Large scale Bayesian inference

Inass Sekkat

École des Ponts
ParisTech

THÈSE DE DOCTORAT
de l'École des Ponts ParisTech

Large Scale Bayesian Inference

École doctorale N532, Mathématiques et STIC (MSTIC)

Spécialité : Mathématiques

Thèse préparée au sein du :
CENTRE D'ENSEIGNEMENT ET DE RECHERCHE EN MATHEMATIQUES ET CALCUL SCIENTIFIQUE

Thèse soutenue le 21 Septembre 2022, par
Inass SEKKAT

Composition du jury:

M. Fabien PANLOUP LAREMA, Université d'Angers	*Président*
M. Arnak DALALYAN ENSAE / CREST	*Rapporteur*
M. Konstantinos ZYGALAKIS School of Mathematics, University of Edinburgh	*Rapporteur*
Mme Marylou GABRIÉ École Polytechnique (CMAP)	*Examinatrice*
Mme Sophie LARUELLE University Paris-Est Créteil	*Examinatrice*
M. Gabriel STOLTZ École nationale des ponts et chaussées	*Directeur de thèse*

REMERCIEMENTS

J'aimerais avant tout adresser ma reconnaissance et mes remerciements à mon directeur de thèse Gabriel Stoltz. Je lui suis redevable pour sa présence, sa disponibilité et ses encouragements, sans lesquels ce travail n'aurait pas été possible. Je le remercie de m'avoir donné cette opportunité, de m'avoir accompagné et guidé durant cette thèse. J'ai beaucoup apprécié et énormément appris durant ces 4 années.

Je tiens aussi à exprimer ma reconnaissance à Arnak Dalalyan et à Konstantinos Zygalakis pour avoir accepté d'être les rapporteurs de ma thèse et pour l'intérêt présenté aux questions abordées dans ce manuscrit. Je voudrais également remercier les membres du jury qui ont accepté de participer à ma soutenance : Marylou Gabrié, Sophie Laruelle et Fabien Panloup.

Je tiens à exprimer mes sincères remerciements à l'Université Mohammed VI Polytechnique pour avoir financé ma thèse. Je remercie particulièrement Salma Lahbabi et Abdellah Chkifa pour toutes les discussions intéressantes que nous avons eues.

Je voudrais aussi remercier Rajaa Aboulaich qui m'a initié à la recherche et sans qui cette thèse n'aurait pas vu le jour. Je tiens tout particulièrement à remercier Tony Lelièvre et Geneviève Robin avec qui j'ai eu l'occasion de collaborer pendant le CEMRACS, je les remercie pour m'avoir accompagné et conseillé pour ce projet. Un grand merci évidement à Ben Leimkuhler et Tiffany Vlaar qui m'ont accueillie pendant deux mois à l'université d'Edimbourg. Grâce à eux, mon séjour en Ecosse fut une très belle expérience scientifique et humaine.

Mes remerciements vont de soi pour l'ensemble du CERMICS et toutes les personnes que j'ai pu y côtoyer tout au long de ces années. Je dois particulièrement remercier Isabelle Simunic et Stéphanie Bonnel pour leur accompagnement et leurs aides pendant ces années. Ces années au CERMICS ont été une expérience inoubliable, et c'est en grande partie grâce à tout ces doctorants et docteurs qui sont devenus de vrais amis. J'aimerais remercier infiniment les anciens : Grégoire, Lingling, Rafael, Pierre-Loïk, Oumaima, William, Adel, Benoît, Mouad, Sami, les contemporains : Dylan, Gaspard, Michel, Rémi, Raed, Cyrille, Sébastien, Thomas, Jean, Laurent, Rutger, Roberta, Etienne, Mohamed, et les nouveaux : Alfred, Coco, Éloïse, Emanuele, Régis, Renato, Zoe, Noé, Alberic...

Je tiens également à remercier tout mes amis qui ont toujours été là pour moi, d'abord Salma Salimixx, Mahmoud (et ses bolognaises du soir), Ayoub^2, Zineb (Khojii), YounOus, Oumama (H), Houssam, Khansae et Hicham. Mes amis de prépa : Dina, Achraf, et Bounouar. Je remercie de même mes amis de toujours : Zineb, Salima et Achraf, notamment pour nos brunchs du Dimanche qui m'ont tant apporté. Et bien-sur merci à Zakaria d'avoir égayé mes deux dernières années de thèse.

Pour finir, je voudrais dédier cette thèse à ma chère famille que je ne remercierai jamais assez. D'abord à mes parents maman et papa à qui je dois tout et qui m'ont toujours soutenue, et bien-sur à Sophia et Ghali, mes piliers qui ont toujours cru en moi et qui ont su me redonner

le sourire dans les moments difficiles, les mots ne serons jamais assez forts pour exprimer ma reconnaissance, enfin à ma grand mère Aicha qui ne m'oublie jamais dans ses prières, et à toutes mes tantes et cousins au Maroc et ailleurs.

