since $\overline{v}|x\sim\mathcal{N}(0,M(x))$ and $z\sim\mathcal{N}(0,M(x))$ are independent Gaussians, the update rule $v=\sqrt{\beta}\overline{v}+\sqrt{1-\beta}z$ implies $\pi(v|x)=\mathcal{N}(0,M(x))$. Let $\mathbf{P}(z)$ be the probability density and C be the normalization constant for Gaussian

$\mathcal{N}(0, M(x))$. Then, the time-reversibility w.r.t. $\pi(v|x)$ is immediate from the following computation:

$$\begin{aligned}
\pi(\overline{v}|x)\mathbf{P}(\overline{v} \to v) &= C^2 \exp(-\frac{1}{2}\overline{v}^\top M^\dagger \overline{v}) \cdot \exp(-\frac{1}{2}\frac{(v-\sqrt{\beta}\overline{v})^\top M^\dagger (v-\sqrt{\beta}\overline{v})}{1-\beta}) \\
&= C^2 \exp\left(-\frac{1}{2}\left(\overline{v}^\top M^\dagger \overline{v} + \frac{v^\top M^\dagger v}{1-\beta} + \frac{\beta \overline{v}^\top M^\dagger \overline{v}}{1-\beta} - \frac{\sqrt{\beta}}{1-\beta}(\overline{v}^\top M^\dagger v + v^\top M^\dagger \overline{v})\right)\right) \\
&= C^2 \exp\left(-\frac{1}{2}\left(\frac{v^\top M^\dagger v}{1-\beta} + \frac{\overline{v}^\top M^\dagger \overline{v}}{1-\beta} - \frac{\sqrt{\beta}}{1-\beta}(\overline{v}^\top M^\dagger v + v^\top M^\dagger \overline{v})\right)\right) \\
\pi(v|x)\mathbf{P}(v \to \overline{v}) &= C^2 \exp(-\frac{1}{2}v^\top M^\dagger v) \cdot \exp(-\frac{1}{2}\frac{(\overline{v}-\sqrt{\beta}v)^\top M^\dagger (\overline{v}-\sqrt{\beta}v)}{1-\beta}) \\
&= C^2 \exp\left(-\frac{1}{2}\left(v^\top M^\dagger v + \frac{\overline{v}^\top M^\dagger \overline{v}}{1-\beta} + \frac{\beta v^\top M^\dagger v}{1-\beta} - \frac{\sqrt{\beta}}{1-\beta}(\overline{v}^\top M^\dagger v + v^\top M^\dagger \overline{v})\right)\right) \\
&= C^2 \exp\left(-\frac{1}{2}\left(\frac{v^\top M^\dagger v}{1-\beta} + \frac{\overline{v}^\top M^\dagger \overline{v}}{1-\beta} - \frac{\sqrt{\beta}}{1-\beta}(\overline{v}^\top M^\dagger v + v^\top M^\dagger \overline{v})\right)\right) \\
\implies \quad \pi(\overline{v}|x)\mathbf{P}(\overline{v} \to v) &= \pi(v|x)\mathbf{P}(v \to \overline{v})
\end{aligned}$$

The second part follows from a stronger statement due to symmetry of v in $H(x, v)$: In the space where (x, v) and $(x, -v)$ are identified, the Markov chain defined by Step 2 and 3 satisfies detailed balance with respect the density $\pi([x, v])$ proportional to $\exp(-H(x, v))$, where $[x, v]$ denotes the identified point for (x, v) and $(x, -v)$. Consider the pairs $[x, v] = \{(x, v), (x, -v)\}$ and $[x', v'] = \{(x', v'), (x', -v')\}$ where in Step 2 (x, v) goes to (x', v') and $(x', -v')$ goes to $(x, -v)$ due to reversibility of the implicit midpoint method. We now verify that the filtering probability is the same in either direction, using the measure-preserving property of Step 2

$$\begin{aligned}
\pi(x, v)\mathbf{P}\left((x, v) \to (x', v')\right) &= \pi(x, v) \min\left\{1, \frac{\pi(x', v')}{\pi(x, v)}\right\} \\
&= \min\left\{\pi(x, v), \pi(x', v')\right\} \\
&= \min\left\{\pi(x, -v), \pi(x', -v')\right\} \\
&= \pi(x', -v') \min\left\{1, \frac{\pi(x, -v)}{\pi(x', -v')}\right\} \\
&= \pi(x', -v')\mathbf{P}\left((x', -v') \to (x, -v)\right).
\end{aligned}$$

Therefore, for any two pairs $[x, v]$ and $[x', v']$, we have $\pi([x, v])\mathbf{P}\left([x, v] \to [x', v']\right) = \pi([x', v']\mathbf{P}\left([x', v'] \to [x, v]\right)$. Finally, since the Markov chain defined by Algorithm 2 is irreducible, it has a unique stationary distribution proportional to $\exp(-f(x))$. □

Theorem 8 shows that if Step 2 of Algorithm 2 can be solved exactly, the algorithm will converge to the stationary distribution. Next, we show in Lemma 11 that Algorithm 3 converges to the solution of (2.15) in logarithmically many iterations. Theorem 8 and Lemma 11 together show that our algorithm can converge to the stationary distribution efficiently (see Remark 12). To show the convergence of Algorithm 3, we denote by $\mathcal{T}$ the map induced by one iteration of Step 2.

Definition 9. Let

$$\mathcal{T}(x, v, \nu) = \begin{pmatrix} x_{\frac{1}{3}} + hg(x_{\mathrm{mid}})^{-1}(v_{\mathrm{mid}} - A^\top \lambda_1) \\ v_{\frac{1}{3}} - \frac{h}{2} Dg(x_{\mathrm{mid}})[g(x_{\mathrm{mid}})^{-1}(v_{\mathrm{mid}} - A^\top \lambda_1), g(x_{\mathrm{mid}})^{-1}(v_{\mathrm{mid}} - A^\top \lambda_1)] \\ \lambda_1 \end{pmatrix}$$

where $x_{\mathrm{mid}} = \frac{1}{2}(x_{\frac{1}{3}} + x)$, $v_{\mathrm{mid}} = \frac{1}{2}(v_{\frac{1}{3}} + v)$, and $\lambda_1 = \nu + (LL^\top)^{-1} A g(v_{\mathrm{mid}})^{-1}\left(v_{\mathrm{mid}} - A^\top \nu\right)$. Let $(x^*_{\frac{2}{3}}, v^*_{\frac{2}{3}}, \nu^*)$ be the fixed point of $\mathcal{T}$.

We assume g is given by the Hessian of a *highly self-concordant* barrier ϕ.

Definition 10. We say a barrier ϕ is highly self-concordant if it satisfies for all $h \in \mathbb{R}^d$ and $x \in \mathbb{R}^d$

$$\left|D^3\phi(x)[h,h,h]\right| \le 2\left(D^2\phi(x)[h,h]\right)^{3/2} \quad \text{and} \quad \left|D^4\phi(x)[h,h,h,h]\right| \le 6\left(D^2\phi(x)[h,h]\right)^2$$

Note that the log-barrier is highly self-concordant. We can show that for small enough step size h, Algorithm 3 can solve (2.15) to δ-accuracy in logarithmically many iterations.

Lemma 11. *Suppose $g(x) = \nabla^2\phi(x)$ for some highly self-concordant barrier ϕ. For any input $(x_{\frac{1}{3}}, v_{\frac{1}{3}})$, let $(x_{\frac{2}{3}}^{(k)}, v_{\frac{2}{3}}^{(k)}, \nu^{(k)})$ be points obtained after k iterations in Step 2 of Algorithm 3. Let $(\widetilde{x}_{\frac{2}{3}}, \widetilde{v}_{\frac{2}{3}})$ be the solution for $(x_{\frac{2}{3}}, v_{\frac{2}{3}})$ in the following equation*

$$x_{\frac{2}{3}} = x_{\frac{1}{3}} + h\frac{\partial \overline{H}_2}{\partial v}\left(\frac{x_{\frac{1}{3}} + x_{\frac{2}{3}}}{2}, \frac{v_{\frac{1}{3}} + v_{\frac{2}{3}}}{2}\right), \ v_{\frac{2}{3}} = v_{\frac{1}{3}} - h\frac{\partial \overline{H}_2}{\partial x}\left(\frac{x_{\frac{1}{3}} + x_{\frac{2}{3}}}{2}, \frac{v_{\frac{1}{3}} + v_{\frac{2}{3}}}{2}\right).$$

Let $\|x\|_A := x^\top A x$ for a matrix A. For any (x, v, ν), define the norm

$$\|(x, v, \lambda)\| := \|x\|_{g(x_{\frac{1}{3}})} + \|v\|_{g(x_{\frac{1}{3}})^{-1}} + h\|A^\top \nu\|_{g(x_{\frac{1}{3}})^{-1}}.$$

If $\left\|(x_{\frac{2}{3}}^{(0)}, v_{\frac{2}{3}}^{(0)}, \nu^{(0)}) - (\widetilde{x}_{\frac{2}{3}}, \widetilde{v}_{\frac{2}{3}}, \nu^*)\right\| \le r$ with $h \le r \le \min(\frac{1}{10}, \frac{\sqrt{h}}{4}, \frac{\|v^*\|_{g(x_0)^{-1}}}{4})$, then

$$\left\|(x_{\frac{2}{3}}^{(L)}, v_{\frac{2}{3}}^{(L)}, \nu^{(L)}) - (\widetilde{x}_{\frac{2}{3}}, \widetilde{v}_{\frac{2}{3}}, \nu^*)\right\| \le \delta$$

for some $L = O\left(\log_{1/C}\frac{r}{\delta}\right)$, where $C = O_n(h)$ is the Lipschitz constant of the map $\mathcal{T}$.

