Indian Institute of Technology Kanpur

Department of Mathematics and Statistics

Unadjusted Barker's Algorithm on constrained spaces

Project Report for MTH598

Author:
Saee Kamat

Supervisor:
Prof. Dootika Vats

November 23, 2024

Abstract

The Unadjusted Langevin Algorithm is used for sampling from distributions especially when it is computationally intensive for other traditional MCMC algorithms. However, for the distributions which are not log gradient Lipschitz the Unadjusted Langevin Algorithm breaks down. In this project, we studied Unadjusted Barker's algorithm, which overcomes this problem. We further study some existing solutions to the issue of sampling from distributions having constrained support and then extend the Unadjusted Barker's algorithm for one-dimensional cases.

Contents

1 Introduction

Let us consider the problem of sampling from a distribution. For known pdf, inversion method (for exponential distribution) and accept- reject method(for normal distribution) works well. The accept reject method depends on whether we can find a suitable proposal distribution for given target. For normal distribution, t distribution is suitable proposal as the tails of t distribution are fatter than lighter. If have a distribution with light tails than any known distribution, we have to go on hard search of "nice" proposal. Thus, it is important that we have algorithms which do not have this drawback. However for high dimension problems like sampling from 1000-dimensional uniform distribution or posterior distributions occurring in Bayesian statistics, these methods are not effective. Many samples have to be discarded because of the high dimension of the support or the byzantine nature of the distribution. In this case Markov Chain Monte Carlo (MCMC) algorithms are used for sampling.

One of the widely used Markov Chain Monte Carlo algorithms for such tasks is Metropolis Hastings Algorithm (MH). To draw samples from distribution F having density function $f(x)$, the algorithm uses kernel $Q(x, \cdot)$ which has proposal density $q(x, y)$, that is $Q(x, dy) = q(x, y)dy$. Transition density is chosen in such a way that it is easy to sample from it. The algorithm is given as follows:

Algorithm 1 Metropolis Hastings Algorithm

Set $X_n = x$. To obtain X_{n+1}

1. $Y \sim Q(x, \cdot)$ and independently $U \sim U(0, 1)$
2. If
$$U < \min\left\{1, \frac{f(y)q(y, x)}{f(x)q(x, y)}\right\}$$
then set $X_{n+1} = y$
3. Else set $X_{n+1} = x$

Observe that, here the Markov chain updates new step with the acceptance ratio

$$\alpha(x, y) = \min\left\{1, \frac{f(y)q(y, x)}{f(x)q(x, y)}\right\}$$

which depends on the choice of kernel density $q(x, y)$ chosen, which is the decisive factor in performance of the algorithm. In order to sample from the regions of f which have high probability, and hence get better proposals, we need the kernel Q to capture this informa-

tion from f. In Metropolis Adjusted Lagevin Algorithm (MALA), the mean gets moved in the direction of $\nabla \log f(x)$, moving the proposal closer to area of higher probability. For example, consider Normal distribution as kernel. In simple MH, $Q(x, \cdot) = N(x, h)$ while in MALA it will be $Q(x, \cdot) = N\left(x + h/2\nabla \log f(x), h\right)$.

Though MALA is effective for lower dimensional problems, for high dimensions, calculating gradients in $\nabla f(x)$ and the ratio $\alpha(x, y)$ is numerically intensive. Thus, the "MH filter" gives computational burden while sampling from complicated posteriors. One such example is bayesian logistic regression model. We run MALA updates to find posterior mean of coefficients of regressors in bayesian logistic regression model, to find that they are prohibitively slow.

Suppose we have $i = 1, 2, 3 \cdots, N$ observations of dependent binary variable Y and $j = 1, 2, \cdots, K$ regressor, i.e. $X_i = (X_{1i}, X_{2i}, X_{3i}, \cdots, X_{Ki})$. Then Bayesian logistic model is given by:

$$Y_i \mid X_i \sim \left(\frac{1}{1 + e^{-X_i\beta}}\right) \quad \beta \sim N(0, I_K)$$

For conciseness of notation, let

$$p_i = \frac{1}{1 + e^{-X_i\beta}}$$

We run MALA chain to find posterior mean of β. The joint distribution of y given x, β is $g(y_i \mid y_i, p_i)$. The prior distribution of β is $h(\beta)$. By baye's theorem, we can write posterior distribution of β as $f(\underline{\beta} \mid \underline{y}, \underline{x}) \propto g(\underline{y} \mid \underline{x}, \underline{p}) \cdot h(\beta)$. The calculations are as follows:

$$g(y_i \mid y_i, p_i) = \prod_{i=1}^{n} (p_i)^{y_i} (1 - p_i)^{(1-y_i)}$$

$$h(\beta) = (2\pi)^{-k/2} k^{-\frac{k}{2}} \exp\left\{-\frac{\beta^T\beta}{2}\right\}.$$

$$f(\underline{\beta} \mid \underline{y}, \underline{x}) \propto g(\underline{y} \mid \underline{x}, \underline{p}) \cdot h(\underline{\beta}).$$

$$f(\underline{\beta} \mid \underline{y}, \underline{x}) \propto \exp\left\{-\frac{\beta^\top\beta}{2}\right\} \prod_{i=1}^{n} (p_i)^{y} (1 - p_i)^{(1-y_i)}$$

To avoid numerical instability, we will do computations in log scale.

$$\log f(\beta) = C - \frac{\beta^T\beta}{2} + \sum_{i=1}^{n} \left(y_i \log p_i + (1 - y_i) \log(1 - p_i)\right)$$

In order to run MALA, we must know $\nabla \log f$. We calculate the $\nabla \log f$ as follows:

$$\nabla \log f(\beta) = -\beta + \sum_{i=1}^{n} \frac{y_i}{p_i} \frac{dp_i}{d\beta} - \sum_{i=1}^{n} \frac{1-y_i}{1-p_i} \frac{dp_i}{d\beta},$$
$$\text{where, } \frac{dp_i}{d\beta} = \frac{X_i e^{-X_i\beta}}{1+e^{-X_i\beta}}$$

Substituting and simplifying we get :

$$\nabla \log f(\beta) = -\beta + \sum_{i=1}^{n} \frac{y_i X_i e^{-X_i\beta} - (1-y_i)X_i}{1+e^{-X_i\beta}}$$

Here we use multivariate normal distribution as kernel $Q(x,)$with mean updating by $\nabla \log f(x)h/2$ and K dimensional identity matrix as variance covariance matrix, with h as tuning parameter. Thus our kernel $Q(x,\cdot)$ looks as follows:

$$Q(x,\cdot) = N\left(x + \frac{h}{2}\nabla \log f(x), I_k\right)$$

We first simulate the data by setting $K = 5$ and $n = 1000$. Let the true values of β be randomly sampled from K dimensional standard multivariate gaussian distribution.

By running the scheme we see that though MALA is effective for lower dimensional problems and small datasets, for high dimensions, calculating gradients in $\nabla f(x)$ and the ratio $\alpha(x,y)$ is numerically intensive. Thus, the "MH filter" gives computational burden while sampling from complicated posteriors. This gives motivation for exploring other kinds of algorithms, which will not be compuationally burdensome, yet they will sample by taking proposal which are informed by location of high probability regions of f.
One such algorithm is Unadjusted Langevin Algorithm (ULA). The term "unadjusted" comes from the fact that, this algorithm does not use the MH filter for making updates. The algorithm is based on simulating Langevin Dynamics, which is represented by a Stochastic Differential Equation(SDE) that describes motion of particle, influenced by both deterministic and stochastic forces. It is used for the purpose of sampling because the movement of the particle follows the target distribution if we run the chain for sufficiently large number of steps (i.e as $n \to \infty, X \sim F$, where F is target distribution). These are also called as diffusion processes. A general SDE is given by:

$$dY_t = \mu(Y_t)\,dt + \sigma(Y_t)\,dW_t, \tag{1}$$

where $\mu(Y_t)$ represents the drift term, $\sigma(Y_t)$ represents the volatility term and W_t is called Wiener Process. Here we can think of Y_t playing the role of deterministic force and W_t that of stochastic

Generatlly it is hard to find the exact solution of SDE analytically. That is why, we need numerical schemes to discretize and approximate the process. As dt represents the "infinitesimally small" time, in order to discretize the process, we replace dt by Δt, a small "time step". Simillary, dY_t, the infinitesimally small difference in Y can be replaced by $\Delta Y = Y_{t+1} - Y_t$ and simillary for dW_t by ΔW. As Wiener process is also a function of t, it is approximated by $z\sqrt{\Delta t}$ where $z \sim N(0,1)$. This discritization is called as Euler Maruyama discretization (EM).The update of this scheme is given by following algorithm:

Algorithm 2 Euler-Maruyama Discretization

Set $X_0 = y_0$ and fix a time discretization step Δt.
For n in 1 to N do :

1. Draw $\nu \sim N(0,1)$.
2. $X_{n+1} = X_n + \mu(X_n)\Delta t + \sqrt{\Delta t}\sigma(X_n)v_n, \quad X_0 = y_0$

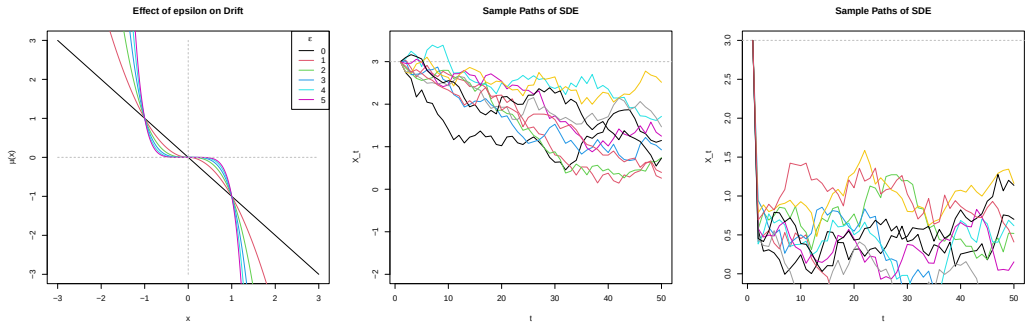
However, the scheme has various shortcomings like it does not approximate the dynamics to desired degree of acurracy, when drift is not globally Lipschitz it leads to numerical instability and it causes issues in different domains of application. Hence, the paper **Skew-symmetric schemes for stochastic differential equations with non-Lipschitz drift: an unadjusted Barker algorithm** Livingstone et al. (2024) proposes a new skew symmetric scheme that is highly robust and numerically stable for SDEs with globally non-lipschitz drift The theoretical results also show that such simulation is approximating the dynamics of diffusion process well.

In the next section, we first understand the intuition behind the scheme in one dimension, followed by it's extension to general d dimensional problem. Then we understand the main assumptions for this scheme and state the important theoretical results. We then compare this scheme with ULA for one dimensional target distribution and for multidimensional case with target distribution in poisson random effects model.

2 The Skew Symmetric Scheme

2.1 Intuition and Algorithms

In Euler-Maruyama Scheme, the size of jump in each update which is $X_{n+1} - X_n$, is dependent on drift $\mu(X_t)$ and volatility $\sigma(X_t)$. We can observe that drift is expected value of X_{t+1} given X_t, while $\sigma(X_t)$ is the variance. Hence for a SDE with constant volatility, and polynomially growing drift, EM scheme becomes numerically unstable. Figure 1a shows such non Lipschitz drifts which are of the form $\mu(x) = -\operatorname{sgn}|x|^{1+\epsilon}, x \in \mathbb{R}, \epsilon \geq 0$, and Figure 1b and 1c show sample paths for lipschitz and globally non Lipschitz drift SDEs.



(a) Non Lipschitz drifts (b) Lipschitz Drift, $\epsilon = 0$ (c) Drift with $\epsilon = 4$

Figure 1: Sample paths for SDE having drifts growing polynomially from left to right

We can observe that as drift grows polynomially, sample paths vary a lot. But sample paths lie below the starting (grey dotted) line, because of negative mean of the process. Observe that, drift pulls the paths towards negative side of the number line on Y axis. Hence, such kind of drifts are called "pointing inward" or "confining". This nature of drifts, is the cause behind numerical instability.