Résumé

Cette thèse s'intéresse à divers problèmes d'échantillonnage. La première partie de ce travail est consacrée au problème de l'inférence Bayésienne. Dans ce contexte, les méthodes de Monte Carlo habituelles ont un coût de calcul qui croît linéairement avec le nombre de points de données. L'échantillonnage par des discrétisations de dynamiques de type Langevin, avec une estimation de la force par minibatching pour limiter le coût de calcul, permet une simulation plus efficace, mais induit un biais sur la mesure de probabilité effectivement échantillonnée. La dynamique de Langevin adaptative corrige automatiquement le bruit supplémentaire résultant du minibatching. Nous étudions la pertinence pratique des hypothèses qui sous-tendent la dynamique de Langevin adaptative (notamment covariance de l'estimateur de la force constante), qui ne sont pas satisfaites pour certains modèles typiques d'inférence Bayésienne, et nous quantifions le biais induit par le minibatching dans ce cas. Nous montrons également comment étendre la dynamique de Langevin adaptative afin de réduire systématiquement le biais sur la distribution postérieure en considérant une friction dynamique dépendant de la valeur courante du paramètre à échantillonner. La deuxième partie de ce travail étudie l'erreur de minibatching lors de l'échantillonnage de la distribution a posteriori des paramètres d'un réseau de neurones Bayésien. Nous étudions numériquement la matrice de covariance de l'estimateur stochastique de la force, qui s'avère être de rang faible, suggérant qu'elle peut être efficacement approchée. Ceci ouvre la voie au développement d'algorithmes à coût de calcul raisonnable basés sur la dynamique de Langevin adaptative pour réduire le biais. La dernière partie de cette thèse considère l'échantillonnage des chemins de transition reliant un état métastable à un autre, difficiles à échantillonner par des méthodes numériques directes. Nous explorons certaines techniques d'apprentissage automatique pour générer plus efficacement lesdits chemins de transition.

Abstract

This thesis is concerned with various sampling problems. The first part of this work is dedicated to Bayesian inference problems where usual Monte Carlo methods scale linearly with the number of data points. Resorting to minibatching in conjunction with discretizations of Langevin-like dynamics to circumvent this issue induces bias on the invariant probability measure. Using Adaptive Langevin dynamics automatically corrects for the extra noise arising from minibatching. We investigate the practical relevance of the assumptions underpinning Adaptive Langevin dynamics (in particular, constant covariance for the estimation of the gradient), which are not satisfied in typical models of Bayesian inference, and quantify the bias induced by minibatching in this case. We also show how to extend Adaptive Langevin dynamics in order to systematically reduce the bias on the posterior distribution by considering a dynamical friction depending on the current value of the parameter to sample. The second part of this work studies the error arising from minibatching error when sampling the posterior distribution of parameters of Bayesian neural networks. We numerically investigate the covariance matrix of the stochastic estimator of the force, which turns out to be of low rank, suggesting that it can efficiently be approximated. This opens the way to the development of scalable algorithms based on the adaptive Langevin dynamics to reduce the bias. The final part of this thesis is concerned with sampling transition paths linking one metastable state to another, which can be difficult by direct numerical methods. We explore some machine learning techniques to more efficiently generate transition paths.

Preamble

One of the main challenges in problems of machine learning and computational statistics is the estimation of parameters of some model and the description of their behavior. In the Bayesian framework, the parameters are considered as random variables distributed according to a probability measure. The goal is to sample these parameters with respect to their posterior probability measure, obtained as the product of a prior distribution and the likelihood of the data points at hand under the model associated with the value of the parameters. Typical quantities of interest are expectations with respect to the posterior distribution of the parameters. Methods based on Markov Chain Monte Carlo are among the most popular techniques to sample from high-dimensional and complex probability measures, such as the a posteriori measures on parameters encountered in the Bayesian context. Two major classes can be distinguished: (i) Metropolis based algorithms; (ii) algorithm based on the discretization of stochastic differential equations. One can also mix both, as for example in the Metropolis-adjusted Langevin algorithm.

We tackle in this thesis three problems: (i) Bayesian inference problems; (ii) training Bayesian neural networks; (iii) sampling transition paths. We give below a more precise overview of the organization of this manuscript.

Chapter 1. This chapter is intended to provide a general introduction to the manuscript. First, in Section 1.1, we set the motivation for the three main considered problems. In Section 1.2, we give a review of some of the most popular Markov chain Monte Carlo methods to sample from a given probability measure. We recall in Section 1.3 methods to sample from probability distributions in a large data context, namely the minibatching procedure. The main contributions of this thesis are summarized in Section 1.4.

