Skoltech

ISP PROJECT REPORT

Adaptive Metropolis-Adjusted Langevin Algorithm

Master's Educational Program: Data Science

Student: Kamil Mardanshin

Project Supervisor: Aleksandr Katrutsa,
Maxim Rakhuba,
Sergey Samsonov

January, 2024

Contents

PROJECT'S PURPOSE

Abstract

The problem of sampling from a given distribution π is well-known and often arises in numerous machine learning problems. When the density of π is known up to the normalization factor a family of Markov chain Monte Carlo (MCMC) algorithms can be applied to sample from π. Wide known instruments to implement MCMC algorithms are the Langevin diffusion and the Metropolis Hastings (MH) algorithm. In our work we implement [7] and compare Metropolis Adjusted Langevin algorithms (MALA) with different step size heuristics and the Fisher Adaptive Metropolis Adjusted Langevin algorithm (FisherMALA) [5] [4] [6].

Preliminaries

MCMC

Consider a probability space $(X, \mathcal{F}, \mathbb{P})$. Given a target distribution π a corresponding Markov chain with the Markov kernel $P : X \times \mathcal{F} \to [0, 1]$ and a limiting distribution π can be constructed, such that for an arbitrary x it is known how to effectively sample from $P(x, \cdot)$. Then one starts sampling procedure from $x_0 \in \mathbb{R}^d$ by drawing sequential samples

$$x_{n+1} \sim P(x_n, \cdot),$$

that converge to samples from π. The *mixing* of chain, i.e. dependence of samples over iterations depends on the kernel P, which gives a rise to a large family of different MCMC algorithms.

MH algorithm

Consider a kernel $Q : X \times \mathcal{F} \to [0, 1]$, such that it is known how to sample from $Q(x, \cdot)$ for any $x \in X$. Suppose Q has the density q, that is $Q(x, A) = \int\limits_A q(x, y)dy$. The Metropolis Hastings algorithm uses the distribution $Q(x, \cdot)$ as a *proposal* distribution in the following accept-reject scheme

Algorithm 1 Metropolis-Hastings step

```
$y_n \sim Q(x_n, \cdot)$
$\alpha(x_n, y_n) \leftarrow \min(1, \frac{\pi(y_n)q(y_n,x_n)}{\pi(x_n)q(x_n,y_n)})$
$x_{n+1} \sim \alpha(x_n, y_n)\delta_{y_n} + (1 - \alpha(x_n, y_n))\delta_{x_n}$
```

Langevin diffusion

One of the ways to form a Markov chain is to consider Euler-Maruyama discretization of a stochastic differntial equation (SDE).

Suppose we are given a distribution π on $\mathbb{R}^d$ with the density proportional to the function $\pi(x)$. Then a random process with the stationary distribution π can be given as the solution of the SDE called *Langevin diffusion*

$$dx_t = \frac{1}{2}A\nabla \log \pi(x_t)dt + \sqrt{A}dB_t, \tag{1}$$

where $A \in \mathbb{R}^{d\times d}$ is a *preconditioner*, which is a symmetric positive definite matrix, and $\sqrt{A}$ is such a matrix that $\sqrt{A}\sqrt{A}^T = A$. B_t is the d-dimensional Wiener process.

MALA

Metropolis adjusted Langevin algorithm takes a proposal distribution based on the Langevin diffusion.
Namely, consider a d-dimensional gaussian distribution

$$Q(x, \cdot) = \mathcal{N}\left(x + \frac{\sigma^2}{2} A\nabla \log \pi(x), \sigma^2 A\right), \tag{2}$$

where σ controls the rate of discretization.

Thus, for MALA without preconditioner we have for a starting point x_0, learning rate $\rho = 0.015,\ \alpha^* = 0.574$. Denote by φ the density of $\mathcal{N}(0, I_d)$.