The proposed skew symmetric numerical scheme, considers such SDEs and makes updates in two steps, for each dimension independently . Fist, the size of the update is calculated without influence of drift, it is either set to +1 or -1 and second, the direction of the update is decided based on probability which depends only on drift (which is $\nabla \log f(x)$) and volatility of the process, at current position X_t. In this way, the chain keeps on moving towards the region with higher probability, without inducing numerical instability. Algorithm 3 describes the skew symmetric scheme in one dimenson:

Algorithm 3 Skew-Symmetric Numerical Scheme for one dimension

Input: Initial point $x \in \mathbb{R}$, number of iterations $N \in \mathbb{N}$, step size $\Delta t > 0$, probability function $p : \mathbb{R} \times \mathbb{R} \to (0, 1)$

Goal: Approximate samples from $Y_{\Delta t}, Y_{2\Delta t}, \ldots, Y_{N\Delta t}$

Output: $X_1, X_2, \ldots, X_N$

1. Set $X_0 = x$
2. **For** $n = 0$ **to** $N - 1$ **do**
   (a) Sample $v \sim N(0, 1)$
   (b) Set
   $$b = \begin{cases} +1 & \text{with probability } p(X_n, v), \\ -1 & \text{otherwise} \end{cases}$$
   (c) Set $X_{n+1} = X_n + \sqrt{\Delta t} \cdot b \cdot \sigma(X_n) v$

Here $p : \mathbb{R} \times \mathbb{R} \to [0, 1]$ is probability function that satisfies following criterion as $\Delta t \to 0$:

$$\partial_v|_{v=0}\, p(x, v) = \frac{\mu(x)}{2\sigma(x)} \sqrt{\Delta t}, \quad \forall x \in \mathbb{R} \tag{2}$$

The above condition is important for the skew symmetric scheme to converge to true diffusion process in weak sense. We will understand more details about this probability function in next subsection.

Now, we extend the same idea to d dimensions in following Algortihm 4. Observe that, the only difference is that updates in each dimension are done independently.

Algorithm 4 Skew symmetric scheme for general dimensions

Input: Initial point $x \in \mathbb{R}^d$, number of iterations $N \in \mathbb{N}$, step size $\Delta t > 0$, probability functions $p_i : \mathbb{R}^d \times \mathbb{R}^d \to (0, 1)$
Goal: Approximate samples from $Y_{\Delta t}, Y_{2\Delta t}, \ldots, Y_{N\Delta t}$
Output: $X_1, X_2, \ldots, X_N$

1. Set $X_0 = x$
2. **For** $n = 0$ to $N - 1$ **do**
   (a) Sample $v = \left(v^1, \ldots, v^d\right) \sim N\left(0, I_d\right)$
   (b) **For** each $i \in \{1, \ldots, d\}$ **do**
   - Set
   $$b^i = \begin{cases} +1 & \text{with probability } p_i(X_n, v), \\ -1 & \text{otherwise} \end{cases}$$
   (c) Set $b = \left(b^1, \ldots, b^d\right)$
   (d) Set $X_{n+1} = X_n + \sqrt{\Delta t} \cdot b \cdot \sigma(X_n)v$

Note that here $\cdot$ represents element wise multiplication and probability function p_i satifies the condition (2) in each of the i^{th} dimension. The conditions and characterisation on these $p_i's$ is done in next subsection which includes the assumptions and main results of the paper.

2.2 Theoretical Assumptions and Results

For this scheme to converge to true diffusion in weak sense, some assumptions are made on drift, volatility and probability function.

1. **Assumption on Drift:** This assumption on drift states that $\exists M > 0, M \in R$ such that,
   $$(\mu(y) - \mu(x)) \cdot (y - x) \leq M^2 \left(1 + \|y - x\|^2\right) \tag{3}$$
   This is called as one sided Lipschitz condition. It ensures that $\|\mu(x)\|$ as $\|x\| \to \infty$, whenever μ is "inward pointing" or "confining" for all $x, y \in \mathbb{R}^d$. This asusmption is made so that drift can be polynomially growing but not unbounded. Hence, it also ensures that the target distribution is also bounded.
2. **Assumption on Volatility:**

(a) For all $i, j \in \{1, \ldots, d\}$, we have $\sigma_{i,j} > 0$. There exists $M > 0, M \in \mathbb{R}$ such that for all $x, y \in \mathbb{R}^d$ we have,

$$\|\sigma(y) - \sigma(x)\|^2 \leq M^2 \left(\|y - x\|^2\right) \tag{4}$$

(b) Diagonal volatility: $\forall i \neq j \in \{1, \ldots, d\}$ and all $x \in \mathbb{R}^d$, we have, $\sigma_{i,j}(x) = 0$. The assumption of diagonal volatility means that volatilises in all dimensions are independent of each other.It is required for skew symmetric scheme to converge to true diffusion process as $\Delta t \to 0$. One sided Lipschitz condition on μ along with Local Lipschitz property of σ both imply that there exists a solution for the SDE (1)

3. **Assumption on probability functions:** For all $i \in \{1, \ldots, d\}$, we have that probabili ty functions $0 < p_i(x, v) < 1$ for all $x, v \in \mathbb{R}^d$. For all $x \in \mathbb{R}^d$, the probability function pi_i satisfies following two conditions,

$$p_i(x, 0) = \frac{1}{2} \quad \text{and} \quad \partial_i^v p_i(x, 0) = \frac{\mu_i(x)}{2\sigma_{i,i}(x)} \sqrt{\Delta t} \tag{5}$$

These conditions are generalisations of the conditions in one dimension as given in (9). It is required for skew symmetric scheme to approximate the diffusion process well at fixed Δt. Based on these, $p(\cdot, \cdot)$ takes following form:

$$p(x, v) = F\left(v \Delta t^{1/2} \frac{\mu(x)}{2f(0)\sigma(x)}\right) \tag{6}$$

where,$v \sim N(0, 1)$, F is cdf corresponding to f, which is pdf of a one dimensional distribution with continuous density which is symmetric under the reflection $x \mapsto -x$, and which satisfies $f(0) \neq 0$. Table 1 shows comparison of pdf $f(x)$, cdf $F(x)$ and probability functions $p(x, \nu)$ for logistic distribution and standard normal distribution.

pdf	**cdf**	**Probability Function**
$f(x) = e^{-x}\left(1 + e^{-x}\right)^{-2}$	$F(x) = \left(1 + e^{-x}\right)^{-1}$	$p(x, v) = \frac{1}{1+\exp\left\{-2\Delta t^{1/2} v \frac{\mu(x)}{\sigma(x)}\right\}}$
$f(x) = (2\pi)^{-1/2} e^{-x^2/2}$	$\Phi(x)$	$p(x, v) = \Phi\left(\sqrt{\frac{\pi}{2}} \Delta t^{1/2} v \frac{\mu(x)}{\sigma(x)}\right)$

Table 1: Comparison of pdf, cdf, and Probability Function for Logistic and Standard Normal Distributions

Under the assumption that $p(X_n, \cdot)$ is a CDF of a centred and symmetric real-valued random variable, the product $b_n v_n$ is known as a skew-symmetric random variable.Azzalini and Regoli (2012) That is why, scheme is called skew symmetric. The Figure 2 shows distribution of $b_n v_n$ when skew symmetric scheme used with CDF of logistic distribution.

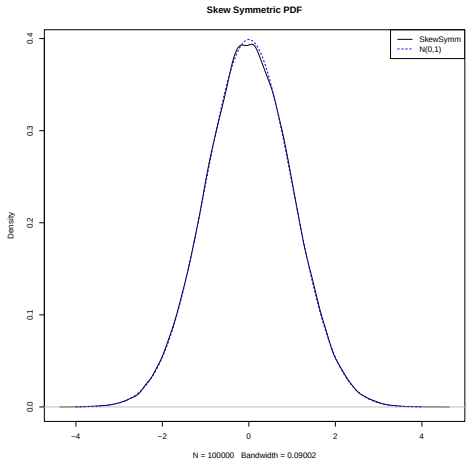


Figure 2: Skew Symmetric Distribution for $b_n v_n$

4. **Assumption on jump size:** An additional assumption of a version of one sided Lipschitz inequality is imposed on jump size of the process ξ. There exists $C > 0$ and $m \in \mathbb{N}$ such that for all $k \leq 4, x, \xi \in \mathbb{R}^d$,

$$\left| q_i^{(k)}(x, \xi) \right| \leq C \left(1 + \|x\|^m + \|\xi\|^m\right) \tag{7}$$

where $q_i^{(k)}$ denotes any partial derivative of q_i with respect to any coordinates in $\mathbb{R}^d \times \mathbb{R}^d$. This assumption is made to underline that the this scheme is only dealing with the density functions, where jump sizes will be bounded and they will not shoot up.

Considering all the above assumptions, following are main theoretical guarantees about the skew symmetric algorithm:

1. The skew-symmetric scheme converges weakly to the true diffusion in $O(\Delta t)$, meaning it is weak order 1 as the step-size $\Delta t \to 0$.

2. The skew symmetric scheme has an invariant probability measure $\pi_{\Delta t}$ and converges to this measure at a geometric rate in total variation distance for any fixed step-size $\Delta t \in (0, 1)$.

3. Let the limiting measure of the skew-symmetric scheme be $\pi_{\Delta t}$, and the true diffusion be π. As the step-size $\Delta t \to 0$, the bias between $\pi_{\Delta t}$ and π vanishes at a rate $O(\Delta t)$.

4. Under the assumptions on the drift and volatility given above, Theorem by Milstein and Tretyakov generalises beyond the globally Lipschitz case.

In the next subsection we consdier the example Overdamped Langevin Dynamics, to understand Unadjusted Barker Algorithm and compare it with Unadjusted Langevin Algorithm with one dimensional target distribution.

2.3 Example of Overdampled Langevin Dynamics

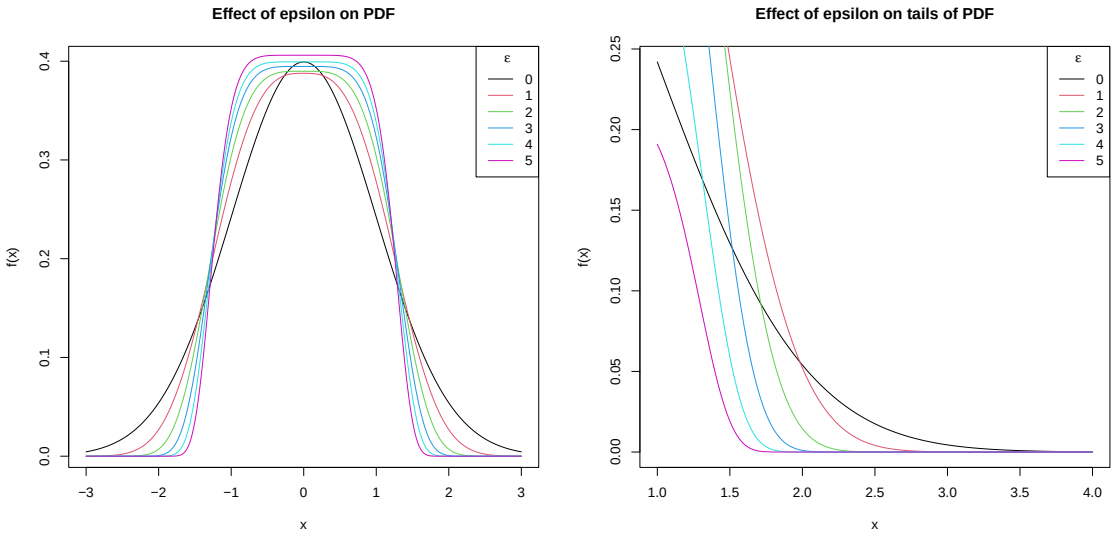
Overdamped Langevin Dynamics is SDE which is given by

$$dX_t = \nabla \log \pi(X_t)\, dt + \sqrt{2}\, dW_t \tag{8}$$

where, $\mu(x) = \nabla \log \pi(x), \sigma(x) = \sqrt{2}$, W_t is Wiener Process and π is underlying probability density function in $\mathbb{R}$. Under mild regularity conditions this diffusion is known to converge as $t \to \infty$ to the law induced by π, in total variation distance. For numerical experiments, we consider target distributions of the form,

$$\pi(x) = \frac{1}{Z} \exp\left\{-c|x|^{2+\varepsilon}\right\}, \tag{9}$$

for some $\varepsilon, c > 0$ where $Z = \int_{\mathbb{R}} \exp\left\{-c|y|^{2+\varepsilon}\right\} \mathrm{d}y$. Observe that π has lighter tails than any Gaussian distribution. We consider this example as π has very light tails, the one sided Lipschitz condition might still hold, but global Lipschitz condition does not. Figure 1a shows the drifts of $\pi(x)$ as function of ϵ. We can see shapes of the π with respect to ϵ, and comparison on their tails in the Figure 3 below.