Chapter 2. The material for this chapter has been preprinted in [137] (submitted to *The Journal of Machine Learning Research*, currently under review). In this chapter, we tackle the problem of sampling a probability measure in a Bayesian inference problem. More precisely, we consider the case when minibatching induces bias on the invariant probability measure. We first provide refined weak error error estimates on the invariant probability measure sampled by discretizations of Langevin-like dynamics when using minibatching. The second part of Chapter 2 is dedicated to a careful (numerical) analysis of Adaptive Langevin dynamics. We provide quantitative estimates of the error arising from minibatching and discretization on the invariant measure actually sampled by the numerical algorithm, even in the case when the usual assumptions underpinning Adaptive Langevin are not satisfied. The last contribution of this chapter is the introduction of an extended version of Adaptive Langevin that allows to systemically reduce the bias.

Chapter 3. The work reported in this chapter started during a two month research visit to Edinburgh, where I worked with Ben Leimkuhler and Tiffany Vlaar (School of Mathematics, University of Edinburgh). We present in this chapter the preliminary results of this work on which we are currently still working. The aim is to analyze the minibatching error when sampling from the posterior distribution of the parameters of a Bayesian neural network. To this end, we numerically analyze the structure of the covariance matrix of the stochastic estimator of the gradient for some toy models. The average isotropic shape of this matrix, and the fact that it is of low rank suggest that it can efficiently be approximated in an

unexpensive way. This opens the way to scalable Adaptive Langevin-like algorithms to reduce the minibatching bias.

Chapter 4. The material for this chapter has been preprinted in [92] (submitted to *ESAIM Proceedings* in the special issue gathering contributions from the research projects initiated during the 6 week long summer research school CEMRACS 2021). The aim of this work is to use recent methods in machine learning, in particular generative methods such as variational auto-encoders and reinforcement learning techniques, to sample rare events in molecular dynamics. These rare events are typically transition paths linking one metastable state to another, which can be difficult to simulate by direct numerical methods.

List of publications

Here is a list of works written during this thesis:

- [137] (with Gabriel Stoltz) *Removing the mini-batching error in Bayesian inference using Adaptive Langevin dynamics*, **arXiv:2105.10347**
- [92] (with Tony Lelièvre, Geneviève Robin, Gabriel Stoltz and Gabriel Victorino Cardoso) *Generative methods for sampling transition paths in molecular dynamics*, **arXiv:2205.02818**

Oral presentations in conferences and seminars

Below is a list of conferences where I presented the above mentioned works:

- Bézout-Facebook scientific workshop, Université Paris–Est, November 4th, 2019.
- International Conference in Monte Carlo & Quasi-Monte Carlo Methods in Scientific Computing MCQMC 2020, online, August 10-14, 2020.
- Bernoulli-IMS One World Symposium, online, August 2020.
- PhD students' seminar, LJLL, Sorbonne université, April 2021.
- SIAM Materials Science 2021, online, May 2021.
- ML/MD seminar, Edinburgh University, October 2021.
- Journées MAS, Rouen, August 2022.

Poster presentations

Below is a list of posters I presented :

- Doctoral school UM6P, Morocco, November 2019.
- Bayes Comp, University of Florida, January 2020.
- Bernoulli-IMS One World Symposium, online, August 2020.

CONTENTS

CHAPTER 1

INTRODUCTION

Contents

This chapter introduces the three problems considered in this thesis and the mathematical tools used to tackle them, namely Markov Chain Monte Carlo (MCMC) methods based on the discretization of stochastic differential equations. The introduction of this thesis is organized as follows. We first give a brief overview of the various sampling problems we consider in Section 1.1. We then present in Section 1.2 some classical methods to sample from a given probability measure, alongside with some elements on the mathematical framework to characterize the errors on the quantities of interest. We next recall in Section 1.3 methods to sample from probability distributions in a large data context. Finally, we summarize the main contributions of this thesis in Section 1.4.

1.1 Motivating sampling problems

In this section, we briefly present the main settings of the various sampling problems we address in this manuscript: Bayesian inference in Section 1.1.1, Bayesian Neural Networks in Section 1.1.2 and sampling transition paths in Section 1.1.3.

1.1.1 General presentation of Bayesian inference problems

We first present the context of Bayesian inference [127, 84]. Let us consider a set of N_{data} independent and identically distributed (i.i.d.) data points denoted by $\mathbf{x} = (x_1, ..., x_{N_{\text{data}}}) \in (\mathbb{R}^{d_{\text{data}}})^{N_{\text{data}}}$. We assume that the elements of the data set are distributed according to a probability measure with density $P_{\text{elem}}(\cdot|\theta)$, parametrized by the unknown vector $\theta \in \mathbb{R}^d$. The likelihood of the data set is then given by

$$P_{\text{likelihood}}(\mathbf{x}|\theta) = \prod_{i=1}^{N} P_{\text{elem}}(x_i|\theta).$$