Proof. Since $(x_{\frac{2}{3}}^*, v_{\frac{2}{3}}^*, \nu^*)$ is the fixed point of $\mathcal{T}$ (i.e., $\nu^* = \lambda_1$), we have

$$\nu^* = \nu^* + (LL^\top)^{-1} A g(x_{\text{mid}})^{-1}\left(v_{\text{mid}} - A^\top \nu^*\right)$$

and thus $Ag(x_{\text{mid}})^{-1}v_{\text{mid}} = Ag(x_{\text{mid}})^{-1}A^\top\nu^*$. For invertible $Ag(x_{\text{mid}})^{-1}A^\top$, we have

$$\nu^* = \left(Ag(x_{\text{mid}})^{-1}A^\top\right)^{-1} Ag(x_{\text{mid}})^{-1}v_{\text{mid}}.$$

Similarly by using the definition of the fixed point and this new formula for ν^*,

$$\begin{aligned}
x_{\frac{2}{3}}^* &= x_{\frac{1}{3}} + hg(x_{\text{mid}})^{-1}v_{\text{mid}} - hg(x_{\text{mid}})^{-1}A^\top\nu^* \\
&= x_{\frac{1}{3}} + hg(x_{\text{mid}})^{-1}v_{\text{mid}} - hg(x_{\text{mid}})^{-1}A^\top\left(Ag(x_{\text{mid}})^{-1}A^\top\right)^{-1}Ag(x_{\text{mid}})^{-1}v_{\text{mid}} \\
&= x_{\frac{1}{3}} + h\frac{\partial \overline{H}_2}{\partial v}(x_{\text{mid}}, v_{\text{mid}})
\end{aligned}$$

and

$$\begin{aligned}
v_{\frac{2}{3}}^* &= v_{\frac{1}{3}} - \frac{h}{2}Dg(x_{\text{mid}})[g(x_{\text{mid}})^{-1}(v_{\text{mid}} - A^\top\nu^*), g(x_{\text{mid}})^{-1}(v_{\text{mid}} - A^\top\nu^*)] \\
&= v_{\frac{1}{3}} - h\frac{\partial \overline{H}_2}{\partial x}(x_{\text{mid}}, v_{\text{mid}})
\end{aligned}$$

which shows that $(x_{\frac{2}{3}}^*, v_{\frac{2}{3}}^*)$ is exactly the solution for (x, v) in the equation

$$x = x_{\frac{1}{3}} + h\frac{\partial \overline{H}_2}{\partial v}\left(\frac{x_{\frac{1}{3}} + x}{2}, \frac{v_{\frac{1}{3}} + v}{2}\right), \ v = v_{\frac{1}{3}} - h\frac{\partial \overline{H}_2}{\partial x}\left(\frac{x_{\frac{1}{3}} + x}{2}, \frac{v_{\frac{1}{3}} + v}{2}\right).$$

Next, we show that the iterations in Step 2 converges to $(x_{\frac{2}{3}}^*, v_{\frac{2}{3}}^*, \nu^*)$. If $\left\|(x_{\frac{2}{3}}^{(0)}, v_{\frac{2}{3}}^{(0)}, \nu^{(0)}) - (x_{\frac{2}{3}}^*, v_{\frac{2}{3}}^*, \nu^*)\right\| \le r$ for some $C = O_n(h)$, we have

$$\begin{aligned}
\left\|(x_{\frac{2}{3}}^{(\ell)}, v_{\frac{2}{3}}^{(\ell)}, \nu^{(\ell)}) - (x_{\frac{2}{3}}^*, v_{\frac{2}{3}}^*, \nu^*)\right\| &= \left\|\mathcal{T}(x_{\frac{2}{3}}^{(\ell-1)}, v_{\frac{2}{3}}^{(\ell-1)}, \nu^{(\ell)}) - \mathcal{T}(x_{\frac{2}{3}}^*, v_{\frac{2}{3}}^*, \nu^*)\right\| \\
&\le C\left\|(x_{\frac{2}{3}}^{(\ell-1)}, v_{\frac{2}{3}}^{(\ell-1)}, \nu^{(\ell-1)}) - (x_{\frac{2}{3}}^*, v_{\frac{2}{3}}^*, \nu^*)\right\| \\
&\le C^\ell\left\|(x_{\frac{2}{3}}^{(0)}, v_{\frac{2}{3}}^{(0)}, \nu^{(0)}) - (x_{\frac{2}{3}}^*, v_{\frac{2}{3}}^*, \nu^*)\right\|,
\end{aligned}$$

Bio Model	Full-dim	Consts (m)	Vars (n)	nnz
ecoli	24	72	95	291
cardiac_mit	12	230	220	228
Aci_D21	103	856	851	1758
Aci_MR95	123	917	994	2859
Abi_49176	157	952	1069	2951
Aci_20731	164	1009	1090	2946
Aci_PHEA	328	1319	1561	4640
iAF1260	572	1668	2382	6368
iJO1366	590	1805	2583	7284
Recon1	932	2766	3742	8717
Recon2	2430	5063	7440	19791
Recon3	5335	8399	13543	48187

LP Model	Full-dim	Consts (m)	Vars (n)	nnz
israel	142	174	316	2519
gfrd_pnc	544	616	1160	2393
25fv47	1056	821	1876	10566
pilot_ja	1002	940	2267	11886
sctap2	1410	1090	2500	7334
ship08l	2700	778	4363	9434
cre_a	3703	3516	7248	17368
woodw	4656	1098	8418	23158
80bau3b	9233	2262	12061	22341
ken_18	49896	105127	154699	295946

Table 1: Constraint-based models. Each constraint-based model has a form of $\{x \in \mathbb{R}^n : Ax = b,\, l \le x \le u\}$ for $A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m$ and $l, u \in \mathbb{R}^n$, where the rows and columns correspond to constraints and variables respectively. The full-dimension of each model is obtained by transforming its degenerate subspace to a full dimensional representation (i.e., $A'x \le b'$), and we count the number of nonzero (nnz) entries of a preprocessed matrix A.

where the first equality follows from $(x^*_{\frac{2}{3}}, v^*_{\frac{2}{3}}, \nu^*)$ is the fixed point of $\mathcal{T}$ and the second inequality follows from Lemma 14. Therefore, we have $\left\|(x^{(L)}_{\frac{2}{3}}, v^{(L)}_{\frac{2}{3}}, \nu^{(L)}) - (x^*_{\frac{2}{3}}, v^*_{\frac{2}{3}}, \nu^*)\right\| \le \delta$ for $L = O\left(\log_C \frac{r}{\delta}\right)$. $\square$

Remark 12. Lemma 11 shows that Algorithm 3 converges to the solution of (2.15) in logarithmically many iterations for small enough step size h. In Step 1 of Algorithm 2, v is resampled so that every iteration of Algorithm 2 is a non-degenerate map. Then, the total variation distance between the distributions generated by solving (2.15) using Algorithm 3 and solving (2.15) exactly in one iteration of Algorithm 2 can be bounded by error due to Algorithm 3. Theorem 8 shows that if Step 2 of Algorithm 2 can solve (2.15) exactly, then the process will converge to the exact stationary distribution. Therefore, in order for the accumulated error of Algorithm 2 to remain bounded for polynomially many steps, it suffices to run logarithmically many iterations in Algorithm 3. Any small bias due to the numerical error in the ODE computation is corrected by the filter, and maintaining as small error as possible is important to keep the acceptance probability high.

3 Experiments

In this section, we describe experiments on real-world datasets to examine how effective CRHMC is and compare it with existing samplers. We start by elaborating on experimental settings for reproducibility in Section 3.1. Then we demonstrate that CRHMC is able to sample larger models than previously known to be possible, and is significantly faster in terms of rate of convergence and sampling time in Section 3.2 and Section 3.3, respectively. We also examine its behavior on benchmark instances such as simplices and Birkhoff polytopes in Section 3.4.

3.1 Experimental Setting

Specification. We performed experiments on the Standard DS12 v2 model from MS Azure cloud, which has a 2.1GHz Intel Xeon Platinum 8171M CPU (4 cores) and 28GB memory.

Dataset. As listed in Table 1, we used twelve constraint-based metabolic models from molecular systems biology in the COBRA Toolbox v3.0 [18][2] and ten real-world LP examples randomly chosen from NETLIB

[2]https://github.com/opencobra/cobratoolbox

LP test sets[3]. A polytope from each model is defined by

$$\{x \in \mathbb{R}^n : Ax = b, l \leq x \leq u\}$$

for $A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m$, and $l, u \in \mathbb{R}^n$, which is input to CRHMC for uniform sampling. If a model is unbounded, we make it bounded by setting $l = \max(l, -10^7)$ and $u = \min(u, 10^7)$. As existing packages require full-dimensional representations of polytopes (i.e., $\{x : A'x \leq b'\}$), we transformed all constraint-based models to prepare instances for them as follows: (1) first preprocess each model by removing redundant constraints and appropriately scaling it, (2) find its corresponding full-dimensional description, and (3) round it via the maximum volume ellipsoid (MVE) algorithm making the polytope more amenable to sampling. We note that a full-dimensional polytope can be transformed into a constraint-based polytope and vice versa, so CRHMC can be run on either representation.