(a) Shape of pdf for different ϵ (b) Comparison of tails for different ϵ

Figure 3: Visualising effect of ϵ on π

When Overdamped Langevin Dynamics, is approximated by Euler Maruyama discretization without using Metropolis Hastings filter, is called Undajusted Langevin Algorithm(ULA). The update for n^{th} step and fixed time step Δt is given by Algortithm 2. We can calculate the drift for this process as follows:

$$
\begin{aligned}
&\text{For } \pi(x) = \frac{1}{Z} \exp\left\{-\frac{|x|^2 + \varepsilon}{2 + \varepsilon}\right\}, \\
&\Rightarrow \log \pi(x) = -\log Z - \frac{|x|^2 + \varepsilon}{2 + \varepsilon}, \\
&\Rightarrow \nabla \log \pi(x) = -|x|^{1+\varepsilon} \cdot \nabla(|x|) \\
&\Rightarrow \nabla \log \pi(x) = -|x|^{1+\varepsilon} \ \text{sign}(x), \\
&\Rightarrow \nabla \log \pi(x) = -\,\text{sgn}(x) \cdot |x|^{1+\varepsilon} = \mu(x)
\end{aligned}
$$

Algorithm 5 Euler-Maruyama Discretization fo $\pi(x)$

Set $X_0 = y_0$ and fix a time discretization step Δt.
For n in 1 to N do :

1. Draw $\nu \sim N(0,1)$.
2. Set $\mu(x) = -\operatorname{sgn}(x) \cdot |x|^{1+\epsilon}$
3. $X_{n+1} = X_n + \mu(X_n)\,\Delta t + \sqrt{\Delta t}\sigma(X_n)\,v_n$

Refer to appendix to see animation for convergence of ULA to equilibrium distribution for $\epsilon = 0$, i.e. standard normal distribution. Observe that here, drift is globally Lipschitz. When Overdamped Langevin Dynamics is discretized by using skew symmetric scheme, the algorithm is refered to as Unadjusted Barker Algorithm (UBA). It is "unadjusted" since MH filter is not applied and as the idea of making step size independent of drift, and dircetion is decided by suitable probability function, it is a "Barker" algorithm. The probability function here is calculated based on Logistic distribution density and its CDF. Thus we get $p(x, v)$ as given in Table 1 where $\mu(x)$ and $\sigma(x)$ are plugged in according to SDE. For the same π as mentioned above, we get following algorithm:

Algorithm 6 Unadjusted Barker Algorithm for $\pi(x)$

Input: Initial value $X_0 = x$, number of iterations N, step size Δt
Initialize: $x_0 = x$ **For** $n = 0$ to $N-1$ **do**:

1. Draw $\nu_n \sim N(0,1)$
2. Draw $U_n \sim N(0,1)$
3. Set $\mu(x_n) = -\operatorname{sgn}(x_n) \cdot |x_n|^{1+\varepsilon}$
4. Compute
$$p(x_n, \nu_n) = \left[1 + \exp\left\{-2\sqrt{\Delta t} \cdot \nu_n \cdot \frac{\mu(x_n)}{\sqrt{2}}\right\}\right]^{-1}$$
5. If $U_n \leq p(x_n, \nu_n)$ then set $b_n = 1$
6. Else set $b_n = -1$
7. Update $x_{n+1} = x_n + \sqrt{\Delta t} \cdot b_n \cdot \sigma \cdot \nu_n$

To compare equilibrium distribution of $\pi(x)$ in ULA and UBA with the true $\pi(x)$, at different values of ϵ and Δt, we consider the follwoing experimental setting. Let $\epsilon \in$

$\{3, 1, 0, 1, 2, 3, 4, 5\}$ and $\Delta t \in \{2, 1, 0.1, 0.01, 0.001, 0.0001\}$. Simulations are conducted with 10^5 steps in the process. Figure 4 compares the true distribution $\pi(x)$ (black) with the equilibrium distributions obtained using the Euler-Maruyama scheme (blue) and the Unadjusted Barker algorithm (red). For larger timesteps $\Delta t = 2, 1$ ULA breaks down. For smaller timesteps $\Delta t = 10^{-4}, 10^{-5}$ process moves very slowly, hence UBA and ULA both show poor approximation. For timesteps smaller than this, approximations are better if we run the chain for 10^7 steps as showsn in Figure 5.

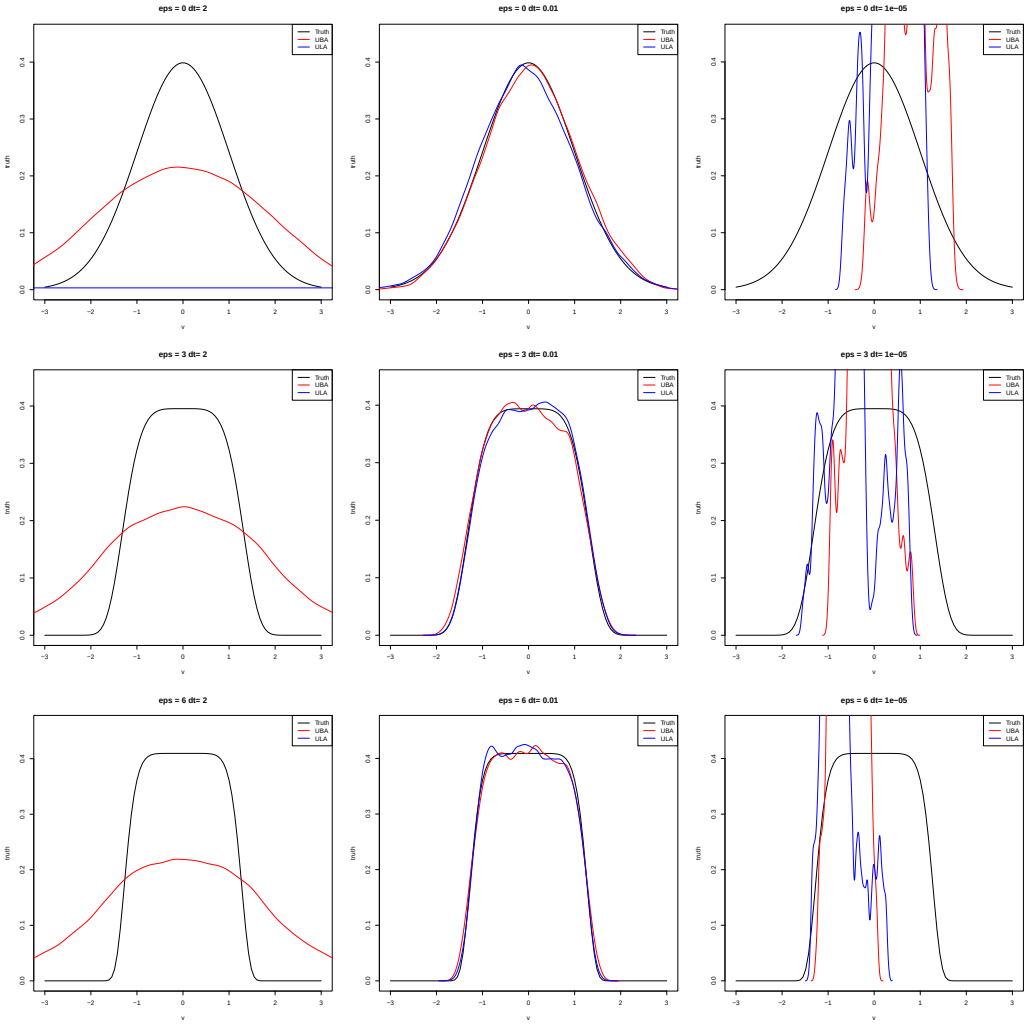


Figure 4: Comparison of equilibrium distrbution

In the next subsection, we compare performance of UBA and ULA for higher dimensional

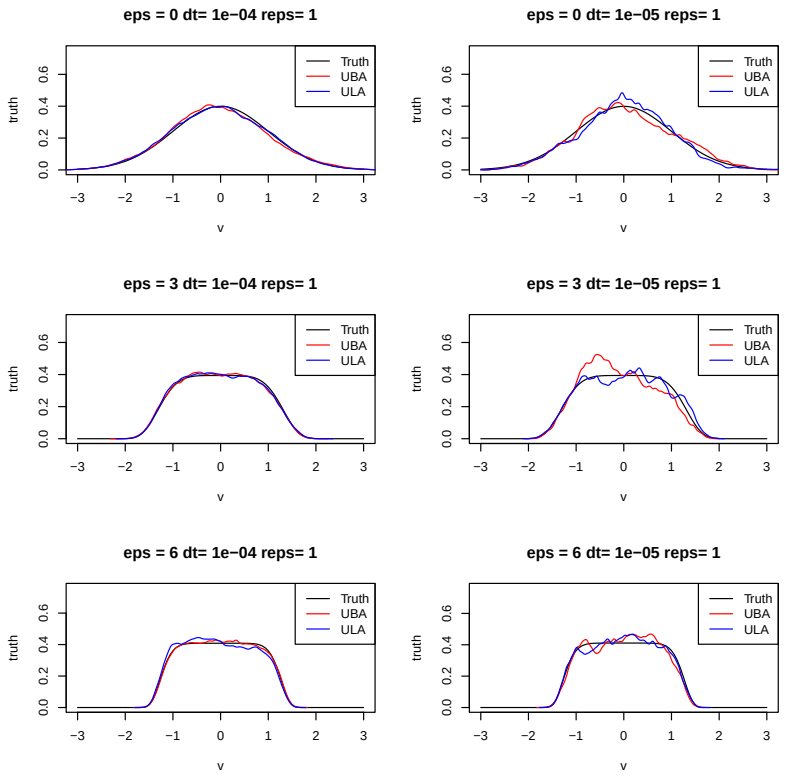


Figure 5: Comparison of equilibrium distrbution when timesteps are very small

target distribution with Poisson random effects model.

2.4 Poisson Random Effects model

Goal of this experiment is to to compare the Euler Maruyama Scheme (ULA) with proposed skew symmetric scheme(UBA) for long time simulation of Overdamped Langevin Dynamics process for posterior distribution of poisson random effects model. The goal of the experiment is to find posterior mean of μ by using the numerical scheme as we cannot analytically find the marginal posterior distribution of μ by integrating over all η_i. The Poisson random effects model is of the following form:

$$\begin{aligned} y_{ij} \mid \eta_i &\sim \text{Poi}\left(e^{\eta_i}\right), & j &= 1, \ldots, J \\ \eta_i \mid \mu &\sim N(\mu, 1), & i &= 1, \ldots, I, \\ \mu &\sim N\left(0, \sigma_\mu^2\right). \end{aligned}$$

We calculate the posterior as follows:

$$\begin{aligned} g(y_{ij} \mid \eta_i) &\propto e^{-e^{n_i}(e^{\eta_i})^{y_{ij}}} \\ \Rightarrow g(y \mid \eta_i) &\propto \exp -J \sum_{i=1}^{I} e^{\eta_i} + \sum_{ij} \eta_i y_{ij} \\ h(\eta \mid \mu) &\propto \exp - \sum_i \frac{(\eta_i - \mu)^2}{2} \\ q(\lambda) &= \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{\mu^2}{2\sigma_\mu^2}} \end{aligned}$$

We define the state vector $x = (\mu, \eta_1, \ldots, \eta_I)$. By using Baye's theorem we get:

$$\begin{aligned} f(x|y) &\propto g(y \mid \eta_1, \cdots, \eta_I, \mu) h(\eta_1, \cdots, \eta_I \mid \mu) q(\mu) \\ \Rightarrow f(x \mid y) &\propto \exp\left\{ -J \sum_{i=1}^{I} e^{\eta_i} - \sum_{i,j} \eta_i y_{ij} + \frac{\sum_i (\eta_i - \lambda)^2}{2} + \frac{\mu^2}{2\sigma_\mu^2} \right\} \\ \Rightarrow f(x \mid y) &\propto e^{-U(x)} \end{aligned}$$