Algorithm 2 MALA1

```
function MALA1($x_0,\ \rho, \alpha^*, N_{\sigma^2}, N$)
    Tune $\sigma^2$ for $N_{\sigma^2}$ iterations
    Then collect $N$ samples in to set $M$
    for $n = 0, \dots, N_{\sigma^2} + N - 1$ do
        $\eta \sim \mathcal{N}(0, I_d)$
        $y_n \leftarrow x_n + \frac{\sigma^2}{2}\nabla \log \pi(x_n) + \sigma\eta$
        $q(x_n, y_n) \leftarrow \varphi(\eta)$
        $q(y_n, x_n) \leftarrow \varphi(\frac{1}{\sigma}(x_n - y_n - \frac{\sigma^2}{2}\nabla \log \pi(y_n))$
        $\alpha(x_n, y_n) \leftarrow \min(1, \frac{\pi(y_n)q(y_n,x_n)}{\pi(x_n)q(x_n,y_n)})$
        $x_{n+1} \sim \alpha(x_n, y_n)\delta_{y_n} + (1 - \alpha(x_n, y_n))\delta_{x_n}$
        $\sigma^2 \leftarrow \sigma^2(1 + \rho(\alpha(x_n, y_n) - \alpha^*)$
        if $i \geq N_{\sigma^2}$ then
            $M \leftarrow M \cup \{x_{n+1}\}$
        end if
    end for
return $M, \sigma^2$
end function
```

Later we compare MALA1 with MALA2 that differs from MALA1 only in the fact that it does not adapt σ^2.

To avoid computing A^{-1} in computation of $q(y, x)$ the following proposition can be used, which allows $\mathrm{O}(d^2)$ operations for computing accept probability.

Proposition 1. *For the proposal distribution given by MALA*

$$\frac{q(y, x)}{q(x, y)} = \exp(h(x, y) - h(y, x)),$$

where

$$h(z, v) = \frac{1}{2}\left(z - v - \frac{\sigma^2}{4} A\nabla \log \pi(v)\right)^T \log \pi(v).$$

Optimal preconditioner

Preconditioners are used to enhance mixing of chains.
The preconditioner in MALA can be adjusted at each iteration to provide better mixing.
To define the optimality of a preconditioner we consider the *expected squared jumped distance.*

For a discretization step $\delta = \sigma^2$ we obtain for (1)

$$x_{t+\delta} - x_t = \frac{\delta}{2} A \nabla \log \pi(x_t) + \sqrt{A}(B_{t+\delta} - B_t) + \mathrm{o}(\delta).$$

Then the expected squared jumped distance is defined by

$$J(\delta, A) := \mathbb{E}\|x_{t+\delta} - x_t\|_2^2.$$

The covariance of the samples can be estimated by the following proposition.

Proposition 2. *For $x_t \sim \pi$ we have*

$$\mathbb{E}[x_{t+\delta} - x_t] = 0$$
$$\operatorname{cov}(x_{t+\delta}, x_t) = \mathbb{E}[(x_{t+\delta} - x_t)(x_{t+\delta} - x_t)^T] = \frac{\delta^2}{4} A\, \mathbb{E}[\nabla \log \pi(x_t) \nabla \log \pi(x_t)^T] A + \delta A.$$

Moreover,

$$\operatorname{tr}\operatorname{cov}(x_{t+\delta}, x_t)) = \mathbb{E}\operatorname{tr}((x_{t+\delta} - x_t)(x_{t+\delta} - x_t)^T) = J(\delta, A)$$

The minimization problem of $J(\delta, A)$ for a given δ can be solved using the Fisher information.

Proposition 3. *For a given constant $c > 0$ consider symmetric positive definite matrices A with $\operatorname{tr} A = c$. Then*

$$\operatorname{argmin}_A J(\delta, A) = k\mathcal{I}^{-1},$$

where

$$k = \frac{c}{\sum\limits_{i=1}^{d} \mu_i^{-1}}, \; \mathcal{I} = \mathbb{E}_{x\sim\pi}[\nabla \log \pi(x) \nabla \log \pi(x)^T],$$

and $\mu_i > 0$ are eigenvalues of $\mathcal{I}$.

Adaptive MCMC: FisherMALA

The idea is to optimize the proposal given by (2). Therefore an adaptive algorithm need to learn the global variance σ^2 and the preconditioner A.
σ^2 is learned to reach an average acceptance rate around $\alpha^* = 0.574$ as suggested by the results in [1] and A is learned to be proportional to the inverse of the Fisher information as Proposition 3 suggests.