Implementation. CRHMC implemented in MATLAB and C++ is available at this repository, which is also integrated into the COBRA Toolbox. Our implementation can be run to sample from general logconcave densities and has a feature for parallelization.

Preprocessing. We preprocessed each constrained-based model prior to sampling. This preprocessing consists mainly of simplifying polytopes, scaling properly for numerical stability, and finding a feasible starting point. To simplify a given polytope, we check if $l_i = u_i$ for each $i \in [n]$ and then incorporate such variables x_i into $Ax = b$. Any dense column is split into several columns with less non-zero entries by introducing additional variables. Then we remove dependent rows of A by Cholesky decomposition. Then we find the Dikin ellipsoid of the polytope. If the width along some axis is smaller than a preset tolerance, then we fix variables in such directions, reducing columns of A. Lastly, we run the interior-point method with the log-barrier to find an analytic center of the polytope, which will be used as a starting point in sampling. When finding the analytic center of the simplified polytope, if a coordinate of the analytic center is too close to a boundary (to be precise, smaller than a preset tolerance boundary 10^{-8}), then we assume that the inequality constraint (either $x_i \leq u_i$ or $l_i \leq x_i$) is tight, and we collapse such a variable by moving it into the constraints $Ax = b$. We go back to the step for removing dependent rows and repeat until no more changes are made to A. Along with simplification, we keep rescaling A, b, l, u for numerical stability.

Competitors. For comparison, we used as a baseline the Coordinate Hit-and-Run algorithm implemented in two different languages. The former is Coordinate Hit-and-Run with Rounding (CHRR) written in MATLAB [9, 17] and the latter is the same algorithm (CDHR) with an R interface and a C++ library, VolEsti [5]. We note that popular sampling packages such as STAN and Pyro were not included in the experiments as they do not support constrained-based models. Even after transforming our dataset to their formats, the transformed dataset were too ill-conditioned for those algorithms to run.

We briefly explain how CHRR works. First, rounding via the MVE algorithm finds the maximum volume ellipsoid inscribed in the polytope and applies, to the polytope, an affine transformation that makes this ellipsoid a unit ball. This procedure puts a possibly highly-skewed polytope into John's position, which guarantees that the polytope contains a unit ball and is contained in a ball of radius n. This position still has a beneficial effect on sampling in practice in the sense that the random walk can converge in fewer steps. After the transformation, the random walk based on Coordinate Hit-and-Run (CHAR) chooses a random coordinate and moves to a random point on the line through the current point along the chosen coordinate.

When running CHRR and CDHR, we recorded a sample every n^2 steps. The mixing rate (i.e., the number of steps required to get a sample from a target distribution) of Hit-and-Run (HAR), a general version of CHAR choosing a random direction (unit vector) instead of a random coordinate, is $O^*(n^2R^2)$ for a polytope P with $B_n \subseteq P \subseteq R \cdot B_n$, where B_n is the unit ball in $\mathbb{R}^n$ [31]. It was proved only recently that CHAR mixes in $O^*(n^9R^2)$ steps on such a polytope [25, 33]. Even though this bound is not as tight as the mixing-rate bound for HAR, it was reported in [17] that CHRR mixes in the same number of steps as HAR empirically. Moreover, the per-step complexity of CHAR can be n times faster than that of HAR, so CHAR brings a significant speed-up in practice.

[3]http://www.netlib.org/lp/data/

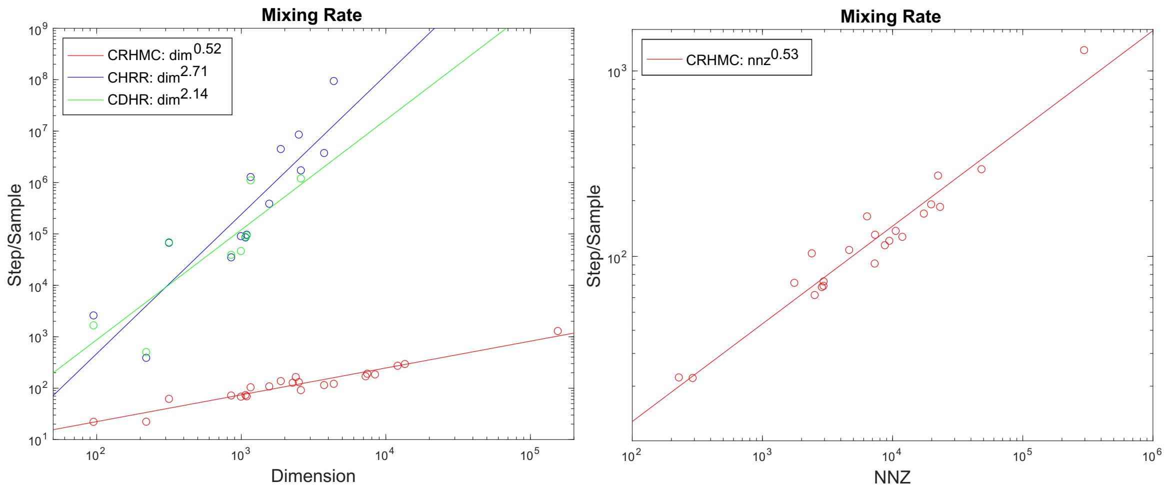

Figure 3.1: Mixing rate of CRHMC and the competitors. Mixing rate of CRHMC was sub-linear in dimension and the nnz of a preprocessed matrix A in a model, whereas the others needed quadratically many steps to converge to uniform distribution. In particular for our dataset, CRHMC mixed up to 6 orders of magnitude earlier than the others. Note that mixing rate of CHAR was very close to quadratic growth when using the full-dimensional scale (the first column in Table 1).

We proceeded with the following additional steps for fair comparison. First, as the VolEsti package does not support the MVE rounding, we rounded each polytope by the MVE algorithm in the CHRR package and then transformed the rounded polytope so that the R interface can read the data file. Next, we limited all algorithms to a single core, since the R interface uses a single core as a default whereas MATLAB uses as many available cores as possible.

Measurements. To evaluate the quality of sampling methods, we measured two quantities, the *number of steps per effective sample* N_s (i.e., mixing rate) and the *sampling time per effective sample,* T_s. The *effective sample size* (ESS) can be thought of as the number of actual independent samples, taking into account correlation of samples from a target distribution. Thus the number of steps per effective sample N_s is estimated by the total number of steps divided by the effective sample size, and the sampling time T_s is estimated as the total sampling time divided by the effective sample size.

In the experiments, each algorithm attempted to draw 1000 uniform samples from a model to get accurate estimates, with limits on running time set to 1 day (3 days for the largest instance *ken_18*) and memory usage to 6GB. If an algorithm passes either the time or the memory limit, we stop the algorithm and measure the quantities of interest based on samples drawn until that moment. After getting uniform samples, we thinned the samples twice to ensure independence of samples; first we computed the ESS of the samples, only kept ESS many samples, and repeated this one more time.

In what follows, we estimated the above quantities only if the ESS is more than 10 and an algorithm does not run into any error while running[4].

3.2 Sub-linear Mixing Rate

We examined how the number of steps per effective sample N_s grows with the number of nonzeros (nnz) of matrix A (after preprocessing) and the number of variables (dimension in the figure). To this end, we counted the total number of steps taken until termination of algorithms and divided it by the effective sample size of drawn samples. Note that we thinned twice to ensure independence of samples used.

The mixing rate of CRHMC was sub-linear in both dimension and nnz, whereas previous implementations based on CHAR required at least n^2 steps per sample as seen in Figure 3.1. On the dataset, mixing rate

[4]When running CDHR from the VolEsti package on some instances, we got an error message "R session aborted and R encountered a fatal error".