Thus we have a posterior distribution with density $\pi(x) \propto \exp\{-U(x)\}$ and corresponding

potential is: We define the state vector $x = (\mu, \eta_1, \ldots, \eta_I)$, which is $I+1$ dimensional vector. this results in a posterior distribution with density $\pi(x) \propto \exp\{-U(x)\}$ and corresponding potential is,

$$U(x) = J\sum_i e^{\eta_i} - \sum_{i,j} y_{ij}\eta_i + \frac{1}{2}\sum_i (\eta_i - \mu)^2 + \frac{\mu^2}{2\sigma_\mu^2}.$$

Then, the drift in Overdamped Langevin Dynamics process is given by ∇U. We calculate the gradient of $U(x)$ as given below. First differentiating with respect to μ we get:

$$\frac{\partial U}{\partial \mu} = -\sum_i (\eta_i - \mu) + \frac{\mu}{\sigma_\mu^2}$$

Now, differentiating with respect to η_i each we get:

$$\frac{\partial U}{\partial n_i} = Je^{n_i} - \sum_j y_{ij} + (n_i - \mu)$$

Thus, we have our gradient :

$$\nabla U = \left(\frac{\partial U}{\partial \mu}, \frac{\partial U}{\partial \eta_i}\right)$$

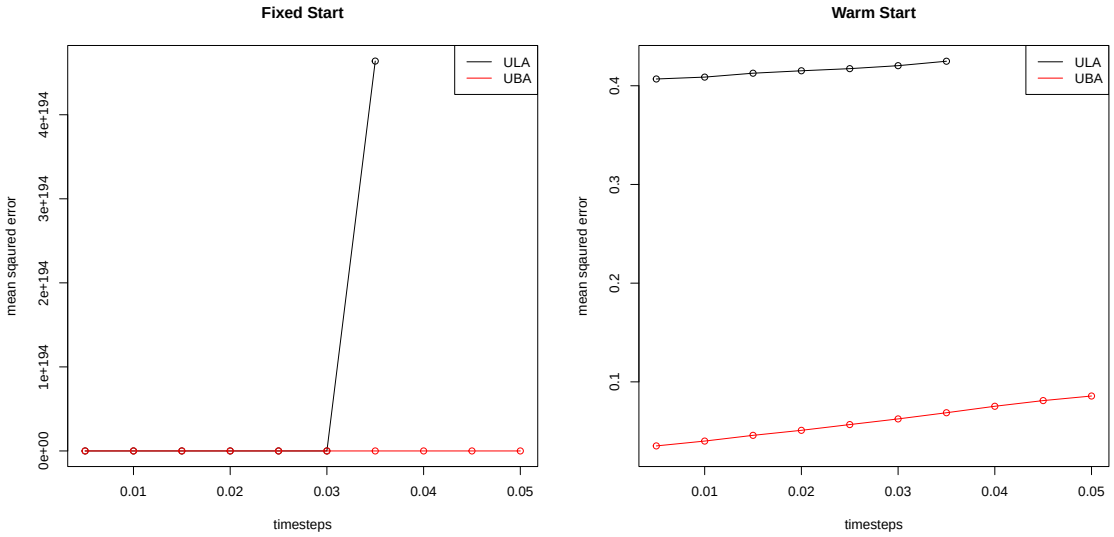
Note that, here drift, $\mu(x) = -\nabla U$.Using this gradient in Algorithm 4 we impliment UBA and similarly ULA.

To compare the two schemes we first generate the synthetically. We generate data y_{ij} for $i = 1, \ldots, I$ and $j = 1, \ldots, J$ were simulated using true parameter values $\mu^* = 5$, with each $\eta_i \sim N(\mu^*, 1)$. To generate the data we set $I = 50, J = 5$ and $\sigma_\mu = 10$.We consider two ways to initialise the scheme. In the "fixed start", we initialise λ at the true value $\mu^* = 5$ and initialize η_i to the same values which are used for generating the data, while in "warm start", we initialise μ by taking a random sample from $N(5, 10^2)$ and initialize η_i to the same values which are used for generating the data. To asses how well the algorithms are working, we use the fact that posterior marginal distribution of μ is known to be centred around μ^*. We calculate Monte Carlo mean squared error by running a chain 100 times,for each time step. This means that our assessment metric is:

$$\frac{\sum_{i=1}^{100}(\hat{\mu_i} - \mu)}{100}$$

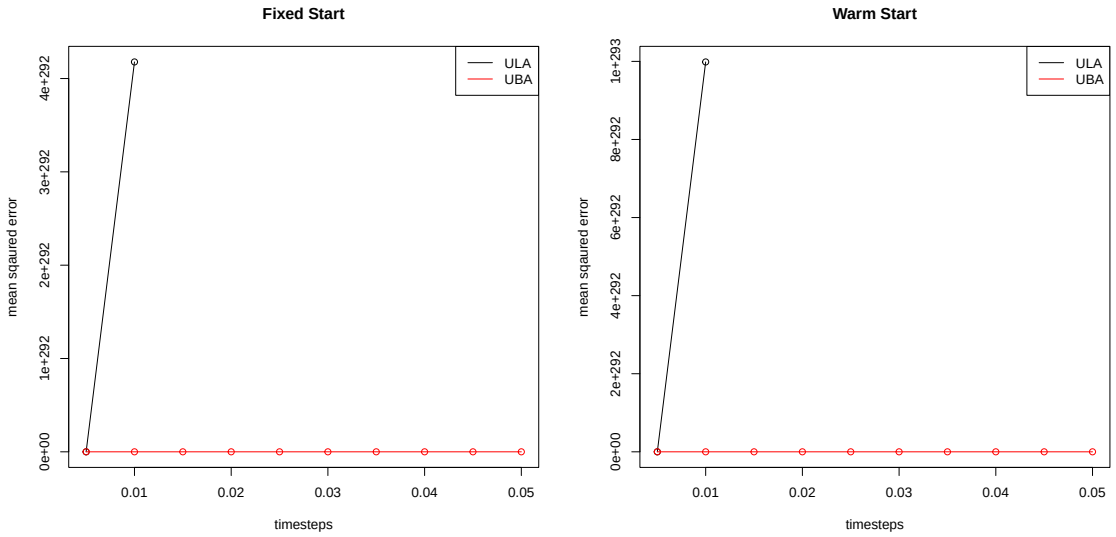
where $\hat{\mu_i}$is the estimate for true parameter in the i^{th} which is calculated by discarding first 10^4 initial steps and then using next 5×10^4 steps to evaluate the mean, in case of UBA,

and taking ergodic average in case of ULA. ULA was run for 5×10^6 steps for each Δt as it takes longer steps to reach equilibrium distribution. Figure 6 shows results for both settings of μ when CDF of logistic distribution is used, while Figure 7 has CDF of N(0,1) as probability function.We observe that ULA breaks down for time steps larger than 0.035 when CDF of logitsic regresison is used as probability function while it breaks down even for smaller time steps when CDF of N (0, 1) is used as probabilty function.



(a) MSE when initialised at the true value μ^* (b) MSE when initialised from sample $N(5, 10^2)$

Figure 6: ULA breaks down for time steps larger than 0.035 when CDF of logitsic regresison is used as probability function



(a) MSE when initialised at the true value μ^* (b) MSE when initialised from sample $N(5, 10^2)$

Figure 7: ULA breaks down even for smaller time steps when CDF of $N(0, 1)$ is used as probabilty function

3 The problem of sampling from constrained space

3.1 The existing solutions to constrained space sampling problem

This section contains a short overview of literature that gives solution to problem of sampling from constrained space. The problem is to sample from $\pi(x) \propto e^{-U(x)}$ where support of $\pi(x)$ is $\mathcal{K}$. The underlying question to which all the papers are trying to find answer is: "Can we somehow change Unadjusted Langevin Algorithm to adapt to the 'geometry i.e. constrained support' of the distribution from which we want to sample?" To do this, we look at variants of gradient descent namely mirrored gradient descent, projected gradient descent,proximal gradient descent and borrow the same idea to convert the problem of sampling from constrained space to unconstrained space. We first understand the Mirrored Langevin Algorithm and implement it for the case of exponential. We then get the overview of some other existing for for sampling from constrained spaces.

3.1.1 Mirrored Langevin Algorithm (MLA)

This Mirrored Langevin Algorithm Hsieh et al. (2018) is Unadjusted Langevin algorithm parallel of mirrored gradient descent algorithm. Figure 8 shows the working of mirrored gradient descent algorithm. In MLA we convert the problem to unconstrained space by mirror map and then instead of making the update on gradient step, we perform ULA update in the dual space and then take inverse of the mirror map $\nabla\phi$ to get the sample from primal space, i.e. our target distribution's support. A mirror function, is the function that has domain as the constrained space K, but has range of $\mathbb{R}^d$. If $\mathcal{K} \subseteq \mathbb{R}^n$ an open convex set, and and $\phi : \mathcal{K} \to \mathbb{R}$ be a strictly convex function. We call ϕ a mirror map if it additionally satisfies:

1. ϕ is differentiable on $\mathcal{K}$.
2. The range of $\nabla\phi : \mathcal{K} \to \mathbb{R}$ is all of $\mathbb{R}^n$
3. $\nabla\phi$ blows up on the boundary of $\mathcal{K}$.

Two prominent examples are:(James.R.Lee (2016))

1. $\mathcal{K} = \mathbb{R}^n$ and $\phi(x) = \|x\|^2$ with $\nabla\phi(x) = 2x$
2. $\mathcal{K} = \mathbb{R}^n_{++} = \{x \in \mathbf{R}^n : x_1, \ldots, x_n > 0\}$ and $\phi(x) = \sum_{i=1}^n x_i \log x_i$ with

$$\nabla\phi(x) = (1 + \log x_1, \cdots, 1 + \log x_n)$$

Some other assumptions made on mirror map and target distribution are as follows:

1. ϕ is closed, proper, and has positive definite hessian matrix i.e. $\nabla^2\phi > 0$ on support of target distribution which is R^d and ϕ is twice differentiable and its second derivative is continuous.
2. ϕ is Legendre Type function, i.e. if it is differentiable and strictly convex in $\int(\mathcal{K})$, and has gradients that become unbounded as we approach the boundary of $\mathcal{K}$. This assumption is required as the Fenchal dual(convex conjugate) of ϕ is same as inverse of $\nabla\phi$, which is required to convert the problem back from dual to primal space.
3. ϕ has to be 1-strongly convex and ϕ should also be 1-stronglgy convex mirror map.
4. All measures (target and push forward) have finite second moments.

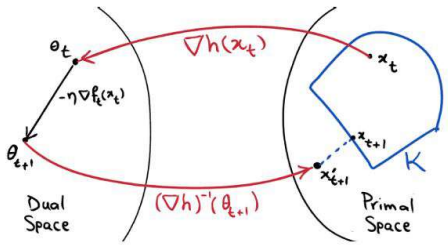


Figure 8: The four basic steps in each iteration of the mirror descent algorithm MirrorDescent (2020)

5. All measures vanish on sets with Hausdorff dimension at most $d-1$.

The algorithm has $\tilde{O}(\epsilon^{-2}d)$ rate of convergence which is much better than rate of convergence of other algorithms which is $\tilde{O}(\epsilon^{-6}d^5)$.This method can also be implemented in stochastic verison i.e. algorithm in minibatch setup as well. Though, there is no specific assumption on nature of potential U is made, performance of the algorithm heavily depends on ϕ. However, there is no explicit formulation for ϕ is provided for any $\mathcal{K}$ in the paper.