Maximum likelihood. A standard approach in the frequentist setting to determine the parameters of the model is to consider the maximum likelihood estimator. It consists in finding the vector of parameters which maximizes the likelihood of the data. The problem reads

$$\theta^* \in \arg\max_{\theta \in \mathbb{R}^d} P_{\text{likelihood}}(\mathbf{x}|\theta) = \arg\max_{\theta \in \mathbb{R}^d} \sum_{i=1}^{N_{\text{data}}} \log P_{\text{elem}}(x_i|\theta).$$

However, the maximum likelihood estimator suffers in some cases from the phenomenon of overfitting. To understand this issue, we refer to the example of coin tossing given in [111, Section 1.1]. The considered data set is composed of results of coin tosses (heads or tails). The likelihood is a Bernoulli distribution of parameter $\theta \in [0, 1]$. If the data set is composed of 3 points, all landing head, the maximum likelihood algorithm will lead to $\theta = 1$, predicting that all the following coin tosses will certainly lead to heads. It is clear that the problem comes from the fact that the dataset is too small. The issue of overfitting is also detailed in Section 1.1.2 in the framework of Bayesian Neural Networks. A common idea to mitigate this issue is to add a penalty term to the objective function, to prevent the parameters from taking large values. We focus instead in this work on the Bayesian approach.

Bayesian approach. The Bayesian approach is based on the idea that the parameters should be considered as random variables, and that by capturing their uncertainty/dispersion, one can avoid over-fitting the data set. A prior distribution on the vector of parameters, denoted by $P_{\text{prior}}(\theta)$, expresses the initial beliefs on the parameters. The proper choice of the prior distribution is still an active research area. To simplify, one possible choice for instance is a Gaussian distribution. Using Bayes' theorem, the posterior distribution $\pi(\theta|\mathbf{x})$ of the vector of parameters θ is given by

$$\pi(\theta|\mathbf{x}) = \frac{P_{\text{prior}}(\theta) P_{\text{likelihood}}(\mathbf{x}|\theta)}{Z}, \qquad Z = \int_{\mathbb{R}^d} P_{\text{prior}}(\theta) P_{\text{likelihood}}(\mathbf{x}|\theta)\, d\theta, \tag{1.1}$$

where Z is the normalization constant. It is in most cases intractable, for example when d is large so that standard quadrature methods cannot be used to compute this quantity. The main goal in the Bayesian approach is to sample from π. The quantities of interest are expectations with respect to the target measure π. Note that to sample from π, one need methods that only require computing π up to a normalization constant since Z is intractable. In Section 1.2, we recall some methods to this end, which provide a basis for the extended version of Adaptive Langevin dynamics [137] introduced in Chapter 2. Sampling from π generally requires the

computation of $\nabla_\theta \log \pi(\cdot|\mathbf{x})$ at each iteration, which is computationally expensive. This issue can be mitigated by resorting minibatching, as discussed in Section 1.3.

Remark 1.1. *Instead of sampling from π, an alternative approach consists in finding the vector of parameters maximizing the posterior distribution:*

$$\theta^* = \arg\max_{\theta \in \mathbb{R}^d} \pi(\theta|\mathbf{x}) = \max_{\theta \in \mathbb{R}^d} \left\{ \sum_{i=1}^{N_{\text{data}}} \log P_{\text{elem}}(x_i|\theta) + \log P_{\text{prior}}(\theta) \right\}.$$

The extra term $\log(P_{\text{prior}}(\theta))$ can be seen in this context as a regularizer for the maximum likelihood. For standard Gaussian priors, it amounts to a ℓ^2 regularizer.

1.1.2 Bayesian neural networks

The second problem we consider is the simulation of Bayesian Neural Networks (BNNs). Let us denote by $\mathbf{y} = (y_1, ..., y_{N_{\text{data}}}) \in \mathcal{Y}^{N_{\text{data}}}$ the labels associated to the data set $\mathbf{x} = (x_1, ..., x_{N_{\text{data}}}) \in \mathcal{X}^{N_{\text{data}}} = (\mathbb{R}^{d_{\text{data}}})^{N_{\text{data}}}$. In general, one can distinguish two types of problems. The first type is classification problems, where each input x_i is classified into one of two or more classes. Binary classification corresponds to the case where $\mathcal{Y}^{N_{\text{data}}} = \{0, 1\}^{N_{\text{data}}}$ (0 corresponds to the first class and 1 to the second one). Multi label classification corresponds to the case where $\mathcal{Y}^{N_{\text{data}}} = \{0, ..., d_{\text{label}}\}^{N_{\text{data}}}$, where d_{label} is the number of classes. The second type of problems is regression problems where the labels $\mathbf{y}$ are continuous outputs. In this case $\mathcal{Y}^{N_{\text{data}}} = (\mathbb{R}^{d_{\text{label}}})^{N_{\text{data}}}$, where d_{label} represents the dimension of the outputs.