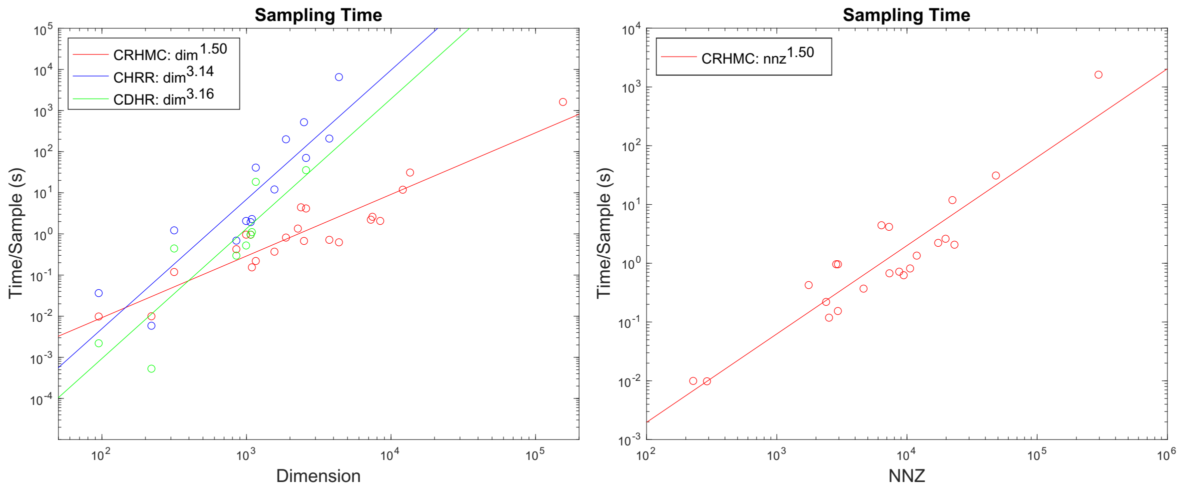

Figure 3.2: Sampling time of CRHMC and the competitors. The sampling time per effective sample of CRHMC was sub-quadratic in dimension and the nnz of a preprocessed matrix A in a model, while the others indicates at least a cubic dependency on dimension. In particular for our dataset, CRHMC was able to obtain a statistically independent sample up to 4 orders of magnitude faster than the others. This benefit of speed-up was actually straightforward from the figure, since CHRR could not obtain enough samples from instances with more than 5000 variables until it ran out of time.

attained was up to 6 orders of magnitude faster for CRHMC compared to CHAR, implying that CRHMC converged to uniform distribution substantially faster than the other competitors. This gap in mixing rate increased super-linearly in dimension, enabling CRHMC to run on large instances of dimension up to 100000.

3.3 Sub-quadratic Sampling Time

We next examined the sampling time T_s in terms of both the nnz of A and the dimension of the instance. We computed the runtime of algorithms until their termination divided by the effective sample size of drawn samples, where we ignored time taken for the preprocessing. Note that the sampling time T_s is essentially multiplication of the mixing rate and the *per-step complexity* (i.e., how much time each step takes).

As shown in Figure 3.2 and Table 2, we found that the per-step complexity was small enough to make the sampling time sub-quadratic in both dimension and nnz, whereas prior implementations based on CHAR had at least a cubic dependency on dimension, despite of a low per-step complexity. On our dataset, the sampling time of CRHMC was up to 4 orders of magnitude less than that of CHRR and CDHR. While CHRR can be used on dimension only up to a few thousands, increasing benefits of sampling time in higher dimension allows CRHMC to run on dimension up to 0.1 million.

3.4 CRHMC on Structured Instances

To see the behavior of CRHMC on very large instances, we ran the algorithm on three families of structured polytopes – hypercube, simplex, and Birkhoff polytope – up to dimension half-million. We attempted to draw 500 uniform samples with a 1 day time limit (except for 2 days for half-million-dimensional Birkhoff polytope). We recall the definitions of these polytopes.

Hypercube. The n-dimensional hypercube is defined by $\{x \in \mathbb{R}^n : -\frac{1}{2} \leq x_i \leq \frac{1}{2}$ for all $i \in [n]\}$. Note that it has no equality constraint and its full-dimension is n.

Simplex. The n-dimensional simplex is defined by $\{x \in \mathbb{R}^n : 0 \leq x_i$ for all $i \in [n]$, $\sum_{i=1}^n x_i = 1\}$. Note that its full-dimension is $n-1$.

Bio Model	Vars (n)	nnz	CRHMC	CHRR	CDHR
ecoli	95	291	0.0098	0.0365	0.0022
cardiac_mit	220	228	0.0100	0.0059	0.0005
Aci_D21	851	1758	0.4257	0.6884	0.2974
Aci_MR95	994	2859	0.9624	2.0668	0.5237
Abi_49176	1069	2951	0.9608	1.9395	0.9622
Aci_20731	1090	2946	0.1540	2.3014	1.1086
Aci_PHEA	1561	4640	0.3701	12.06	-
iAF1260	2382	6368	4.4355	3687.2	-
iJO1366	2583	7284	4.1608	70.5	35.556
Recon1	3742	8717	0.7184	208.5	-
Recon2	7440	19791	2.6116	10445*	-
Recon3	13543	48187	31.114	29211*	-

LP Model	Vars (n)	nnz	CRHMC	CHRR	CDHR
israel	316	2519	0.1186	1.2224	0.4426
gfrd_pnc	1160	2393	0.2199	40.988	18.468
25fv47	1876	10566	0.8159	199.9	-
pilot_ja	2267	11886	1.3490	5059*	-
sctap2	2500	7334	0.6752	520.2	-
ship08l	4363	9434	0.6258	6512	-
cre_a	7248	17368	2.2205	30455*	-
woodw	8418	23158	2.0689	30307*	-
80bau3b	12061	22341	11.881	47432*	-
ken_18	154699	295946	1616.3	-	-

Table 2: Sampling time per effective sample of CHRR and CRHMC. We note that CRHMC is 1000 times faster than CHRR on the latest metabolic network (Recon3). Sampling time with asterisk (*) indicates that the effective sample size is less than 10.

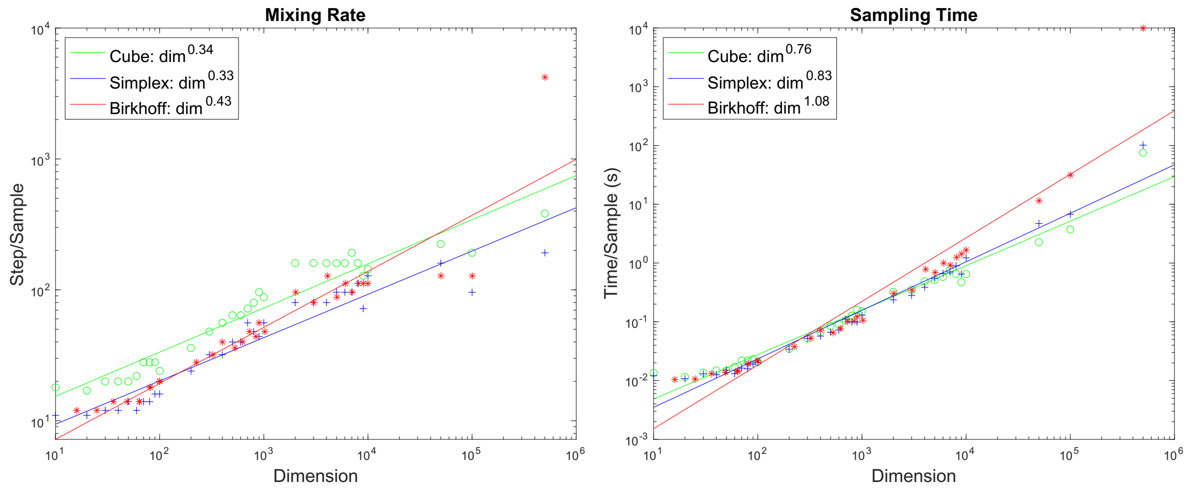

Figure 3.3: Mixing rate and sampling time on structured polytopes including hybercubes, simplices, and Birkhoff polytopes. CRHMC is scalable up to 0.5 million dimension on hypercubes and simplices and up to 0.1 million dimension on Birkhoff polytopes. We note that on the 0.5 million dimensional Birkhoff polytope the ESS is only 16, which is not reliable compared to the ESS on the other instances.

Birkhoff Polytope. The n^{th} Birkhoff polytope B_n is the set of all doubly stochastic $n \times n$ matrices (or the convex hull of all permutation matrices), which is defined as

$$B_n = \{(X_{ij})_{i,j\in[n]} : \sum_j X_{ij} = 1 \text{ for all } i \in [n], \sum_i X_{ij} = 1 \text{ for all } j \in [n], \text{ and } X_{ij} \geq 0\}.$$

Namely, B_n is defined in a constrained $\mathbb{R}^{n^2}$-dimensional space, and its full-dimension is $n^2-(2n-1) = (n-1)^2$. We ran CRHMC on $B_{\sqrt{n}}$ to examine its efficiency on (roughly) n-dimensional Birkhoff polytope.

To the best of our knowledge, this is the first demonstration that it is possible to sample such a large model. As seen in Figure 3.3, CRHMC can scale smoothly up to half-million dimension on hypercubes and simplices and up to dimension 10^5 for Birkhoff polytopes (we could not obtain a reliable estimate of mixing rate and sampling time on the half-million dimensional Birkhoff polytope, as the ESS is only 16 after 2 days). However, we believe that one can find room for further improvement of CRHMC by tuning parameters or leveraging engineering techniques. We also expect that CRHMC enables us to estimate the volume of B_n for $n \geq 20$, going well beyond the previously best possible dimension.

Acknowledgment. The authors are grateful to Ben Cousins for helpful discussions, and to Ronan Fleming, Ines Thiele and their research groups for advice on metabolic models. This work was supported in part by NSF awards DMS-1839116, DMS-1839323, **CCF-1909756, CCF-2007443 and CCF-2134105**.

References

[1] Ivona Bezáková, Daniel Štefankovič, Vijay V Vazirani, and Eric Vigoda. Accelerating simulated annealing for the permanent and combinatorial counting problems. *SIAM Journal on Computing*, 37(5):1429–1454, 2008.