Algorithm 7 Mirrored Langevin Algorithm

Input: step size $\Delta t > 0$, initial state $X_0 \in \mathcal{K}$

For $n = 0, 1, 2, \ldots N-1$

1. Draw $\nu \sim \mathcal{N}(0, I_d)$
2. $Y_n = \nabla\phi(X_n)$
3. $Y_{n+1} = Y_n - \Delta t \nabla^2\phi^{-1}(X_n)[\nabla U(X_n) + \nabla \log\det \nabla^2\phi(X_n)] + \sqrt{2\Delta t}\nu$
4. $X_{n+1} = (\nabla\phi)^{-1}(Y_{n+1})$

3.1.2 MLA for Exponential distribution

After understanding the generalised version of MLA, we now implement it for a one dimensional tagret distribution to study the possible drawback of the numerical scheme. Consider the problem of sampling from $\pi(x) = \lambda e^{-\lambda x}$. We implement MLD as given in algorithm 8 with mirror function as entropic function i.e. $\phi(x) = x\log x$. Thus we first calculate terms required in the scheme according to this mirror function.

$$\begin{aligned}
\phi(x) &= x\log x \\
\Rightarrow \nabla\phi(x) &= 1 + \log x \\
\Rightarrow \nabla^2\phi(x) &= \frac{1}{x} \\
(\nabla\phi)^{-1}(x) &= e^{x-1} \\
\Rightarrow (\nabla^2\phi)^{-1}(x) &= x \\
\nabla U(x) &= \lambda \\
\nabla \log\det \nabla^2\phi(x) &= \frac{-1}{x}
\end{aligned}$$

To take point X_n in primal space to dual space, i.e. from constrained to unconstrained problem, in the n^{th} iterate, we first use the mirror map to get $Y_n = \nabla\phi_1(X_n) = 1 + \log X_n$. The Langevin update in the dual space as given in algorithm 8 is

$$Y_{n+1} = Y_n - \Delta t \nabla^2\phi_1^{-1}(X_n)[\nabla U(X_n) + \nabla \log\det \nabla^2\phi_1(X_n)] + \sqrt{2\Delta t}\nu$$

where ν is draw from standard Gaussian distribution. Substituting (17) to (20) in this

equation we get following discretized update:

$$Y_{n+1} = Y_n - \Delta t X_n(\lambda - \frac{1}{X_n}) + \sqrt{2\Delta t}\nu$$

We take the inverse of gradient of mirror function at Y_{n+1} to get the next sample X_{n+1}, i.e. $X_{n+1} = (\nabla\phi_1)^{-1}(Y_{n+1}) = e^{Y_{n+1}-1}$. In this way MLD for exponential distribution simplifies to the numerical scheme written in algorithm 8.

Algorithm 8 Mirrored Langevin Algorithm with $\phi_1(x) = x\log x$ for $\pi(x) = \lambda e^{-\lambda x}$

Input: step size $\Delta t > 0$, initial state $X_0 \in \mathcal{K}$
For $n = 0, 1, 2, \ldots N-1$

1. Draw $\nu \sim \mathcal{N}(0, I_d)$
2. $Y_n = 1 + \log x$
3. $Y_{n+1} = Y_n - \Delta t\, X_n \left(\lambda - \frac{1}{X_n}\right) + \sqrt{2\Delta t}\nu$
4. $X_{n+1} = e^{Y_{n+1}}$

Running this chain for various timesteps "dt" and steps "N", we comapre MLD with AugUBA in figure 9.
Simillary, we implement MLD using another mirror function $\phi_2(x) = -\log x$. Calculating the required terms first to run the scheme, we get follwing expressions.

$$\begin{aligned}
\phi_2(x) &= -\log x \\
\Rightarrow \nabla\phi_2(x) &= \frac{-1}{x} \\
\Rightarrow \nabla^2\phi_2(x) &= \frac{1}{x^2} \\
(\nabla\phi_2(x))^{-1}(x) &= \frac{-1}{x} \\
\Rightarrow (\nabla^2\phi_2(x))^{-1} &= x^2 \\
\nabla \log\det \nabla^2\phi_2(x) &= \frac{-2}{x}
\end{aligned}$$

To take point X_n in primal space to dual space, i.e. from constrained to unconstrained problem, in the n^{th} iterate, we first use the mirror map to get $Y_n = \nabla\phi_2(X_n) = \frac{-1}{X_n}$. The

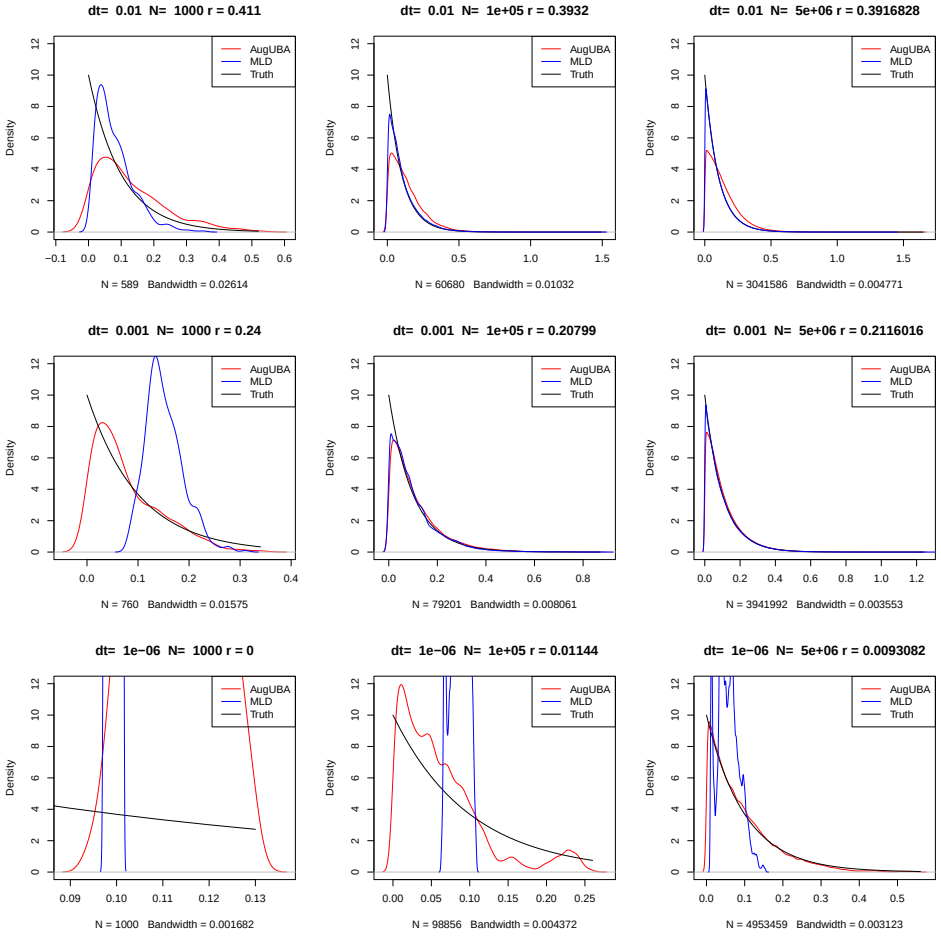


Figure 9: Comparison of MLD with Augmented UBA and Truth for exponential distribution when $\phi_1(x) = x \log x$

Langevin update in the dual space as given in algorithm 8 is

$$Y_{n+1} = Y_n - \Delta t \nabla^2 \phi^{-1}(X_n)[\nabla U(X_n) + \nabla \log\det \nabla^2 \phi(X_n)] + \sqrt{2\Delta t}\nu$$

where ν is draw from standard Gaussian distribution. Substituting (32) to (37) in this equation we get following discretized update:

$$Y_{n+1} = Y_n - \Delta t X_n^2 \left(\lambda - \frac{2}{X_n}\right) + \sqrt{2\Delta t}\nu$$

We take the inverse of gradient of mirror function at Y_{n+1} to get the next sample X_{n+1}, i.e. $X_{n+1} = (\nabla \phi_2)^{-1}(Y_{n+1}) = (-Y_{n+1})^{-1}$. In this way MLD for exponential distribution simplifies to the numerical scheme written in algorithm 9.

Algorithm 9 Mirrored Langevin Algorithm with $h(x) = \log x$ for $\pi(x) = \lambda e^{-\lambda x}$

Input: step size $\Delta t > 0$, initial state $X_0 \in \mathcal{K}$
For $n = 0, 1, 2, \ldots N-1$

1. Draw $\nu \sim \mathcal{N}(0, I_d)$
2. $Y_n = \dfrac{-1}{X_n}$
3. $Y_{n+1} = Y_n - \Delta t X_n^2 \left(\lambda - \dfrac{2}{X_n}\right) + \sqrt{2\Delta t}\nu$
4. $X_{n+1} = \dfrac{-1}{Y_{n+1}}$

Running the chain for time steps "dt" and steps "N", we compare it with true distribution and Augmnted UBA. Comparison plots are shown in Figure 10. We can clearly see that MLD with ϕ_2 is biased does not approximate the target distribution at all. As MLD is a variant of Langevin Dynamics, we check for gradient Lipschitzness for the log pdf of push forward measure of the target distribution. In simple words, if X is an exponential random variable with rate λ, and $Y = -\log X$, we want to check whether log of the density of Y, i.e. $\log f(y)$ is Lipschitz or not.
Let pdf of random variable X be $f(x) = \lambda e^{-}\lambda x$, having cdf $F(x)$ Let $Y = -\log X$. Let $g(y)$ and $G(y)$ denote pdf and cdf of random variable Y. Now we derive $g(y)$ the pdf Y as

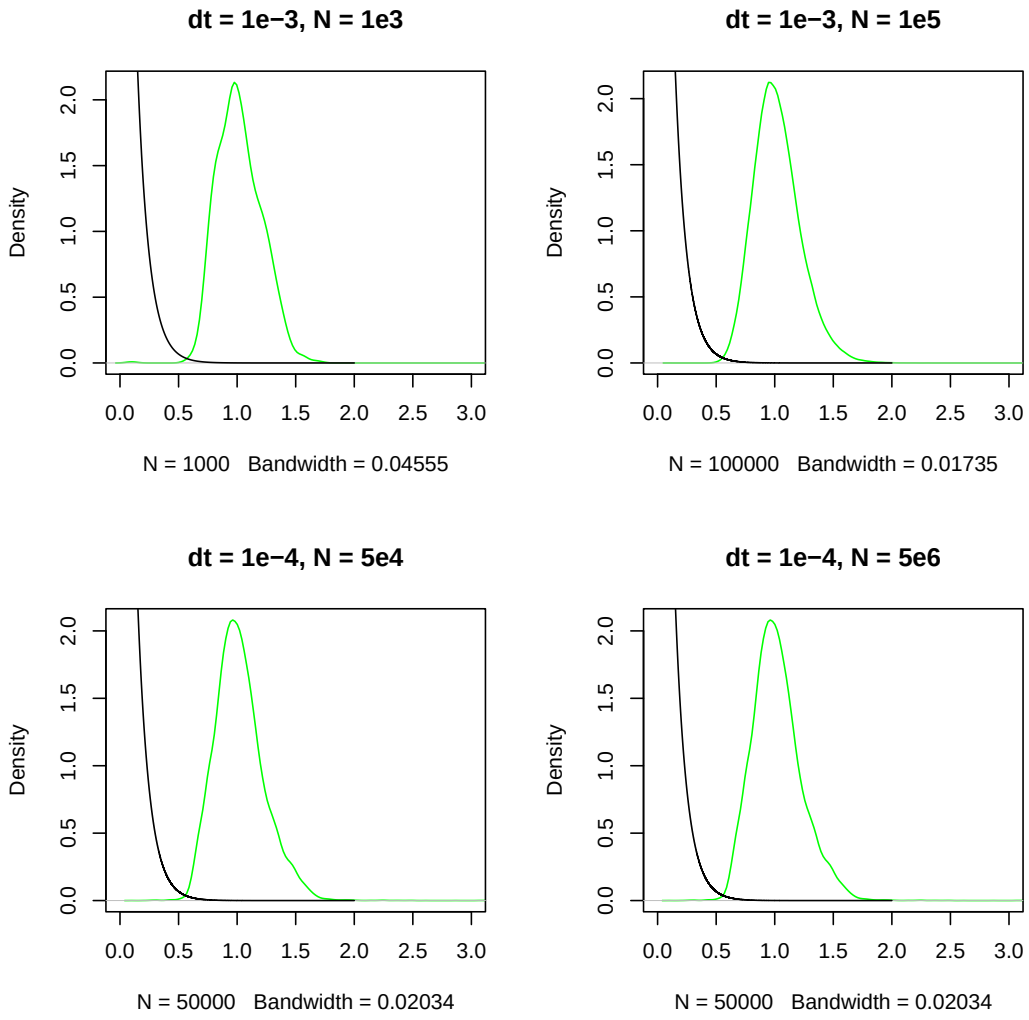


Figure 10: MLD (green) with mirror map $\phi_2(x) = -\log x$ does not approximate target distribution (black) well

follows:

$$\begin{aligned} G(y) &= P(Y \leq y) \\ &= P(-X \log X \leq y) \\ &= P(\log X \geq -y) \\ &= P(X \geq e^{-y}) \\ &= 1 - P(X \leq e^{-y}) \\ &= 1 - F(e^{-y}) \end{aligned}$$