We assume that the elements of the data set are independent samples of an intractable distribution denoted by ϕ, namely $(x_i, y_i) \sim \phi$ for any $1 \leqslant i \leqslant N_{\text{data}}$. In machine learning, the goal of supervised learning is, given a data set $(\mathbf{x}, \mathbf{y})$, to construct a predictor which will predict an output y from a new given input x. The idea is to learn a function $f : \mathcal{X} \to \mathcal{Y}$, mapping inputs to outputs, such that the expectation with respect to ϕ of some loss function, denoted by L_f, is minimized. The loss function quantifies the quality of the prediction provided by f. Since the distribution ϕ is intractable, and we only have access to a sample of ϕ, a reasonable approximation to the expectation of the loss function with respect to the distribution ϕ is given by

$$\mathbb{E}_\phi[L_f(X, Y)] \approx \frac{1}{N_{\text{data}}} \sum_{i=1}^{N_{\text{data}}} L_f(x_i, y_i).$$

This approximation is justified by the law of large numbers. For example, in a regression problem, one can consider the least square loss function, given by

$$L_f(x, y) = \frac{1}{2}(f(x) - y)^2, \tag{1.2}$$

for $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. For a binary classification problem, a well adapted loss function is the binary cross entropy loss, which reads

$$L_f(x, y) = y \log(f(x)) + (1 - y) \log(1 - f(x)), \tag{1.3}$$

for $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. One way to understand this loss is by considering that the output of the predictor f represents the probability for an input $x \in \mathcal{X}$ to be in class 1. The binary cross entropy loss is the negative of the log of the likelihood of the data point. For multi label classification, one can use the generalization given by categorical cross entropy. In this case, the predictor's output are probabilities to be in each one of the classes.

Neural networks [81, 82, 16] are a class of functions (predictors) that have shown a great ability to solve machine learning problems. They can be defined as parametric functions $N_\theta : \mathcal{X} \to \mathcal{Y}$ parametrized by $\theta \in \mathbb{R}^d$. In this work we focus on feed–forward, fully connected

neural networks. These can be defined as successive compositions of linear transformation and non–linear activation functions. A neural network is typically composed of K stacked hidden layers. Each layer $\ell \in \{1, ..., K\}$ takes an input $x_{\text{in},\ell} \in \mathbb{R}^{d_\ell}$ and produce an output $x_{\text{out},\ell} \in \mathbb{R}^{d_{\ell+1}}$ such that

$$x_{\text{out},\ell} = \sigma_\ell(b_\ell + W_\ell x_{\text{in},\ell}),$$

where $b_\ell \in \mathbb{R}^{d_{\ell+1}}$ is referred to as bias, $W_\ell \in \mathbb{R}^{d_{\ell+1} \times d_\ell}$ is refered to as weight and $\sigma_\ell : \mathbb{R}^{d_{\ell+1}} \to \mathbb{R}^{d_{\ell+1}}$ is a non linear function called the activation function. For example one can use the rectified linear function ReLU, which, for a vector $z \in \mathbb{R}^{d_{\ell+1}}$, act componentwise as

$$(\sigma_\ell(z))_i = \max(0, z_i), \tag{1.4}$$

for $i \in \{1, ..., d_{\ell+1}\}$. It is clear that for $\ell = 1$, the dimension of the input is $d_1 = d_{\text{data}}$ so that the matrix W^1 should have d_{data} columns. The last layer depend on the problem at hand: (i) for a regression problem, $d_{K+1} = d_{\text{label}}$; (ii) for a binary classification problem $d_{K+1} = 1$; (iii) for a multi label classification problem $d_{K+1} = d_{\text{label}}$. The map N_θ can be recursively defined as

$$\begin{cases} z_1 = x, \\ z_k = \sigma(b_{k-1} + W^{k-1} z_{k-1}), \quad 2 \leqslant k \leqslant K \\ N_\theta(x) = \sigma_K(b_K + W^K z_K). \end{cases} \tag{1.5}$$

The neural network's parameters are composed of all the matrices and the biases, *i.e.* $\theta = (W^\ell, b^\ell)_{1 \leqslant \ell \leqslant K}$. For neural networks, we denote the loss function by $L(x, y, \theta)$ instead of $L_{N_\theta}(x, y)$.