[2] Eli Bingham, Jonathan P. Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul A. Szerlip, Paul Horsfall, and Noah D. Goodman. Pyro: Deep Universal Probabilistic Programming. *Journal of Machine Learning Research (JMLR)*, 20:28:1–28:6, 2019.

[3] Marcus Brubaker, Mathieu Salzmann, and Raquel Urtasun. A family of MCMC methods on implicitly defined manifolds. In *Artificial intelligence and statistics (AISTATS)*, pages 161–172, 2012.

[4] Yogin E Campbell and Timothy A Davis. Computing the sparse inverse subset: an inverse multifrontal approach. *University of Florida, Technical Report TR-95-021*, 1995.

[5] Apostolos Chalkis and Vissarion Fisikopoulos. volEsti: Volume approximation and sampling for convex polytopes in R. *arXiv preprint arXiv:2007.01578*, 2020.

[6] Yuansi Chen, Raaz Dwivedi, Martin J Wainwright, and Bin Yu. Fast mixing of Metropolized Hamiltonian Monte Carlo: Benefits of multi-step gradients. *Journal of Machine Learning Research (JMLR)*, 21:92–1, 2020.

[7] Xiang Cheng, Niladri S Chatterji, Peter L Bartlett, and Michael I Jordan. Underdamped Langevin MCMC: a non-asymptotic analysis. In *Conference on Learning Theory (COLT)*, pages 300–323. PMLR, 2018.

[8] Sinho Chewi, Thibaut Le Gouic, Chen Lu, Tyler Maunu, Philippe Rigollet, and Austin J Stromme. Exponential ergodicity of mirror-Langevin diffusions. *arXiv preprint arXiv:2005.09669*, 2020.

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

[10] Timothy A Davis. *Direct methods for sparse linear systems.* SIAM, 2006.

[11] James W Demmel. *Applied numerical linear algebra.* SIAM, 1997.

[12] Simon Duane, Anthony D Kennedy, Brian J Pendleton, and Duncan Roweth. Hybrid Monte Carlo. *Physics letters B*, 195(2):216–222, 1987.

[13] Raaz Dwivedi, Yuansi Chen, Martin J Wainwright, and Bin Yu. Log-concave sampling: Metropolis-Hastings algorithms are fast! In *Conference on Learning Theory (COLT)*, pages 793–797. PMLR, 2018.

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

[15] Mark Girolami and Ben Calderhead. Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, 2011.

[16] Andreas Griewank and Andrea Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation.* SIAM, 2008.

[17] Hulda S Haraldsdóttir, Ben Cousins, Ines Thiele, Ronan MT Fleming, and Santosh Vempala. Chrr: coordinate hit-and-run with rounding for uniform sampling of constraint-based models. *Bioinformatics*, 33(11):1741–1743, 2017.

[18] Laurent Heirendt, Sylvain Arreckx, Thomas Pfau, Sebastián N Mendoza, Anne Richelle, Almut Heinken, Hulda S Haraldsdóttir, Jacek Wachowiak, Sarah M Keating, Vanja Vlasov, et al. Creation and analysis of biochemical constraint-based models using the COBRA Toolbox V. 3.0. *Nature protocols*, 14(3):639–702, 2019.

[19] Matthew D Hoffman, Andrew Gelman, et al. The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research (JMLR)*, 15(1):1593–1623, 2014.

[20] Mark Jerrum, Alistair Sinclair, and Eric Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *Journal of the ACM (JACM)*, 51(4):671–697, 2004.

[21] He Jia, Aditi Laddha, Yin Tat Lee, and Santosh Vempala. Reducing isotropy and volume to KLS: an $O^*(n^3\psi^2)$ volume algorithm. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 961–974, 2021.

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

[23] Ravindran Kannan and Hariharan Narayanan. Random walks on polytopes and an affine interior point method for linear programming. *Mathematics of Operations Research*, 37(1):1–20, 2012.

[24] Zachary A King, Justin Lu, Andreas Dräger, Philip Miller, Stephen Federowicz, Joshua A Lerman, Ali Ebrahim, Bernhard O Palsson, and Nathan E Lewis. BiGG Models: A platform for integrating, standardizing and sharing genome-scale models. *Nucleic acids research*, 44(D1):D515–D522, 2016.

[25] Aditi Laddha and Santosh Vempala. Convergence of Gibbs sampling: Coordinate Hit-and-Run mixes fast. *The 37th International Symposium on Computational Geometry (SoCG)*, 2021.

[26] Yin Tat Lee, Ruoqi Shen, and Kevin Tian. Logsmooth gradient concentration and tighter runtimes for metropolized Hamiltonian Monte Carlo. In *Conference on Learning Theory (COLT)*, pages 2565–2597. PMLR, 2020.

[27] Yin Tat Lee and Santosh S Vempala. Geodesic walks in polytopes. In *Proceedings of the 49th Annual ACM SIGACT Symposium on theory of Computing (STOC)*, pages 927–940, 2017.

[28] Yin Tat Lee and Santosh S Vempala. Convergence rate of Riemannian Hamiltonian Monte Carlo and faster polytope volume computation. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1115–1121, 2018.

[29] Yin Tat Lee and Man-Chung Yue. Universal barrier is n-self-concordant. *Mathematics of Operations Research*, 2021.

[30] Nathan E Lewis, Harish Nagarajan, and Bernhard O Palsson. Constraining the metabolic genotype–phenotype relationship using a phylogeny of in silico methods. *Nature Reviews Microbiology*, 10(4):291–305, 2012.

[31] László Lovász and Santosh Vempala. Hit-and-run from a corner. *SIAM Journal on Computing*, 35(4):985–1005, 2006.

[32] Hariharan Narayanan. Randomized interior point methods for sampling and optimization. *The Annals of Applied Probability*, 26(1):597–641, 2016.

[33] Hariharan Narayanan and Piyush Srivastava. On the mixing time of coordinate Hit-and-Run. *Combinatorics, Probability and Computing*, pages 1–13, 2021.

[34] Radford M Neal et al. MCMC using Hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.

[35] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2003.

[36] Yurii Nesterov and Arkadii Nemirovskii. *Interior-point polynomial algorithms in convex programming.* SIAM, 1994.

[37] Gareth O Roberts and Richard L Tweedie. Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, pages 341–363, 1996.

[38] John Salvatier, Thomas V Wiecki, and Christopher Fonnesbeck. Probabilistic programming in Python using PyMC3. *PeerJ Computer Science*, 2:e55, 2016.

[39] Ruoqi Shen and Yin Tat Lee. The randomized midpoint method for log-concave sampling. *arXiv preprint arXiv:1909.05503*, 2019.

[40] Umut Simsekli, Roland Badeau, Taylan Cemgil, and Gaël Richard. Stochastic quasi-newton Langevin Monte Carlo. In *International Conference on Machine Learning (ICML)*, pages 642–651. PMLR, 2016.

[41] Stan Development Team. RStan: the R interface to Stan, 2020. R package version 2.21.2.

[42] Kazuhiro Takahashi. Formation of sparse bus impedance matrix and its application to short circuit study. In *Proceeding of PICA Conference, June, 1973*, 1973.

[43] Ines Thiele, Neil Swainston, Ronan MT Fleming, Andreas Hoppe, Swagatika Sahoo, Maike K Aurich, Hulda Haraldsdottir, Monica L Mo, Ottar Rolfsson, Miranda D Stobbe, et al. A community-driven global reconstruction of human metabolism. *Nature biotechnology*, 31(5):419–425, 2013.

A Missing Definitions and Details

A.1 Definitions

Pseudo-inverse. For a matrix $A \in \mathbb{R}^{m\times n}$, it is well known that there always exists the unique pseudo-inverse matrix $A^\dagger$ that satisfies the following conditions:

1. $A^\dagger A A^\dagger = A^\dagger$
2. $AA^\dagger A = A$
3. $AA^\dagger$ and $A^\dagger A$ are symmetric

It is also well known that $\mathrm{Null}(A^\dagger) = \mathrm{Null}(A^\top)$ and $\mathrm{Range}(A^\dagger) = \mathrm{Range}(A^\top)$.

Pseudo-determinant. For a square matrix A, its pseudo-determinant $\mathrm{pdet}(A)$ is defined as the product of non-zero eigenvalues of A.

Leverage Score. For a matrix $A \in \mathbb{R}^{m\times n}$, the leverage score of the i^{th} row is $(A(A^\top A)^\dagger A^\top)_{ii}$ for $i \in [m]$. When A is full-rank, it is simply $(A(A^\top A)^{-1}A^\top)_{ii}$.

Log-barrier. For a polytope $P = \{x \in \mathbb{R}^n : Ax \le b\}$ where $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$, let us denote the i^{th} row of A by a_i and the i^{th} row of b by b_i. The log-barrier of P is defined by

$$\phi(x) = -\sum_{i=1}^{m} \log(b_i - a_i^\top x).$$

Note that its Hessian matrix $\nabla^2\phi(x)$ is diagonal.