Differentiating both sides with respect to y we get,

$$\begin{aligned} g(y) &= e^{-y} f(e^{-y}) \\ \Rightarrow g(y) &= e^{-y} \lambda e^{-\lambda e^{-y}} \end{aligned}$$

Now to check Lipschitzness of gradient of the log,

$$\begin{aligned} \log g(y) &= \log \lambda - y - \lambda e^{-y} \\ \Rightarrow -\log g(y) &= -\log \lambda + y + \lambda e^{-y} \\ \Rightarrow -\nabla \log g(y) &= 1 - \lambda e^{-y} \\ \nabla \tilde{U}(y) &= 1 - \lambda e^{-y} \end{aligned}$$

Using roof top theorem we will find bounds to show Lipschitz property. We first expand e^{-x} about y using first order Taylor series expansion.

$$\begin{aligned} e^{-x} &\geq e^{-y} + \nabla \left(e^{-x}\right)_y (x - y) \\ \Rightarrow -e^{-x} &\leq -e^{-y} + (x - y)e^{-y} \\ \Rightarrow e^{-y} - e^{-x} &\leq (x - y)e^{-y} \\ \text{Since, } |\nabla \tilde{U}(x) - \nabla \tilde{U}(y)| &= |1 - \lambda e^{(} - x) - 1 + \lambda e^{-y}| \\ &= |\lambda||e^{-y} - e^{-x}| \\ &\leq \lambda |e^{-y} - e^{-y} + (x - y)e^{-y}| \\ &= \lambda |(x - y)e^{-y}| \end{aligned}$$

Now we use the result that if first derivative of a function $h(x)$, $h'(x)$ is not bounded, then

$h(x)$ is not Lipschitz. Observe that $-\lambda e^{-x}$ is not bounded, which is derivative of $\nabla\tilde{U}(x)$. Thus, $\nabla\tilde{U}(x)$ is not Lipschitz function.
If we try to do the same analysis for mirrror function $\phi_1(x) = x\log x$, we will not be able to get the closed form of pushforward measure of x, i.e. pdf of $Y = X\log X$. Hence, we may not be able to always answer the question:
If X has pdf $\pi(x) = e^{-U(x)}$ on its constrained support, such that, $\nabla U(x)$ is gradient Lipschitz, then is the push forward measure of X, i.e., pdf of $Y = \phi(X)$, given by , $\pi'(x) = e^{-W(y)}$ on the unconstrained support, is Lipschitz?
Getting answer to this question is important as Mirrored Langevin Algorithm is a variant of Langevin Algorithm, which works well only when the potential of target distribution is gradient Lipschitz. Apart from this drawback, this numerical experiment also illustrates that performance of this numerical scheme is heavily dependent on the choice of mirror map $\nabla\phi$. It can also become computationally intensive as calculating inverse of the hessian matrix of ϕ can be cumbersome for high dimensional target density and some choices of mirror map. Thus we need other numerical schemes which do not have such drawbacks.

3.1.3 Metropolis adjusted Mirrored Langevin Algorithm (MAMLA)

The Metropolis Adjusted Mirrored Langevin Algorithm Srinivasan et al. (2024) is Mirrored Langevin with MH filter with only assumption of "self concordance" on the potential U. The scheme is unbiased relative to target distribution. This is highlight of the algorithm as MLA or other discretizations have asymptotic bias. The scheme generated a markov chain that is reversible unlike MLA. However, the disadvantage is that because of the MH filter, algorithm becomes computationally intensive. Another disadvantage is that assumption of Lipschitz continuity is also imposed along with self concordance and relative smoothness, and authors have not checked the performance of the scheme when these assumptions are not met.
In MAMLA algorithm, for the mirror function ϕ, we define convex conjugate ϕ^* as:

$$\phi^*(y) = \max_{x\in\mathcal{K}} \langle x, y\rangle - \phi(x)$$

Inverse of the mirror map $\nabla\phi$ is $\nabla\phi^*$. For the smooth transition from primal space (i.e. the original restricted support) to dual space (i.e. the unrestricted $\mathbb{R}^d$) some assumptions are made on ϕ. They are self concordance, relative complexity and smoothness and Lipschitz continuity. After making a usual ULA update, we generate point in the dual space z'. After taking the inverse $\nabla\phi*$ to get z, we accept this proposal with probability $p_{\text{accept}}(z; x_n)$ This

is the step of MH filtering. The acceptance probability is given by:

$$p^{\mathbf{P}}_{\text{accept}}(z;x) = \min\left\{1, \frac{\pi(z)p_z(x)}{\pi(x)p_x(z)}\right\}$$

. Here $\mathbf{P}$ is the be a Markov chain which defines a collection of proposal distributions at each $x \in \mathcal{K}$ with densities $\mathcal{P} : x \in K$. For $\mathcal{N}(x;\mu,\Sigma)$, we have proposal density $p_x(z)$ for any $z \in \mathcal{K}$ defined as:

$$\begin{aligned} p_x(z) &= \mathcal{N}\left((\nabla\phi^*)^{-1}(z); \nabla\phi(x) - \Delta t \cdot \nabla U(x), 2\Delta t \cdot \nabla^2\phi(x)\right) \cdot \left|\det J\left(\nabla\phi^*\right)^{-1}(z)\right| \\ &= \frac{\det \nabla^2\phi(z)}{(4\Delta t\pi)^{d/2} \cdot \sqrt{\det \nabla^2\phi(x)}} \exp\left(-\frac{\|\nabla\phi(z) - \nabla\phi(x) + \Delta t \cdot \nabla U(x)\|^2_{\nabla^2\phi(x)^{-1}}}{4\Delta t}\right) \end{aligned}$$

Keeping these notations and assumptions in mind, we have following MAMLA algorithm:

Algorithm 10 Metropolis-adjusted Mirror Langevin algorithm (MAMLA)

Input: Potential $U : \mathcal{K} \to \mathbb{R}$, mirror function $\phi : \mathcal{K} \to \mathbb{R} \cup \{\infty\}$, iterations N, initial distribution π, step size $\Delta t > 0$.**Initialize** $x_0 \sim \pi$
For $n = 0$ to $N - 1$ **do**

1. Sample a random vector $\nu \sim N(0, I)$
2. Generate proposal

$$z = \nabla\phi^*\left(\nabla\phi(x_n) - \Delta t \cdot \nabla U(x_n) + \sqrt{2\Delta t \cdot \nabla^2\phi(x_n)}\nu\right)$$

3. Compute acceptance ratio

$$p_{\text{accept}}(z;x_n) = \min\left\{1, \frac{\pi(z)p_z(x_n)}{\pi(x_n)p_{x_n}(z)}\right\}$$

4. Sample $U \sim \text{Unif}(0,1)$
5. **If** $U \leq p_{\text{accept}}(z;x_n)$ **then**

$$x_{n+1} = z$$

6. **Else**

$$x_{n+1} = x_n$$

3.1.4 Sampling from a log-concave distribution with Projected Langevin Monte Carlo

The Projected Langevin algorithm Bubeck et al. (2018) is Langevin Dynamics parallel of projected gradient descent algorithm. In the projected gradient descent, we first make the usual gradient descent update, which is then followed by taking the projection on to the restricted set. This means that, whenever the updated point goes out of the constrained space, we replace it by the closed point which is inside $\mathcal{K}$ otherwise we make the next gradient descent update. The projected gradient descent updates in the t^{th} iteration for a stepsize η are:

$$y_{t+1} = x_t - \eta \nabla f(x_t)$$
$$x_{t+1} = P_{\mathcal{K}}(y_{t+1})$$

Similarly this algorithm extends the Lagevin Monte Carlo to compactly supported distributions via a projection step.Let $\mathcal{K}$ be convex and $0 \in \mathcal{K}$ and $\mathcal{K}$ contains a Euclidean ball of radius r, and it is contained in a Euclidean ball of radius of R. We denote euclidean projection K by $\mathcal{P}_K$. We assume that potential U be an L-Lipschitz and β smooth convex function. The algorithm samples in polynomial time which efficient more efficient than others. The performance is simillar as Hit and Run algorithm .In comparison to older methods such as the lattice walk, Projected LMC achieves significantly better mixing time bounds. The drawback of the algorithm is that we have to choose Δt and N properly, moreover, potential needs to satisfy the condition of β smoothness and Lipschitzness for convergence. We have following Projected Langevin Monte Carlo algorithm:

Algorithm 11 Projected Langevin Monte Carlo Algorithm

Set $X_0 = 0 \in \mathbb{R}^d$ and fix a time discretization step Δt.
For n in 1 to N do :

1. Draw $\nu \sim N(0, \mathbb{I}_d)$.
2. $X_{n+1} = \mathcal{P}_K(X_n - \nabla U(X_n)\frac{\Delta t}{2} - 2\sqrt{\Delta t}v)$

3.1.5 Sampling from a log-concave distribution with compact support with proximal Langevin Monte Carlo

The MYULA algorithm Brosse et al. (2017) Langevin Dynamics parallel to proximal gradient descent, for constrained space. Proximal gradient descent is the optimization algorithm for minimizing unconstrained f which can be decomposed into convex function g and non smooth convex function H. The algorithm is a generalization of projected gradient descent. If you take a point ν outside the domain of f the prox operator will return a point which is in the domain close to the original point (where the function f is small). If you are inside the domain then prox operator will move you towards the minimum of f, and if you are at the minimizer of f you will stay there. The proximal gradient descent updates for t^{th} iteration and non-negative step size η_t are:

$$\begin{aligned} 1: \quad & y^{t+1} = x^t - \eta_t \nabla g\left(x^t\right) \\ 2: \quad & x^{t+1} = \arg\min_{x \in \mathbb{R}^d} \left[h(x) + \frac{1}{2\eta_t} \left\| x - y^{t+1} \right\|_2^2 \right] = \operatorname{prox}_{h\eta_t}(y^{t+1}) \end{aligned}$$

In this algorithm we first change or regualrize our target distribution on the compact support into a function into the unrestricted space by Moreau-Yosida envelope. It then follows Euler Maruyama discretization step in ULA followed by taking proximal of the generated update. We now describe these steps mathematically.
Let $\mathcal{K}$ is a d dimensional compact convex set with non-empty interior. Define $l_{\mathrm{K}} : \mathbb{R}^d \to \{0, +\infty\}$ as the (convex) indicator function of $\mathcal{K}$, defined for $x \in \mathbb{R}^d$ by,

$$l_{\mathrm{K}}(x) = \begin{cases} +\infty & \text{if } x \notin \mathcal{K} \\ 0 & \text{if } x \in \mathcal{K}. \end{cases}$$

Let $f : \mathbb{R}^d \to \mathbb{R}$. Consider target probability density π having to a potential $U : \mathbb{R}^d \to (-\infty, +\infty]$ which can be written as

$$U = f + l_{\mathcal{K}}$$

and assume that the function f and the convex set $\mathcal{K}$ satisfy the following assumptions:

1. f is convex.
2. f satisfies the condition that $\forall x, y \in \mathbb{R}^d$ we have, $\|\nabla f(x) - \nabla f(y)\| \leq L_f \|x - y\|$.

That is, it is continuously differentiable on $\mathbb{R}^d$ and gradient Lipschitz with constant L_f.

3. $\mathcal{K}$ contains an euclidean ball centred at 0 with radius r and it is contained in an Euclidean ball centred at 0 with radius R, where $r < R$

We need to regularize U, i.e. find another function that approximates U, in such a way that it satisfies following three conditions:

1. U remains convex even after its decomposition in f and $l_{\mathcal{K}}$.
2. The regularized form of U is continuously differentiable and gradient Lipschitz.
3. The resulting approximation is close to π (e.g. in total variation norm).