In [6] the global scale $\operatorname{tr} A = c$ is moved from A to σ^2. Specifically, the proposal distribution becomes

$$Q(x, \cdot) = \mathcal{N}\left(x + \frac{\sigma^2}{\frac{2}{d}\operatorname{tr} A} A \nabla \log \pi(x), \frac{\sigma^2}{\frac{1}{d}\operatorname{tr} A} A\right). \tag{3}$$

For simplicity denote

$$s_n := \nabla \log \pi(x_n),$$

then the empirical estimator of the Fisher information is

$$\hat{\mathcal{I}}_n = \frac{1}{n}\sum_{i=1}^{n} s_i s_i^T + \frac{\lambda}{n} I_d,$$

where $\lambda > 0$ is a dumping parameter, which provides a regularization and ensures that the eigenvalues of $\hat{\mathcal{I}}_n$ are positive.

Under certain conditions [2][3] $\lim\limits_{n\to\infty} \hat{\mathcal{I}}_n = \mathcal{I}$, therefore we can set $A_n = \hat{\mathcal{I}}_n^{-1}$.

The following proposition provides the way to compute A_n via matrix R_n such that $R_n R_n^T = A_n$ in $\mathrm{O}(d^2)$.

Proposition 4. *Initializing R_1 as*

$$R_1 = \frac{1}{\lambda}\left(I_d - r_1 \frac{s_1 s_1^T}{\lambda s_1^T s_1}\right),\ r_1 = \frac{1}{1 + \frac{\lambda}{\lambda + s_1^T s_1}}$$

and defining R_n recursively as

$$R_n = R_{n-1} - r_n \frac{(R_{n-1}\phi_n)\phi_n^T}{1 + \phi_n^T \phi_n},\ \phi_n = R_{n-1}^T s_n,\ r_n = \frac{1}{1 + \frac{1}{1+\phi_n^T \phi_n}}$$

it holds that $R_n R_n^T = A_n$.

Algorithm 3 FisherMALA

```
function FISHERMALA($x_0,\ \rho, \alpha^*, N_{\sigma^2}, N_R, N$)
    Tune $\sigma^2$ for $N_{\sigma^2}$ iterations using MALA1
    $x_0, \sigma^2 \leftarrow$ MALA1($x_0,\ \rho, \alpha^*, N_{\sigma^2} - 1, 1$)
    Tune the preconditioner $R$ for $N_R$ iterations.
    $R \leftarrow I_d,\ \sigma_R^2 \leftarrow \sigma^2$
    for $n = 0, \dots, N_R + N - 1$ do
        $\eta \sim \mathcal{N}(0, I_d)$
        $y_n \leftarrow x_n + \frac{\sigma_R^2}{2} R(R^T \nabla \log \pi(x_n)) + \sigma_R R \eta$
        $\alpha(x_n, y_n) \leftarrow \min(1, \exp(\log \pi(y_n) + h(x_n, y_n) - \log \pi(x_n) - h(y_n, x_n)))$
        $s_n^\delta \leftarrow \sqrt{\alpha(x_n, y_n)}(\nabla \log \pi(y_n) - \nabla \log \pi(x_n))$
        Adapt $R$ using $s_n^\delta$ instead of $s_n$ in Proposition 4
        $\sigma^2 \leftarrow \sigma^2(1 + \rho(\alpha(x_n, y_n) - \alpha^*)$
        $\sigma_R \leftarrow \frac{\sigma^2}{\frac{1}{d}\mathrm{tr}(RR^T)}$
        $x_{n+1} \sim \alpha(x_n, y_n)\delta_{y_n} + (1 - \alpha(x_n, y_n))\delta_{x_n}$
        if $i \geq N_{\sigma^2}$ then
            $M \leftarrow M \cup \{x_{n+1}\}$
        end if
    end for
return $M, \sigma^2$
end function
```

PERFORMED TASKS

We compared the sampling algorithms using d-dimensional gaussian mixtures as the target distribution.
Suppose a set of samples X was drawn from a d-gaussian mixture π, and a set of samples Y with equal number of elements was produced by a sampling algorithm with the target distribution π. Then the following metrics can be calculated.

Total variation distance

Since the number of points required to build a dense histogram grows exonentially with the growth of d we calculate the total variation distance across projections on random 1-dimensional subspaces and then average the sum.

Definition 1. For two probability measures P and Q on $\mathbb{R}$ with densities p, q the total variation distance is given by

$$\|P - Q\|_{TV} := \frac{1}{2} \int\limits_{\mathbb{R}} |p(x) - q(x)| dx.$$

To calculate the total variation distance between projections we need the densities of the corresponding distributions. These densities can be approximated using gaussian kernel densities estimators (KDE).