A.2 Details

Inverse and Determinant of Block Matrix. For a square matrix $M = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$ with blocks A, B, C, D of same size, if D and $A - BD^{-1}C$ are invertible, then its inverse and determinant can be computed by

$$\begin{aligned} M^{-1} &= \begin{bmatrix} (A-BD^{-1}C)^{-1} & -(A-BD^{-1}C)^{-1}BD^{-1} \\ -D^{-1}C(A-BD^{-1}C)^{-1} & D^{-1} + D^{-1}C(A-BD^{-1}C)^{-1}BD^{-1} \end{bmatrix} \\ \det(M) &= \det(D)\det(A - BD^{-1}C). \end{aligned}$$

Cholesky Decomposition. For a symmetric positive definite matrix A, there exists a lower triangular matrix L such that $LL^\top = A$.

B Bound on Algorithm Error

Recall that $\|v\|_A^2 = v^\top A v$ for a matrix A.

Lemma 13. *Suppose $g(x) = \nabla^2\phi(x)$ for some highly self-concordance barrier ϕ. Then, we have that*

- $(1-\|y-x\|_{g(x)})^2 g(x) \preceq g(y) \preceq \frac{1}{(1-\|y-x\|_{g(x)})^2} g(x)$,
- $\|Dg(x)[v,v]\|_{g(x)^{-1}} \leq 2\|v\|_{g(x)}^2$,
- $\|Dg(x)[v,v] - Dg(y)[v,v]\|_{g(x)^{-1}} \leq \frac{6}{(1-\|y-x\|_{g(x)})^3}\|v\|_{g(x)}^2 \|y-x\|_{g(x)}$.

Proof. The first fact follows from Theorem 4.1.6 in [35]. The second fact follows from Lemma 4.1.2 in [35]. The third fact is from the following calculation:

$$\begin{aligned}
&\|Dg(y)[v,v] - Dg(x)[v,v]\|_{g(x)^{-1}} \\
\leq &\int_0^1 \|D^2 g(x+t(y-x))[v,v,y-x]\|_{g(x)^{-1}} dt \\
\leq &\int_0^1 \frac{1}{1-t\|y-x\|_{g(x)}} \|D^2 g(x+t(y-x))[v,v,y-x]\|_{g(x+t(y-x))^{-1}} dt \\
\leq &\int_0^1 \frac{6}{1-t\|y-x\|_{g(x)}} \|v\|_{g(x+t(y-x))}^2 \|y-x\|_{g(x+t(y-x))} dt \\
\leq &\int_0^1 \frac{6}{(1-t\|y-x\|_{g(x)})^4} dt \cdot \|v\|_{g(x)}^2 \|y-x\|_{g(x)} \\
\leq &\frac{6}{(1-\|y-x\|_{g(x)})^3} \|v\|_{g(x)}^2 \|y-x\|_{g(x)}.
\end{aligned}$$

where the third and fifth line follow from the first fact, and the fourth line follows from Proposition 9.1.1 in [36]. ☐

Lemma 14. *Let $g(x) = \nabla^2\phi(x)$ for some highly self-concordance barrier ϕ. Given x_0, v_0 and L such that $LL^\top = Ag(x_0)^{-1}A^\top$, consider the map*

$$\mathcal{T}(x,v,\lambda) = \begin{pmatrix} x_0 + hg(x_{1/2})^{-1}(v_{1/2} - A^\top\lambda_1) \\ v_0 - \frac{h}{2} Dg(x_{1/2})[g(x_{1/2})^{-1}(v_{1/2} - A^\top\lambda_1), g(x_{1/2})^{-1}(v_{1/2} - A^\top\lambda_1)] \\ \lambda_1 \end{pmatrix}$$

where $x_{1/2} = (x_0+x)/2$, $v_{1/2} = (v_0+v)/2$ and $\lambda_1 = \lambda + (LL^\top)^{-1} A g(x_{1/2})^{-1}\left(v_{1/2} - A^\top\lambda\right)$. Let (x^*, v^*, λ^*) be a fixed point of $\mathcal{T}$. For any x, v, λ, we define the norm

$$\|(x,v,\lambda)\| = \|x\|_{g(x_0)} + \|v\|_{g(x_0)^{-1}} + h\|A^\top\lambda\|_{g(x_0)^{-1}}.$$

Let $\Omega = \{(x,v,\lambda) : \|(x,v,\lambda) - (x^*,v^*,\lambda^*)\| \leq r\}$ with $h \leq r \leq \min(\frac{1}{10}, \frac{\sqrt{h}}{4}, \frac{\|v^*\|_{g(x_0)^{-1}}}{4})$. Suppose that $(x_0, v_0, 0) \in \Omega$. Then, for any $(x,v,\lambda), (\overline{x},\overline{v},\overline{\lambda}) \in \Omega$, we have

$$\|\mathcal{T}(x,v,\lambda) - \mathcal{T}(\overline{x},\overline{v},\overline{\lambda})\| \leq C\|(x,v,\lambda) - (\overline{x},\overline{v},\overline{\lambda})\|$$

where $C = (\frac{3r}{h} + \|v^*\|_{g(x_0)^{-1}})(400r + 18h\|v^*\|_{g(x_0)^{-1}})$.

Remark 15. Note that we should think $r = \Theta_n(h)$ because that is the distance between $(x_0, v_0, 0)$ and (x^*, v^*, λ^*). In that case, the Lipschitz constant of $\mathcal{T}$ is $O_n(h\|v^*\|_{g(x_0)^{-1}}^2) = O_n(h)$. Hence, if the step size h is small enough, then $\mathcal{T}$ is a contractive mapping. In practice, we can take h close to a constant because g is decomposable into barriers in each dimension and the bound can be improved using this.

Proof. We use $\mathcal{T}(x,v,\lambda)_x$ to denote the x component of $\mathcal{T}(x,v,\lambda)$ and similarly for $\mathcal{T}(x,v,\lambda)_v$ and $\mathcal{T}(x,v,\lambda)_\lambda$. For simplicity, we write $g_0 = g(x_0)$, $g_{1/2} = g(x_{1/2})$ and $\overline{g}_{1/2} = g(\overline{x_{1/2}})$. By the assumption, we have that

$$\|x - x_0\|_{g_0} \le \|x - x^*\|_{g_0} + \|x^* - x_0\|_{g_0} \le 2r.$$

Similarly, $\|\overline{x} - x_0\|_{g_0} \le 2r$.

We first bound $\mathcal{T}(x,v,\lambda)_\lambda$. Note that

$$\mathcal{T}(\overline{x},\overline{v},\overline{\lambda})_\lambda - \mathcal{T}(x,v,\lambda)_\lambda = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4$$

where

$$\begin{aligned}
\alpha_1 &= (I - (LL^\top)^{-1} A g_0^{-1} A^\top)(\overline{\lambda} - \lambda),\\
\alpha_2 &= (LL^\top)^{-1} A g_0^{-1} (\overline{v_{1/2}} - v_{1/2}),\\
\alpha_3 &= (LL^\top)^{-1} A (g_{1/2}^{-1} - g_0^{-1})((\overline{v_{1/2}} - A^\top \overline{\lambda}) - (v_{1/2} - A^\top \lambda)),\\
\alpha_4 &= (LL^\top)^{-1} A (\overline{g}_{1/2}^{-1} - g_{1/2}^{-1})(\overline{v_{1/2}} - A^\top \overline{\lambda}).
\end{aligned}$$

Using that $LL^\top = Ag(x_0)^{-1}A^\top$, we have $\alpha_1 = 0$. For α_2, we have

$$\begin{aligned}
\|A^\top \alpha_2\|_{g_0^{-1}}^2 &= (\overline{v_{1/2}} - v_{1/2})^\top g_0^{-1} A^\top (LL^\top)^{-1} A g_0^{-1} A^\top (L^\top L)^{-1} A g_0^{-1} (\overline{v_{1/2}} - v_{1/2})\\
&= (\overline{v_{1/2}} - v_{1/2})^\top g_0^{-1} A^\top (A g_0^{-1} A^\top)^{-1} A g_0^{-1} (\overline{v_{1/2}} - v_{1/2})\\
&\le (\overline{v_{1/2}} - v_{1/2})^\top g_0^{-1} (\overline{v_{1/2}} - v_{1/2})\\
&= \frac{1}{4} \|\overline{v} - v\|_{g_0^{-1}}^2
\end{aligned}$$

where we use $LL^\top = Ag(x_0)^{-1}A^\top$ and $g_0^{-1/2} A^\top (A g_0^{-1} A^\top)^{-1} A g_0^{-1/2} = B^\top (BB^\top)^{-1} B \preceq I$ for $B = A g_0^{-1/2}$. For α_3, by self-concordance of g (Lemma 13) and $\|x - x_0\|_{g_0} \le 2r$, we have

$$(1-r)^2 g_0 \preceq g_{1/2} \preceq \frac{1}{(1-r)^2} g_0 \tag{B.1}$$

and hence $(g_0^{1/2}(g_{1/2}^{-1} - g_0^{-1})g_0^{1/2})^2 \preceq ((1-r)^{-2} - 1)^2 I$. Using this and $P = g_0^{-1/2} A^\top (A g_0^{-1} A^\top)^{-1} A g_0^{-1/2} \preceq I$, we have

$$\begin{aligned}
\|A^\top \alpha_3\|_{g_0^{-1}} &= \|g_0^{1/2}(g_{1/2}^{-1} - g_0^{-1})((\overline{v_{1/2}} - A^\top \overline{\lambda}) - (v_{1/2} - A^\top \lambda))\|_P\\
&\le \|g_0^{1/2}(g_{1/2}^{-1} - g_0^{-1})((\overline{v_{1/2}} - A^\top \overline{\lambda}) - (v_{1/2} - A^\top \lambda))\|_2\\
&\le ((1-r)^{-2} - 1)\|g_0^{-1/2}((\overline{v_{1/2}} - A^\top \overline{\lambda}) - (v_{1/2} - A^\top \lambda))\|_2\\
&\le ((1-r)^{-2} - 1)(\frac{1}{2}\|\overline{v} - v\|_{g_0^{-1}} + \|A^\top(\overline{\lambda} - \lambda)\|_{g_0^{-1}}).
\end{aligned}$$