Thus, Moreau-Yosida regularized envelope is given as: $U^\lambda = f(x) + {\iota_K}^\lambda$, where $U^\lambda : \mathbb{R}^d \to \mathbb{R}$ and ${l_K}^\lambda$ is given by:

$$\iota_{\mathrm{K}}^{\lambda}(x) = \inf_{y \in \mathbb{R}^d} \left(\iota_{\mathrm{K}}(y) + (2\lambda)^{-1} \|x - y\|^2 \right) = (2\lambda)^{-1} \left\| x - \mathcal{P}_{\mathcal{K}}(x) \right\|^2$$

Here, $\mathcal{P}_{\mathcal{K}}(x)$ is projection of x onto K and λ is regularization parameter which is always non negative. Now, applying ULA on U^λ and taking it's proximal, we get following Moreau-Yosida Unadjusted Langevin Algorithm (MYULA).

Algorithm 12 Moreau-Yosida Unadjusted Langevin Algorithm (MYULA)

Input: step size $\Delta t > 0$, regularization parameter $\lambda > 0$, initial state $X_0 \in \mathbb{R}^d$

1. For $n = 0, 1, 2, \ldots N - 1$
2. Draw $\nu \sim \mathcal{N}(0, I_d)$
3. Compute $\nabla U^\lambda(X_n) = \nabla f(X_n) + \frac{1}{\lambda}(X_n - \mathrm{proj}_K(X_n))$
4. Update X_{n+1}:

$$X_{n+1} = \left(1 - \frac{\Delta t}{\lambda}\right) X_n - \Delta t \nabla U_\lambda(X_n) + \sqrt{2\Delta t}\nu$$

3.1.6 Penalized Overdamped and Underdamped Langevin Monte Carlo Algorithms for Constrained Sampling

In the Penalized Langevin MCMC algorithm on constrained space Gurbuzbalaban et al. (2024) algorithm constrained space problem is converted to unconstrained one by introducing a penalty function in the potential $U(x)$ of the target distribution. We convert the constrained problem to unconstrained one by sampling from $\pi_\delta(x) \propto -(U(x) + S(x)/\delta)$. Thus,the numerical scheme samples from π is approximately same as π_δ if follwoing assumptions hold true:

1. U is β smooth and L Lipschitz
2. $S(x) = 0$ for any $x \in \mathcal{K}$ and $S(x) > 0$ for any $x \notin \mathcal{K}$
3. $\mathcal{K}$ is a compact convex set, contains an open ball centered at 0 with radius $r > 0$, and is contained in a Euclidean ball centered at 0 with radius $R > 0$

For the given algorithm we assume that $S(x) = (\delta_K(x))^2$ and $\nabla S(x) = 2(x - \mathcal{P}_K(X))$, where $\delta_{\mathcal{K}}(x) = \min_{k \in \mathcal{K}} \|x - k\|$. Keeping these assumptions in mind, we have following Penalized Unadjusted Langevin Algorithm.

Algorithm 13 Penalized Langevin Monte Carlo Algorithm

Set $X_0 \in K$ and fix a time discretization step Δt.
For n in 1 to N do :

1. Draw $\nu \sim N(0, \mathbb{I}_d)$.
2. Set $X_{n+1} = X_n - \Delta t \left(\nabla U(X_n) + \frac{\nabla S(X_n)}{\delta} \right) + \sqrt{2\Delta t}\nu$

4 UBA for restricted spaces

4.1 Augmented Drift

In case of distributions having support which is a subset of $\mathbb{R}$, numerical schemes like ULA generate values outside the support.Consider the case of exponential distribution, $f(x) = \lambda e^{-\lambda x}$ in ULA. Here the drift, is $\Delta U = \lambda^{-1} - x$. As drift is pointing inwards, the scheme will naturally generate negative values for x, which are not even in it's support. ULA and UBA both encounter this same issue. ULA generates most of values which are out of the support. The density approximation by ULA is shown in Figure 11. On the other hand we cannot run UBA as drift is not defined on the negative side of number line. Hence, we define drift, outside the support of the distribution in a way that it will "push or pull" the scheme back into it's support whenever the scheme generates values outside the support.

As an example, consider the case of exponential distribution as mentioned above. In UBA, the updates are either +1 or -1 by probability $p(x_n, v_n)$. Let $p(x_n, v_n)$ be a CDF. Then, whenever value of x_n is negative, to push the chain back into support, we need it to make updates in positive direction with probability 1. We make this possible, defining μ as follows:

$$\mu(x) = \begin{cases} +\infty & \text{if } x < 0 \\ -\lambda & \text{if } x \in \geq 0 \end{cases}$$

With this augmented drift, we implement UBA, and call it as Augmented drift Unadjusted Barkers Algorithm (AgUBA). After running the chain, we discard the values which are out of the support, in this case, negative values. Figure 11 and Figure 12 compare the equilibrium distributions of AgUBA with truth, for $\pi(x) = \lambda e^{-\lambda x}$ when $\lambda = 10$, for different time steps dt and number of steps N. Here "r" denotes the ratio of discarded values to total number of values generated by chain. Both produce similar results.

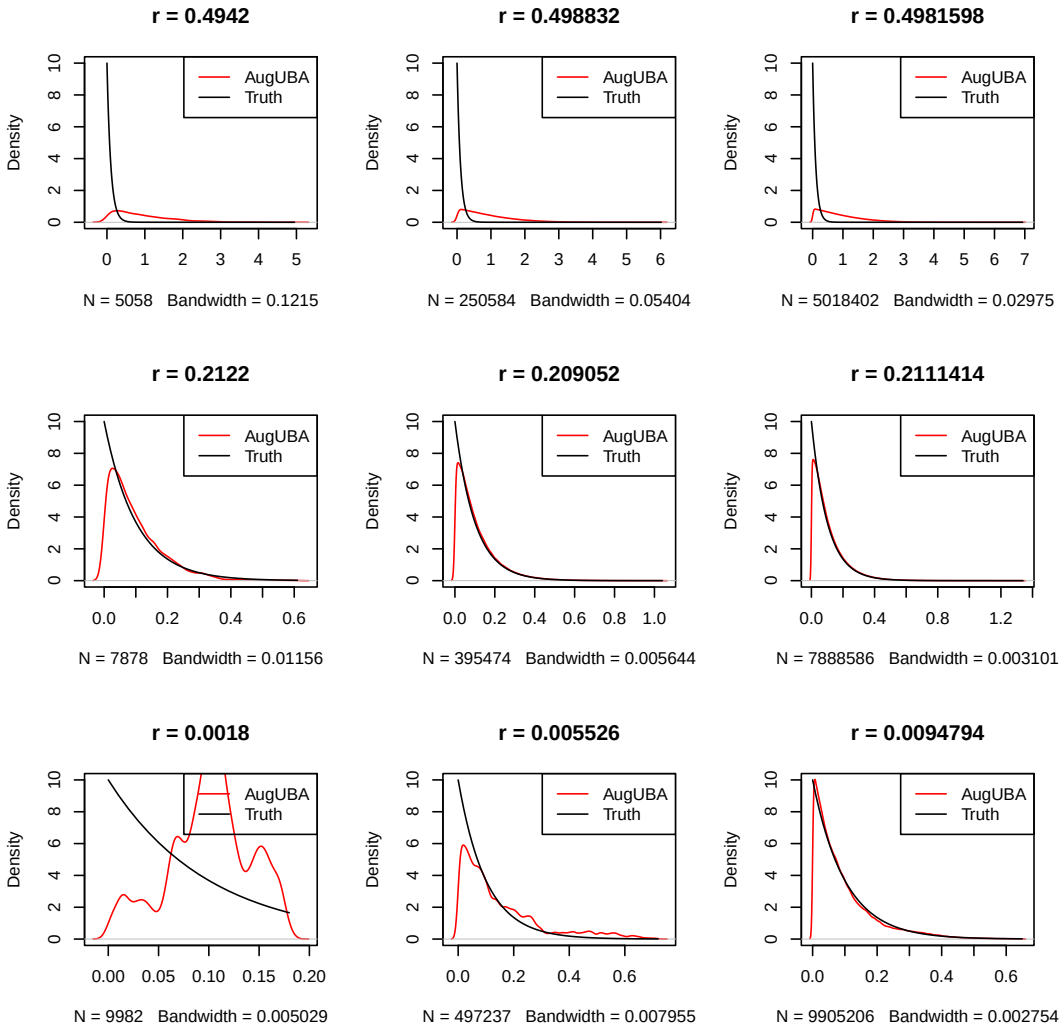


Figure 11: Comparison of equilibrium distribution for logitsic and Logistic CDF as proabbility function

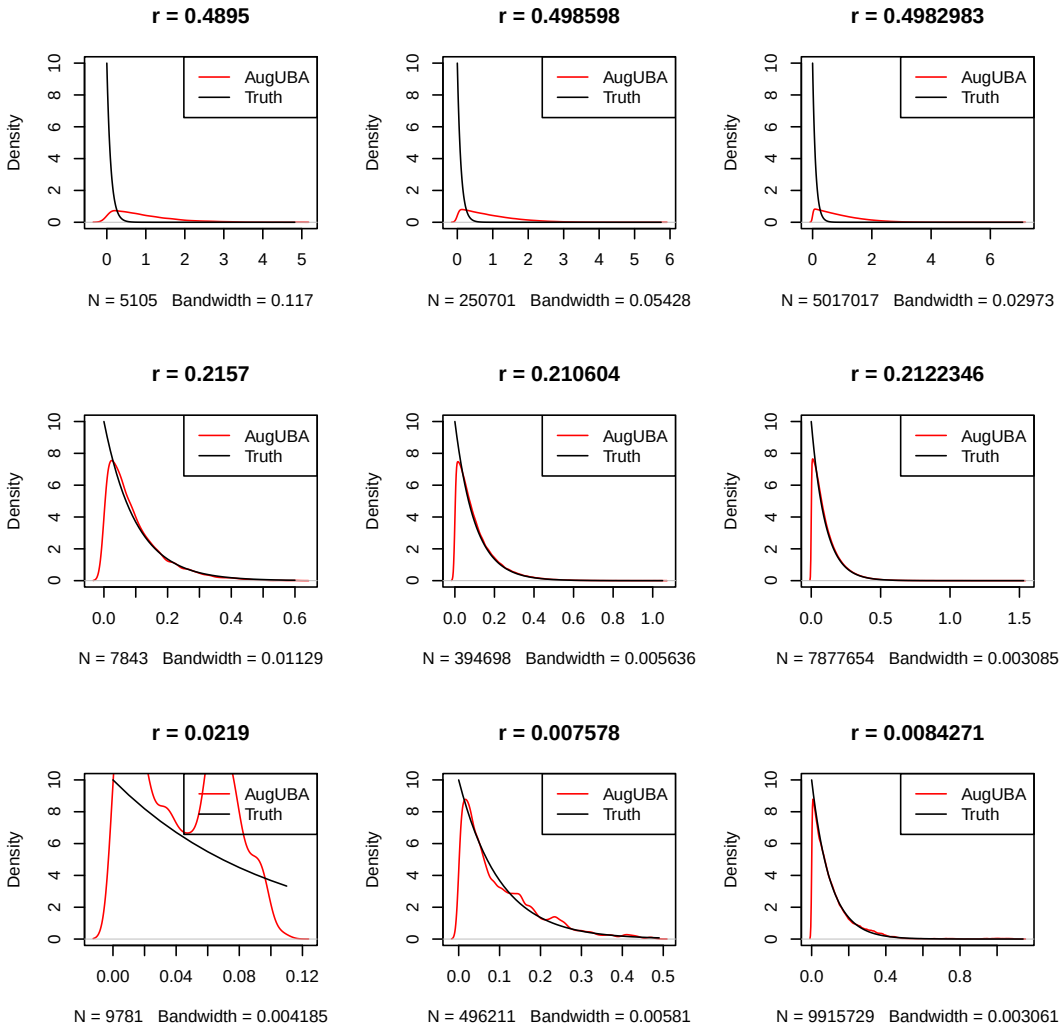


Figure 12: Comparison of equilibrium distribution for Gaussian CDF as proabbility function.