Algorithm 4 Average total variation distance

```
function AvgTV($X, Y, N$)
    $a \leftarrow 0$
    for $n = 1, \dots, N$ do
        $\nu \sim \text{Uniform}(S^{d-1})$
        Calculate projections on the direction $\nu$
        $X_\nu \leftarrow P_\nu(X),\ Y_\nu \leftarrow P_\nu(Y)$
        $p \leftarrow \text{KDE}(X_\nu), q \leftarrow \text{KDE}(Y_\nu)$
        $a \leftarrow a + \|P - Q\|_{TV}$
    end for
return $\frac{a}{N}$
end function
```

Wassertein distance

Definition 2. For two probability measures P and Q on $\mathbb{R}$ the wassertein distance W_1 is given by

$$W_1(P, Q) := \inf_{S \in \Gamma(P,Q)} \mathbb{E}_{(x,y)\sim S} |x - y|,$$

where $\Gamma(P, Q)$ is the set of all couplings of P and Q.

To compute W_1 the optimal transport Python package ot is used.

Effective sample size

Effective sample size measures how many samples from the target distribution give the same variance in calculating the mean of a function as generated samples.
Given a set of samples $X = \{x_1, \dots, x_n\}, x_i \in \mathbb{R}^d$ we compute mean autocovariance for the ith coordinate in the following way.

$$ESS_i := \frac{1}{1 + \sum\limits_{k=1}^{M} \rho_k^{(i)}},$$

where

$$\rho_k^{(i)} := \frac{\mathrm{cov}[X_{t,i}, X_{t+k,i}]}{Var X_{t,i}},$$

is the autocorrelation at lag k for the ith coordinate.
This, the mean effective sample size is given by

$$ESS := \frac{1}{d} \sum_{i=1}^{d} \frac{1}{1 + \sum\limits_{k=1}^{M} \hat{\rho}_k^{(i)}},$$

where $\hat{\rho}_k^{(i)}$ is the sample covariance estimator.

WORK PLAN

1. Study the algorithms in the main article [6].
2. Examine the Ex2MCMC repository [4].
3. Implement MALA with step heuristic from the main article
4. Create a benchmarking framework to compare samples with multidimensional gaussian mixtures.
5. Compare MALA from the main article with MALA without step heuristics.
6. Implement FisherMALA
7. Compare FisherMALA and MALA.

EXPECTED AND ACHIEVED RESULTS

ESS, AvgTV, Wassertein metric on fixed number of samples

We calculate metrics in the following framework. For each dimension $d \in \{2, 5, 10, 25, 50\}$ the target distribution is a mixture of 5 d-dimensional gaussian variables with equal weights, random means $\mu \sim U([-c, c]^d)$ and random covariance matrix $\Sigma = CC^T + I,\ C \sim U([0, 1]^{d \times d}$.
We run 25 chains with 1000 burn-in iterations which generate 1000 samples each. Then we calculate averaged metrics.

dim	**ESS**	**AvgTV**	**Wasserstein W_1**
2	0.066	0.075	0.21
5	0.032	0.088	3.15
10	0.019	0.149	21.36
25	0.015	0.201	160.87
50	0.015	0.269	734.40

Table 1: MALA with adaptive step size (MALA1)

dim	**ESS**	**AvgTV**	**Wasserstein W_1**
2	0.0078	0.288	2.16
5	0.0075	0.345	15.21
10	0.0077	0.417	57.78
25	0.0070	0.478	304.68
50	0.0072	0.565	1023.71

Table 2: MALA with constant step size (MALA2)

dim	**ESS**	**AvgTV**	**Wasserstein W_1**
2	0.065	0.071	0.18
5	0.047	0.094	3.29
10	0.040	0.101	14.95
25	0.020	0.133	112.48
50	0.014	0.205	588.64

Table 3: FisherMALA

Thus, the adaptive step size significantly decreases autocovariation of samples and generally speeds up convergence. Moreover, FisherMALA introduces a considerable decrease of Wasserstein distance.

TV threshold based benchmark

We consider a benchmark of sampling algorithms that counts how many iterations are needed for an algorithm to attain a total variation threshold.