Remark 1.2. *Backpropagation [134] allows to easily compute the gradients of loss functions with respect to the parameters. This makes neural networks a popular supervised learning methods, since optimization algorithms (or sampling ones) requires the computation of the gradient at each iteration.*

Remark 1.3. *If we use a neural network for binary classification, the activation function used in the last layer is often chosen to be the sigmoid function given by*

$$\sigma_K(z) = \frac{\exp(z)}{1 + \exp(z)}. \tag{1.6}$$

This function is useful to encode probabilities as it has values in $[0, 1]$. *For a new input* x, *the predicted label is assigned as*

$$y = \begin{cases} 1 & \text{if} \quad N_\theta(x) \geqslant 1/2, \\ 0 & \text{if} \quad N_\theta(x) < 1/2. \end{cases} \tag{1.7}$$

It can be generalized to a softmax function for multi label classification. Note however that sigmoid functions should not be used as activation function for hidden layers because of the so–called vanishing gradient problem: in the region where the sigmoid is flat, the gradient vanishes, and the learning stops.

Remark 1.4. *We should emphasize that other types of neural networks exist. Among the most important ones, one can cite convolutional neural networks (CNNs) [81, 134] that are more adapted to image processing, and recurrent neural networks [134, 159] that are more adapted to sequential data as encountered for example in speech recognition and natural language processing. We introduce CNNs more precisely and use them in Chapter 4.*

Optimization. In the neural network context, one way to determine a good predictor is to solve the following optimization problem

$$\theta^* = \arg\min_{\theta \in \mathbb{R}^d} \mathbb{E}_\phi[L(X, Y, \theta)].$$

A reasonable approximation to θ^* is to minimize the empirical loss, which reads

$$\theta^* \approx \widehat{\theta}^*_{N_{\text{data}}} = \arg\min_{\theta \in \mathbb{R}^d} \frac{1}{N_{\text{data}}} \sum_{i=1}^{N_{\text{data}}} L(x_i, y_i, \theta). \tag{1.8}$$

Once the parameters of the neural network are defined, for a given input x, the predicted label is given by $y = N_{\widehat{\theta}^*_{N_{\text{data}}}}(x)$, where by an abuse of notation we denote by $\widehat{\theta}^*_{N_{\text{data}}}$ an approximate solution of problem (1.8). Popular algorithms to approximate the solution of (1.8) are stochastic gradient descent (also known as the Robbins-Monro algorithm) [126] and Adam [70]. The data set is generally split in two parts: a first part is used to train the model parameters, while the second one allows to test the results and confirm that the minimum obtained for (1.8) generalizes well to unseen data. Splitting the data set also allows to test different hyperparameters for a model and select the one that generalizes best. To compensate for the limited data size, one can use cross validation techniques, see [16]. One of the main issues when considering predictors based on (1.8) is overfitting. The network can perform very poorly outside the training set $\mathbf{x}$ even when the problem (1.8) is solved perfectly. Several ways to avoid overfitting have been introduced. One can cite for example early stopping [27] where the algorithm is stopped once the loss function starts increasing on the test set. Another popular method is Dropout [141], where units (elements of the matrices and the biases) are randomly dropped during the training. One can also use regularization techniques, by adding a penalty term to the problem (1.8). This term can be the ℓ^1 or ℓ^2 norm of the training parameters. This approach is known as weight decay [78, 62].

Sampling. The Bayesian paradigm has been introduced in the neural network framework as an alternative to optimization to avoid overfitting and asses the uncertainty of the parameters [111]. The idea is to consider the parameters of the neural network as random variables and to infer their posterior distribution $\pi(\theta|\mathbf{x}, \mathbf{y})$ which is given, considering Bayes' rule, by

$$\pi(\theta|\mathbf{x}, \mathbf{y}) \propto P_{\text{prior}}(\theta) \prod_{i=1}^{N} P_{\text{elem}}(y_i, x_i|\theta), \tag{1.9}$$

where $P_{\text{prior}}(\theta)$ is the prior distribution on the vector of parameters and $P_{\text{likelihood}}$ the likelihood of the data, given in term of loss function by

$$P_{\text{elem}}(x, y|\theta) \propto \exp(-L(x, y, \theta)).$$

In this context, for a given input x, the label is predicted for a binary classification problem based on the quantity

$$p(y|x, \mathbf{x}) = \int_\Theta N_\theta(x)\pi(\theta|\mathbf{x})\, d\theta,$$

as

$$y = \begin{cases} 1 & \text{if} \quad p(y|x, \mathbf{x}) \geqslant 1/2, \\ 0 & \text{if} \quad p(y|x, \mathbf{x}) < 1/2. \end{cases} \tag{1.10}$$

For regression problems,

$$y = \int_\Theta N_\theta(x)\pi(\theta|\mathbf{x})\, d\theta.$$

As in Section 1.1.1, the goal here is to sample from the posterior probability density and to compute averages with respect to it. We refer to Section 1.2 for an overview of some sampling methods, and to Chapter 3 for an analysis of minibatching error on the posterior distribution through the numerical analysis of the covariance matrix of the stochastic estimator of the gradient and the use adaptive Langevin dynamics to sample from the posterior probability measure in the BNNs framework.