Using $r \le 1/10$, we have

$$\|A^\top \alpha_3\|_{g_0^{-1}} \le 1.2r\|\overline{v} - v\|_{g_0^{-1}} + 2.4r\|A^\top(\overline{\lambda} - \lambda)\|_{g_0^{-1}}.$$

For α_4, similarly, we have

$$\begin{aligned}
\|A^\top \alpha_4\|_{g_0^{-1}} &\le ((1 - 0.5\|\overline{x} - x\|_{g_{1/2}})^{-2} - 1)\|\overline{v_{1/2}} - A^\top \overline{\lambda}\|_{g_0^{-1}}\\
&\le ((1 - 0.6\|\overline{x} - x\|_{g_0})^{-2} - 1)\|\overline{v_{1/2}} - A^\top \overline{\lambda}\|_{g_0^{-1}}\\
&\le 1.5\|\overline{x} - x\|_{g_0}\|\overline{v_{1/2}} - A^\top \overline{\lambda}\|_{g_0^{-1}}
\end{aligned}$$

where we used $g_{1/2} \preceq 1.2 g_0$ (by (B.1)) in the second inequality and $\|\overline{x} - x\|_{g_0} \le \|\overline{x} - x^*\|_{g_0} + \|x - x^*\|_{g_0} \le \frac{1}{5}$ at the end. Combining everything, we have

$$\begin{aligned}
&\|A^\top(\mathcal{T}(\overline{x},\overline{v},\overline{\lambda})_\lambda - \mathcal{T}(x,v,\lambda)_\lambda)\|_{g_0^{-1}} = \|A^\top(\overline{\lambda}_1 - \lambda_1)\|_{g_0^{-1}}\\
&\quad \le 0.7\|\overline{v} - v\|_{g_0^{-1}} + 2.4r\|A^\top(\overline{\lambda} - \lambda)\|_{g_0^{-1}} + 1.5\|\overline{x} - x\|_{g_0}\|\overline{v_{1/2}} - A^\top \overline{\lambda}\|_{g_0^{-1}}.
\end{aligned} \tag{B.2}$$

Now we bound $\mathcal{T}(x,v,\lambda)_x$. Note that

$$\mathcal{T}(\overline{x},\overline{v},\overline{\lambda})_x - \mathcal{T}(x,v,\lambda)_x = h\beta_1 + h\beta_2$$

where

$$\begin{aligned}\beta_1 =& g_{1/2}^{-1}((\overline{v}_{1/2} - A^\top\overline{\lambda}_1) - (v_{1/2} - A^\top\lambda_1)),\\ \beta_2 =& (\overline{g}_{1/2}^{-1} - g_{1/2}^{-1})(\overline{v}_{1/2} - A^\top\overline{\lambda}_1).\end{aligned}$$

By a proof similar to above, we have

$$\begin{aligned}\|\beta_1\|_{g_0} &\leq 1.2(\|\overline{v}_{1/2} - v_{1/2}\|_{g_0^{-1}} + \|A^\top(\overline{\lambda}_1 - \lambda_1)\|_{g_0^{-1}}),\\ \|\beta_2\|_{g_0} &\leq 0.6\|\overline{x} - x\|_{g_0}\|\overline{v}_{1/2} - A^\top\overline{\lambda}_1\|_{g_0^{-1}}.\end{aligned}$$

and thus

$$\begin{aligned}&\|\mathcal{T}(\overline{x},\overline{v},\overline{\lambda})_x - \mathcal{T}(x,v,\lambda)_x\|_{g_0}\\ &\quad\leq 0.6h\|\overline{v} - v\|_{g_0^{-1}} + 1.2h\|A^\top(\overline{\lambda}_1 - \lambda_1)\|_{g_0^{-1}} + 0.6h\|\overline{x} - x\|_{g_0}\|\overline{v}_{1/2} - A^\top\overline{\lambda}_1\|_{g_0^{-1}}.\end{aligned}$$

Finally, we bound $\mathcal{T}(x,v,\lambda)_v$. We split the term

$$\mathcal{T}(\overline{x},\overline{v},\overline{\lambda})_v - \mathcal{T}(x,v,\lambda)_v = \frac{h}{2}\gamma_1 + \frac{h}{2}\gamma_2$$

where

$$\begin{aligned}\gamma_1 =& Dg(x_{1/2})[\overline{g}_{1/2}^{-1}(\overline{v}_{1/2} - A^\top\overline{\lambda}_1), \overline{g}_{1/2}^{-1}(\overline{v}_{1/2} - A^\top\overline{\lambda}_1)]\\ &- Dg(x_{1/2})[g_{1/2}^{-1}(v_{1/2} - A^\top\lambda_1), g_{1/2}^{-1}(v_{1/2} - A^\top\lambda_1)],\\ \gamma_2 =& Dg(\overline{x}_{1/2})[\overline{g}_{1/2}^{-1}(\overline{v}_{1/2} - A^\top\overline{\lambda}_1), \overline{g}_{1/2}^{-1}(\overline{v}_{1/2} - A^\top\overline{\lambda}_1)]\\ &- Dg(x_{1/2})[\overline{g}_{1/2}^{-1}(\overline{v}_{1/2} - A^\top\overline{\lambda}_1), \overline{g}_{1/2}^{-1}(\overline{v}_{1/2} - A^\top\overline{\lambda}_1)].\end{aligned}$$

Let $\overline{\eta} = \overline{g}_{1/2}^{-1}(\overline{v}_{1/2} - A^\top\overline{\lambda}_1)$ and $\eta = g_{1/2}^{-1}(v_{1/2} - A^\top\lambda_1)$. For γ_1, we have that

$$\begin{aligned}&\|Dg(x_{1/2})[\overline{\eta},\overline{\eta}] - Dg(x_{1/2})[\eta,\eta]\|_{g_{1/2}^{-1}}\\ &\quad\leq 2\|Dg(x_{1/2})[\overline{\eta}-\eta,\overline{\eta}]\|_{g_{1/2}^{-1}} + \|Dg(x_{1/2})[\overline{\eta}-\eta,\overline{\eta}-\eta]\|_{g_{1/2}^{-1}}\\ &\quad\leq 4\|\overline{\eta}-\eta\|_{g_{1/2}}\|\overline{\eta}\|_{g_{1/2}} + 2\|\overline{\eta}-\eta\|_{g_{1/2}}^2\end{aligned}$$

where we use Lemma 13. Using $g_{1/2} \preceq 1.2g_0$ (by (B.1)),

$$\begin{aligned}\|\gamma_1\|_{g_0^{-1}} \leq& 4\|(\overline{v}_{1/2} - A^\top\overline{\lambda}_1) - (v_{1/2} - A^\top\lambda_1)\|_{g_0^{-1}}\|\overline{v}_{1/2} - A^\top\overline{\lambda}_1\|_{g_0^{-1}}\\ &+ 2\|(\overline{v}_{1/2} - A^\top\overline{\lambda}_1) - (v_{1/2} - A^\top\lambda_1)\|_{g_0^{-1}}^2.\end{aligned}$$

For γ_2, we use Lemma 13 and get

$$\begin{aligned}\|\gamma_2\|_{g_0^{-1}} &\leq \frac{4}{(1-0.6\|\overline{x}-x\|_{g_0})^3}\|\overline{v}_{1/2} - A^\top\overline{\lambda}_1\|_{g_0^{-1}}^2\|\overline{x}-x\|_{g_0}\\ &\leq 6\|\overline{v}_{1/2} - A^\top\overline{\lambda}_1\|_{g_0^{-1}}^2\|\overline{x}-x\|_{g_0}.\end{aligned}$$

Combining everything, we have

$$\begin{aligned}&\|\mathcal{T}(\overline{x},\overline{v},\overline{\lambda})_v - \mathcal{T}(x,v,\lambda)_v\|_{g_0^{-1}}\\ &\quad\leq 2h\|(\overline{v}_{1/2} - A^\top\overline{\lambda}_1) - (v_{1/2} - A^\top\lambda_1)\|_{g_0^{-1}}\|\overline{v}_{1/2} - A^\top\overline{\lambda}_1\|_{g_0^{-1}}\\ &\qquad + h\|(\overline{v}_{1/2} - A^\top\overline{\lambda}_1) - (v_{1/2} - A^\top\lambda_1)\|_{g_0^{-1}}^2\\ &\qquad + 3h\|\overline{v}_{1/2} - A^\top\overline{\lambda}_1\|_{g_0^{-1}}^2\|\overline{x}-x\|_{g_0}\end{aligned}$$