The experiment is following. For dimensions $d \in \{2, 5, 10, 20, 50, 100\}$ we generate samples using FisherMALA and MALA with targeted acceptance rate (MALA1). For each dimension d the target distribution is a mixture of 10 d-dimensional gaussian variables with equal weights, random means $\mu \sim U([-c, c]^d)$ and random covariance matrices $\Sigma = CC^T + I,\ C \sim U([0, 1]^{d \times d})$.

a) Burn-in is 10000 iterations. We run 25 chains in parallel and take the average TV.

b) The target threshold is $TV = 0.05$. The limit to the iteration count to generate samples with such TV is 5000.

c) When the iteration step exceeds 2500 we take the last 2500 samples.

d) The TV is computed every 100 iteration. We compute mean TV and 95% confidence interval.

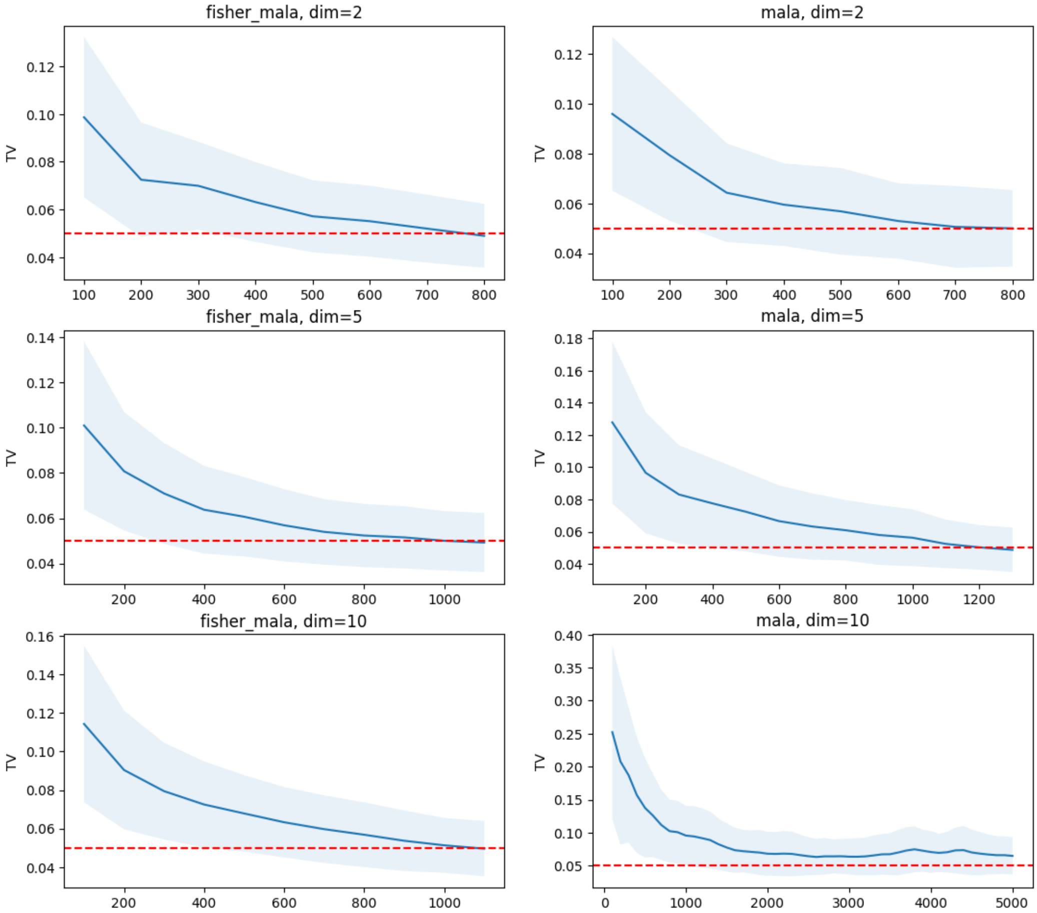

Figure 1: Dimensions $2, 5, 10$

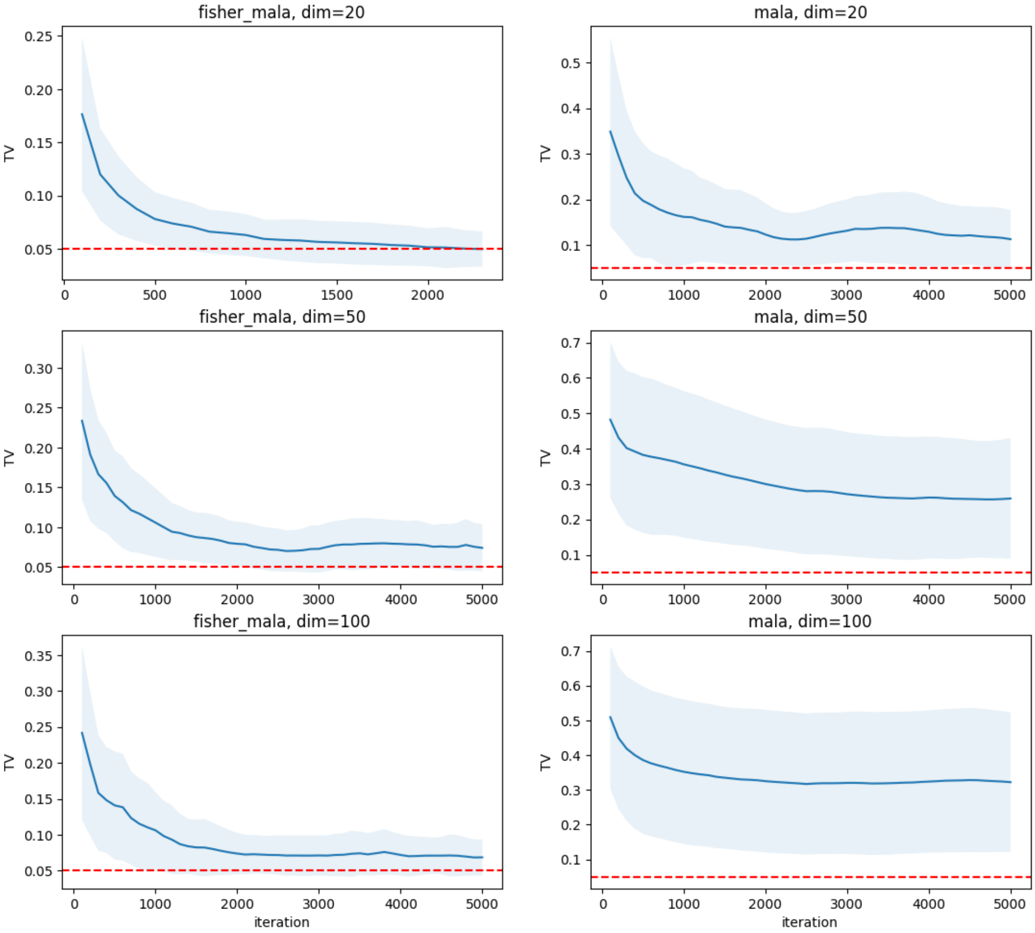

Figure 2: Dimensions $20, 50, 100$

We observe that in higher dimensions FisherMALA outperforms MALA. by Learning of the inverse Fisher information allows FisherMALA to make significantly less iterations to reach TV threshold even in the cases when MALA fails to converge under the time budget.

References

[1] G. Roberts and J. Rosenthal, “Optimal scaling of discrete approximations to langevin diffusions,” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 1, no. 60, pp. 255–268, 1998.

[2] H. Haario, E. Saksman, and J. Tammine, “An adaptive metropolis algorithm,” *Bernoulli*, vol. 2, no. 7, pp. 223–242, 2001.

[3] G. Roberts and J. Rosenthal, “Examples of adaptive mcmc,” *Journal of Computational and Graphical Statistics*, vol. 2, no. 18, pp. 349–367, 2009.

[4] S. Samsonov, E. Lagutin, M. Gabrié, A. Durmus, A. Naumov, and E. Moulines, *Ex2mcmc*, https://github.com/svsamsonov/ex2mcmc_new, 2022.

[5] S. Samsonov, E. Lagutin, M. Gabrié, A. Durmus, A. Naumov, and E. Moulines, “Local-global mcmc kernels: The best of both worlds,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 5178–5193, 2022.

[6] M. K. Titsias, “Optimal preconditioning and fisher adaptive langevin sampling,” 2023. [Online]. Available: https://arxiv.org/pdf/2305.14442.pdf.

[7] K. Mardanshin, *Langevin mcmc*, https://github.com/cepessh/LangevinMCMC, 2024.