1.1.3 Sampling rare events: transition paths

The last problem we consider in this manuscript is the sampling of certain trajectories of stochastic dynamics. Consider a stochastic differential equation given by

$$\begin{cases} dy_t = b(t, y_t)\, dt + \sigma(t, y_t) dW_t, \\ y_0 \quad \text{given}, \end{cases} \tag{1.11}$$

where $y_t \in \mathbb{R}^d$, $b : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^d$, $\sigma : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^{d \times d'}$ and $W_t \in \mathbb{R}^{d'}$ is a standard d'-dimensional Wiener process. Under mild conditions on b and σ, one can prove the existence of a unique solution of (1.11) (see for instance [69, 123]).

A trajectory is a realization of the stochastic process $(y_t)_{0 \leqslant t \leqslant T}$. For certain choices of b and σ, that will be made explicit first in Section 1.2.2.2 and with more details in Chapter 4, a trajectory such that $y_0 \in A \subset \mathbb{R}^d$ and $y_T \in B \subset \mathbb{R}^d$ can be a rare event, and hence can be difficult to simulate in practice. In our setting, we consider that σ is constant and $b = -\nabla_\theta V$, where V represent some potential energy function. In this case, the system tends to stay trapped in some regions of the phase space, namely in the vicinity of local minima of V. Typically, A and B would be neighborhoods of two distinct local minima of V. The goal is then to efficiently simulate transition paths, defined as trajectories which, from a fixed initial condition y_0 located in the initial potential well A, reach the set B before time $T \geqslant 0$.

This problem has attracted a lot of attention and many methods have been developed to efficiently sample transition paths. One can distinguish two major classes: importance sampling techniques [47, 25, 94] and splitting techniques [34, 8, 28, 29]. We applied generative methods from machine learning literature to sample transition paths. More details are given in Section 4.1 (with a summary of the contributions of this thesis in Section 1.4.3).

1.2 Review of some sampling techniques

In this section, we review some of the most popular Markov chain Monte Carlo methods to simulate from a given probability measure. We only consider probability distributions having a density with respect to the Lebesgue measure. The stochastic differential equations presented in this section were originally introduced in the context of molecular dynamics to sample from the Boltzmann–Gibbs distribution given by

$$\mu(d\theta) = \frac{1}{Z} \exp\left(-\beta V(\theta)\right)\, d\theta, \tag{1.12}$$

where Z is the normalization constant, V the potential energy of the system and β is proportional to the inverse temperature of the system. The normalization constant Z is generally intractable. All the methods presented in this section however have the advantage of only requiring the knowledge of μ up to a normalization constant. We therefore keep the notation μ for the target probability measure in this section for convenience. For a given probability measure π with a density with respect to the Lebesgue measure, the reader can set $\beta = 1$ and $V = -\log \pi$ to recover the desired probability measure.

This section is organized as follows. We start in Section 1.2.1 by recalling some mathematical tools for Markov chains, and then introduce the Metropolis–Hastings algorithm. In

Section 1.2.2, we recall some mathematical tools for stochastic differential equations and introduce important building blocks for the extended Adaptive Langevin dynamics (Chapter 2), namely the overdamped Langevin dynamics (Section 1.2.2.2), the Langevin dynamics (Section 1.2.2.3) and the adaptive Langevin dynamics (Section 1.2.2.4). Finally, in Section 1.2.3, we briefly list other popular sampling methods.

1.2.1 Metropolis–Hastings algorithms

Background material on Markov chains. Let us first recall some important definitions and properties for Markov chains. A Markov chain is a discrete-time sequence of random variables $(\theta^m)_{m\in\mathbb{N}}$ taking values in some space Θ, and which satisfies the Markov property, namely the future state θ^{m+1} is independent of the past and only depend on the present stateθ^m. More precisely a Markov chain should satisfy

$$\mathbb{P}(\theta^{m+1} \in A|\theta^m, ..., \theta^0) = \mathbb{P}(\theta^{m+1} \in A|\theta^m).$$

In this case θ^{m+1} is distributed according to the transition kernel $P(\theta^m, d\theta)$ defined as $P(\theta, A) = \mathbb{P}(\theta^{m+1} \in A|\theta)$ for a measurable set $A \subset \Theta$. The transition kernel of the Markov chain is a probability measure, meaning that it is normalized

$$\int_\Theta P(\theta, d\theta') = 1, \tag{1.13}$$

and positive: for any bounded measurable function ϕ such that $\phi(\theta) \geqslant 0$ for P-almost all θ, it holds