□

Combining the bounds for $\mathcal{T}_\lambda, \mathcal{T}_x, \mathcal{T}_v$, we have

$$\begin{aligned}
&\|\mathcal{T}(x,v,\lambda) - \mathcal{T}(\overline{x},\overline{v},\overline{\lambda})\| \\
&\le 0.7h\|\overline{v}-v\|_{g_0^{-1}} + 2.4rh\|A^\top(\overline{\lambda}-\lambda)\|_{g_0^{-1}} + 1.5h\|\overline{x}-x\|_{g_0}\|\overline{v_{1/2}} - A^\top\overline{\lambda}\|_{g_0^{-1}} \\
&\quad + 0.6h\|\overline{v}-v\|_{g_0^{-1}} + 1.2h\|A^\top(\overline{\lambda}_1-\lambda_1)\|_{g_0^{-1}} + 0.6h\|\overline{x}-x\|_{g_0}\|\overline{v}_{1/2} - A^\top\overline{\lambda}_1\|_{g_0^{-1}} \\
&\quad + 2h\|(\overline{v}_{1/2} - A^\top\overline{\lambda}_1) - (v_{1/2} - A^\top\lambda_1)\|_{g_0^{-1}}\|\overline{v}_{1/2} - A^\top\overline{\lambda}_1\|_{g_0^{-1}} \\
&\quad + h\|(\overline{v}_{1/2} - A^\top\overline{\lambda}_1) - (v_{1/2} - A^\top\lambda_1)\|_{g_0^{-1}}^2 \\
&\quad + 3h\|\overline{v}_{1/2} - A^\top\overline{\lambda}_1\|_{g_0^{-1}}^2\|\overline{x}-x\|_{g_0}.
\end{aligned}$$

To simplify the terms, we note that

$$\begin{aligned}
\|\overline{v}_{1/2} - A^\top\overline{\lambda}_1\|_{g_0^{-1}} \le& \|\overline{v}_{1/2} - A^\top\overline{\lambda}\|_{g_0^{-1}} + \|A^\top(LL^\top)^{-1}A\overline{g}_{1/2}^{-1}\left(\overline{v}_{1/2} - A^\top\overline{\lambda}\right)\|_{g_0^{-1}} \\
=& \|\overline{v}_{1/2} - A^\top\overline{\lambda}\|_{g_0^{-1}} + \|g_0^{1/2}\overline{g}_{1/2}^{-1}\left(\overline{v}_{1/2} - A^\top\overline{\lambda}\right)\|_P \\
\le& \|\overline{v}_{1/2} - A^\top\overline{\lambda}\|_{g_0^{-1}} + \|g_0^{1/2}\overline{g}_{1/2}^{-1}\left(\overline{v}_{1/2} - A^\top\overline{\lambda}\right)\|_2 \\
\le& 3\|\overline{v}_{1/2} - A^\top\overline{\lambda}\|_{g_0^{-1}}.
\end{aligned}$$

Using this and simplifying, we have

$$\begin{aligned}
&\|\mathcal{T}(x,v,\lambda) - \mathcal{T}(\overline{x},\overline{v},\overline{\lambda})\| \\
\le& 1.3h\|\overline{v}-v\|_{g_0^{-1}} + 2.4rh\|A^\top(\overline{\lambda}-\lambda)\|_{g_0^{-1}} + 3.3h\|\overline{x}-x\|_{g_0}\|\overline{v_{1/2}} - A^\top\overline{\lambda}\|_{g_0^{-1}} \\
&+ 1.2h\|A^\top(\overline{\lambda}_1-\lambda_1)\|_{g_0^{-1}} \\
&+ 6h(\frac{1}{2}\|\overline{v}-v\|_{g_0^{-1}} + \|A^\top(\overline{\lambda}_1-\lambda_1)\|_{g_0^{-1}})\|\overline{v}_{1/2} - A^\top\overline{\lambda}\|_{g_0^{-1}} \\
&+ h\|\overline{v}-v\|_{g_0^{-1}}^2 + 2h\|A^\top(\overline{\lambda}_1-\lambda_1)\|_{g_0^{-1}}^2 \\
&+ 27h\|\overline{v}_{1/2} - A^\top\overline{\lambda}\|_{g_0^{-1}}^2\|\overline{x}-x\|_{g_0}.
\end{aligned}$$

Next, we note that

$$\begin{aligned}
\|\overline{v_{1/2}} - A^\top\overline{\lambda}\|_{g_0^{-1}} \le& \frac{1}{2}\|\overline{v}-v^*\|_{g_0^{-1}} + \frac{1}{2}\|v_0-v^*\|_{g_0^{-1}} + \|v^*\|_{g_0^{-1}} \\
&+ \frac{1}{2}\|A^\top\overline{\lambda} - A^\top\lambda^*\|_{g_0^{-1}} + \frac{1}{2}\|A^\top\overline{\lambda} - A^\top\lambda^*\|_{g_0^{-1}} + \|A^\top\lambda^*\|_{g_0^{-1}} \\
\le& \frac{1}{2}r + \frac{1}{2}r + \|v^*\|_{g_0^{-1}} + \frac{r}{2h} + \frac{r}{2h} + \frac{r}{h} \le \frac{3r}{h} + \|v^*\|_{g_0^{-1}}
\end{aligned}$$

Using this, (B.2), $h \le r$, $r^2 \le \frac{h}{16}$, $r \le \|v^*\|_{g_0^{-1}}/4$, we have

$$\begin{aligned}
\|A^\top(\overline{\lambda}_1-\lambda_1)\|_{g_0^{-1}} &\le \|\overline{v}-v\|_{g_0^{-1}} + 3r\|A^\top(\overline{\lambda}-\lambda)\|_{g_0^{-1}} + (\frac{5r}{h} + 2\|v^*\|_{g_0^{-1}})\|\overline{x}-x\|_{g_0} \\
&\le r + \frac{3r^2}{h} + \frac{5r^2}{h} + 2r\|v^*\|_{g_0^{-1}} \le \frac{8r^2}{h} + 2r\|v^*\|_{g_0^{-1}} \le 1
\end{aligned}$$

Hence, we can further simplify it to

$$
\begin{aligned}
&\|\mathcal{T}(x,v,\lambda)-\mathcal{T}(\overline{x},\overline{v},\overline{\lambda})\|\\
\leq &2.3h\|\overline{v}-v\|_{g_0^{-1}}+2.4rh\|A^\top(\overline{\lambda}-\lambda)\|_{g_0^{-1}}+3.3h\|\overline{x}-x\|_{g_0}\|\overline{v_{1/2}}-A^\top\overline{\lambda}\|_{g_0^{-1}}\\
&+3.2h\|A^\top(\overline{\lambda}_1-\lambda_1)\|_{g_0^{-1}}\\
&+6h(\frac{1}{2}\|\overline{v}-v\|_{g_0^{-1}}+\|A^\top(\overline{\lambda}_1-\lambda_1)\|_{g_0^{-1}})\|\overline{v}_{1/2}-A^\top\overline{\lambda}\|_{g_0^{-1}}\\
&+27h\|\overline{v}_{1/2}-A^\top\overline{\lambda}\|_{g_0^{-1}}^2\|\overline{x}-x\|_{g_0}\\
\leq &(\frac{3r}{h}+\|v^*\|_{g_0^{-1}})(6h\|\overline{v}-v\|_{g_0^{-1}}+9h\|A^\top(\overline{\lambda}_1-\lambda_1)\|_{g_0^{-1}}+31h\|\overline{x}-x\|_{g_0})\\
&+2.4rh\|A^\top(\overline{\lambda}-\lambda)\|_{g_0^{-1}}
\end{aligned}
$$

where we used $\|\overline{v}_{1/2}-A^\top\overline{\lambda}\|_{g_0^{-1}}\leq\frac{3r}{h}+\|v^*\|_{g_0^{-1}}$ and $r\geq h$. Using the bound on $\|A^\top(\overline{\lambda}_1-\lambda_1)\|_{g_0^{-1}}$, we have

$$
\begin{aligned}
&\|\mathcal{T}(x,v,\lambda)-\mathcal{T}(\overline{x},\overline{v},\overline{\lambda})\|\\
\leq &(\frac{3r}{h}+\|v^*\|_{g_0^{-1}})(15h\|\overline{v}-v\|_{g_0^{-1}}+27rh\|A^\top(\overline{\lambda}-\lambda)\|_{g_0^{-1}}+9h(\frac{36r}{h}+2\|v^*\|_{g_0^{-1}})\|\overline{x}-x\|_{g_0})\\
&+2.4rh\|A^\top(\overline{\lambda}-\lambda)\|_{g_0^{-1}}\\
\leq &(\frac{3r}{h}+\frac{1}{4r})(15h\|\overline{v}-v\|_{g_0^{-1}}+30rh\|A^\top(\overline{\lambda}-\lambda)\|_{g_0^{-1}}+9h(\frac{36r}{h}+2\|v^*\|_{g_0^{-1}})\|\overline{x}-x\|_{g_0})\\
\leq &(\frac{3r}{h}+\|v^*\|_{g_0^{-1}})(400r+18h\|v^*\|_{g_0^{-1}})\|(x,v,\lambda)-(\overline{x},\overline{v},\overline{\lambda})\|.
\end{aligned}
$$