4.2 Poisson Random Effcects model with augmented drift UBA

As we want to check how the augmented drift UBA works in higher dimensions, we implement it on Poisson Random Effects model which has the following form given below. The goal is to estimate the posterior mean of λ.

$$\begin{aligned} y_{ij} \mid \eta_i &\sim \text{Poi}\left(e^{\eta_i}\right), & j &= 1, \ldots, J \\ \eta_i \mid \lambda &\sim N(\lambda, 1), & i &= 1, \ldots, I, \\ \lambda &\sim \text{Exp}\left(\text{rate} = \sigma_l\right). \end{aligned}$$

We calculate the posterior as follows:

$$\begin{aligned} g(y_{ij} \mid \eta_i) &\propto e^{-e^{n_i}(e^{\eta_i})^{y_{ij}}} \\ \Rightarrow g(y \mid \eta_i) &\propto \exp -J \sum_{i=1}^{I} e^{\eta_i} + \sum_{ij} \eta_i y_{ij} \\ h(\eta \mid \lambda) &\propto \exp -\sum_{i} \frac{(\eta_i - \lambda)^2}{2} \\ q(\lambda) &= \sigma_l e^{-\sigma_l \lambda} \end{aligned}$$

We define the state vector $x = (\lambda, \eta_1, \ldots, \eta_I)$. By using Baye's theorem we get:

$$\begin{aligned} f(x|y) &\propto g(y \mid \eta_1, \cdots, \eta_I, \lambda) h(\eta_1, \cdots, \eta_I \mid \lambda) q(\lambda) \\ \Rightarrow f(x \mid y) &\propto \exp - \left(J \sum_{i=1}^{I} e^{\eta_i} - \sum_{i,j} \eta_i y_{ij} + \sum_{i=1}^{I} \frac{(\eta_i - \lambda)^2}{2} + \sigma_l \lambda \right) \\ \Rightarrow f(x \mid y) &\propto e^{-U(x)} \end{aligned}$$

Thus we have a posterior distribution with density $\pi(x) \propto \exp\{-U(x)\}$ and corresponding potential is:

$$U(x) = J \sum_{i} e^{\eta_i} - \sum_{i,j} y_{ij} \eta_i + \frac{1}{2} \sum_{i} (\eta_i - \lambda)^2 + \sigma_l \lambda$$

We calculate the gradient of $U(x)$ as given below. First differentiating with respect to λ

we get:

$$\frac{\partial U}{\partial \lambda} = -\sum_i (\eta_i - \lambda) + \sigma_l$$

Now, differentiating with respect to η_i each we get:

$$\frac{\partial U}{\partial n_i} = Je^{n_i} - \sum_j y_{ij}$$

Thus, we have our gradient:

$$\nabla U = \left(\frac{\partial U}{\partial \lambda}, \frac{\partial U}{\partial \eta_i}\right)$$

If $\lambda \leq 0$ we set the gradient $\frac{\partial U}{\partial \lambda} = -\infty$ or keep it same otherwise. Note that we have drift $\mu(x) = -\nabla U$. Using this augmented drift in 3 we impliment AgUBA, and implement UBA with usual drift. To compare the two schemes we first generate the synthetically.The data y_{ij} for $i = 1, \ldots, I$ and $j = 1, \ldots, J$ were simulated using true parameter values λ^*, with each $\eta_i \sim N(\lambda^*, 1)$. To generate the data we set $I = 50, J = 5$ and $\sigma_l = 10^{-4}$.Then, we consider two ways to initialise the schemes. In "fixed start" we initialise λ at the true value $\lambda^* = 10^{-4}$ and initialize η_i to the same values which are used for generating the data. In "warm start" we initialise λ by taking a random sample from exponential distribution with mean σ_l. To asses the performance of schemes we use the fact that posterior marginal distribution of λ is known to be centred around λ^*, calculate Monte Carlo mean squared error by running a chain 100 times, for each time step. This means that we are our assessment metric is:

$$\frac{\sum_{i=1}^{100}(\hat{\lambda_i} - \lambda^*)^2}{100}$$

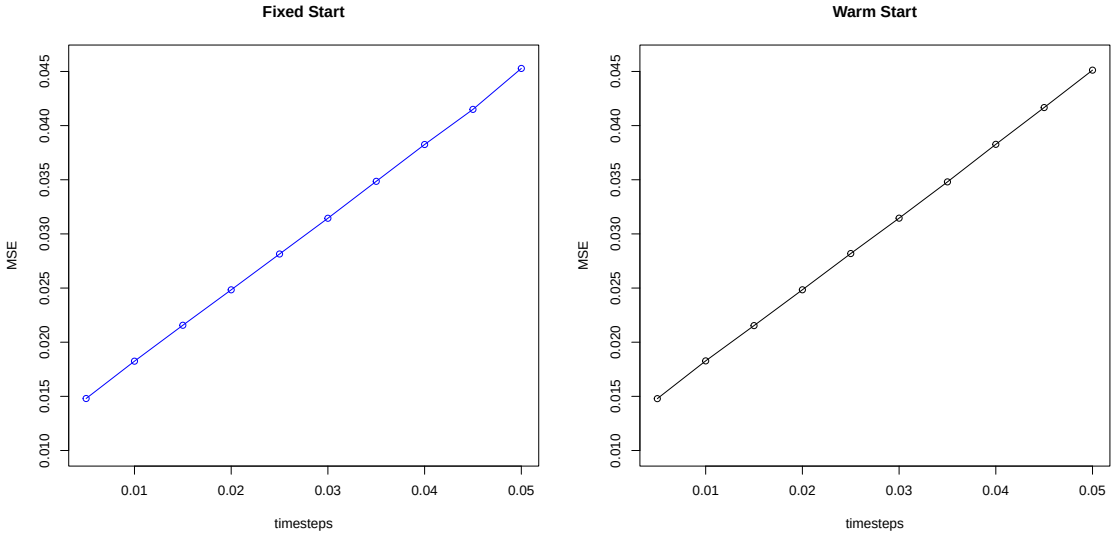
where $\hat{\lambda_i}$ is the estimate for true parameter which is calculated by discarding first 10^4 initial steps and then using next 5×10^4 steps to evaluate the mean. The same procedure is used for assessing AgUBA. As AgUBA discards the negative values, we also calculate the ratio of number of discarded values to total number of steps in the chain. Figure 13 shows comparison of MSE for both schemes on both initialization settings. Figure 14 shows comparison of ratio of negative values produces to total number of steps in the scheme.
The following theoretical questions can be asked for the proposed augmented UBA:

1. What is the rate of convergence of scheme?

2. What is the mixing time of the scheme?

3. Are there any other assumptions required on the potential of target distribution for UBA to converge?

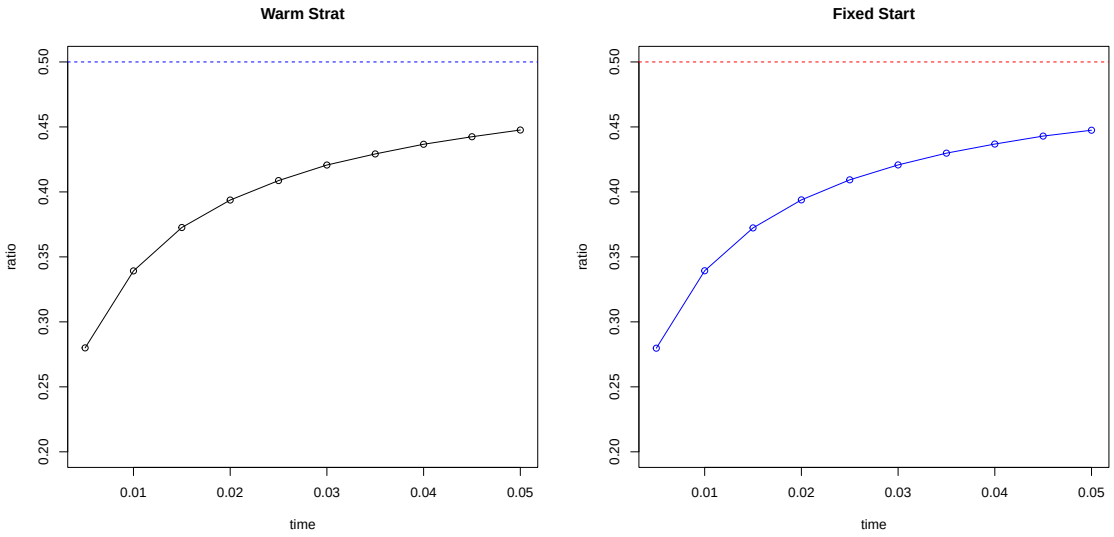
4. Is the equilibrium distribution generated by scheme unbiased and robust?



(a) MSE when initialised at the true value

(b) MSE when initialised by sample

Figure 13: Mean squared error comparisons for the Poisson random effects exponential distribution as prior



(a) warm start

(b) fixed start

Figure 14: For larger time steps, more values are discarded compared to smaller in AgUBA, but they are always less than 0.5

5 Appendix

1. **Wiener Process**: The Wiener process W_t is a continuous-time stochastic process with the following properties:

   (a) Property 1: For each t, the random variable W_t is normally distributed with mean 0 and variance t.

   (b) Property 2: For $t_1 < t_2$, the normal random variable $W_{t_2} - W_{t_1}$ is independent of W_{t_1} and of all W_t for $0 \leq t \leq t_1$

   (c) Property 3: The paths of the Wiener process W_t are continuous.

2. **Globally Lipschitz Functions:** The drift $\mu : \mathbb{R}^d \to \mathbb{R}$ satisfies the global Lipschitz condition if there exists $C > 0$ such that for all $x, y \in \mathbb{R}^d$

$$\|\mu(x) - \mu(y)\| \leq C\|x - y\|$$

   Examples:

   (a) $\mu(x) = x, x \in \mathbb{R}, C = 1$

(b) $\mu(x) = \sqrt{(x^2+5)}, x \in \mathbb{R}, C = 1$

3. **One-Sided Lipschitz Condition for drifts**: The drift satisfies a one-sided Lipschitz condition is there exists $C > 0$ such that

$$(\mu(y) - \mu(x)) \cdot (y - x) \leq C^2 \left(1 + \|y - x\|^2\right)$$

for all $x, y \in \mathbb{R}^d$.
Example: $\mu(x) = e^{-x}, x \in \mathbb{R}, C = 0$

4. β **smooth potential U:**
For a differentiable $U : \mathcal{K} \to \mathbb{R}$, we have, $\forall x, y \in K, |\nabla U(x) - \nabla U(y)| \leq \beta |x - y|$ and $|\nabla U(x)| \leq L$.

References

Azzalini, A. and Regoli, G. (2012). Some properties of skew-symmetric distributions. *Annals of the Institute of Statistical Mathematics*, 64:857–879.

Brosse, N., Durmus, A., Moulines, É., and Pereyra, M. (2017). Sampling from a log-concave distribution with compact support with proximal langevin monte carlo. In *Conference on learning theory*, pages 319–342. PMLR.

Bubeck, S., Eldan, R., and Lehec, J. (2018). Sampling from a log-concave distribution with projected langevin monte carlo. *Discrete & Computational Geometry*, 59:757–783.

Gurbuzbalaban, M., Hu, Y., and Zhu, L. (2024). Penalized overdamped and underdamped langevin monte carlo algorithms for constrained sampling. *Journal of Machine Learning Research*, 25(263):1–67.

Hsieh, Y.-P., Kavis, A., Rolland, P., and Cevher, V. (2018). Mirrored langevin dynamics. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

James.R.Lee (2016). Online mirror descent and density approximation. Technical Report 3, University of Washington. Accessed: 2016-03-01.

Livingstone, S., Nüsken, N., Vasdekis, G., and Zhang, R.-Y. (2024). Skew-symmetric schemes for stochastic differential equations with non-lipschitz drift: an unadjusted barker algorithm. *arXiv preprint arXiv:2405.14373*.

MirrorDescent (2020). Mirror descent. Technical Report 17, Carnigie Mellon University. Accessed: 2020-06-01.

Srinivasan, V., Wibisono, A., and Wilson, A. (2024). Fast sampling from constrained spaces using the metropolis-adjusted mirror langevin algorithm. In Agrawal, S. and Roth, A., editors, *Proceedings of Thirty Seventh Conference on Learning Theory*, volume 247 of *Proceedings of Machine Learning Research*, pages 4593–4635. PMLR.