$$\int \phi(\theta')P(\theta, d\theta') \geqslant 0. \tag{1.14}$$

For a probability measure μ, the probability μP is defined as

$$(\mu P)(d\theta') = \int_\Theta P(\theta, d\theta')\mu(d\theta). \tag{1.15}$$

The probability measure μ is said to be invariant for the Markov chain with the transition kernel P if $\mu P = \mu$. In other words, μ is invariant if, for any $\theta^0 \sim \mu$, one has $\theta^m \sim \mu$ for all $m \geqslant 0$. For two transition kernels Q, R, we define $RQ(\theta, A) = \int_\Theta Q(\theta', A)R(\theta, d\theta')$ for a measurable set $A \subset \Theta$. The transition kernel P^n for $n \geqslant 1$ is then defined as $P^n = PP^{n-1} = P^{n-1}P$. To prove the uniqueness of an invariant probability measure, one needs to prove that the Markov chain is (aperiodically) irreducible, meaning that, for any $\theta_0 \in \Theta$ and for all measurable sets $A \subset \Theta$ such that $\mu(A) > 0$, there exist n for which $P^n(\theta_0, A) > 0$. In other worlds, starting from a given state, every state can be reached.

Before presenting the Metropolis–Hasting algorithm, let us recall two important results (see for instance [106]).

Theorem 1.5. *Let μ be a probability measure. If μ satisfies the detailed balance condition, that is*

$$P(\theta, d\theta')\mu(d\theta) = P(\theta', d\theta)\mu(d\theta'), \tag{1.16}$$

then μ is an invariant probability measure for the Markov chain.

The following theorem can be seen as an equivalent of the law of large numbers for Markov chains.

Theorem 1.6 (pathwise ergodicity)**.** *Let μ be a probability density. If μ is invariant and the Markov chain is aperiodic and irreducible, then, for any bounded measurable function ϕ and for μ-almost all initial conditions θ_0,*

$$\lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} \phi(\theta^i) = \mathbb{E}_\mu[\phi(\theta)], \qquad \text{a.s.}$$

Note that the expectations on the right hand side of the previous equality is computed with respect to the probability measure μ.

Metropolis–Hastings algorithm. The Metropolis–Hastings algorithm was introduced in [104, 58] to sample from a given probability measure μ. It compound to a Markov chain $(\theta^n)_{n\in\mathbb{N}}$ reversible with respect to μ. This requires a transition kernel $T(\theta, d\theta)$ with respect to which we can generate random variables. The pseudo-code of the Metropolis–Hastings algorithm is given in Algorithm 1. The main idea of the algorithm is, at each step, to propose moves $\tilde{\theta}_{k+1}$ according to $T(\theta_k, \cdot)$, which are then accepted with probability $r(\tilde{\theta}_{k+1}, \theta_k)$ given by

$$r(\theta_k, \tilde{\theta}_{k+1}) = \min\left(1, \frac{T(\tilde{\theta}_{k+1}, d\theta_k)\mu(d\tilde{\theta}_{k+1})}{T(\theta_k, d\tilde{\theta}_{k+1})\mu(d\theta_k)}\right). \tag{1.17}$$

When a move is not accepted, the previous one is recounted contrarily to the "classical" rejection method.

Algorithm 1 Metropolis-Hastings algorithm

Require: θ_0
  for $k \geqslant 0$ **do**
    Generate $\tilde{\theta}_{k+1}$ from θ_k according to the transition kernel $T(\theta_k, \cdot)$
    Compute $r(\tilde{\theta}_{k+1}, \theta_k)$ as defined in (1.17)
    Draw a random variable U_k following a uniform law on $[0, 1]$
    if $U_k < r(\theta_k, \tilde{\theta}_{k+1})$ **then**
      $\theta_{k+1} = \tilde{\theta}_{k+1}$
    else
      $\theta_{k+1} = \theta_k$
    end if
  end for

The transition kernel of the resulting Markov chain is given by

$$P(\theta, d\theta') = r(\theta, \theta')T(\theta, \theta')\mu(d\theta') + R(\theta)\delta_\theta(d\theta'), \tag{1.18}$$

where

$$R(\theta) = \int_{\mathbb{R}^d} (1 - r(\theta, \theta'))T(\theta, \theta')\mu(d\theta').$$

One can prove (see for instance [93]) that the transition kernel of the Markov chain obtained by the Metropolis–Hastings algorithm satisfies the detailed balance defined in (1.16) for μ. Using Theorem 1.5, one can prove that μ is an invariant probability measure for the transition kernel (1.18). The detailed balance with respect to μ is ensured by the fact that the function $g(u) = \min(1, u)$ satisfies the property $g(u) = ug(\frac{1}{u})$. The choice $g(u) = \min(1, \mathrm{u})$ is optimal in terms of asymptotic variance [119]. Irreducibility, and therefore pathwise ergodicity depends on the choice of the transition kernel T.

The efficiency of the Metropolis–Hastings algorithm is a trade off between large moves that allow to efficiently explore the phase space but are more likely to be rejected, and small moves that are more likely to be accepted but are more correlated to previous ones.

A key element in the Metropolis–Hastings algorithm is the transition kernel T. We briefly present the transition kernel similar to the one used in the original paper [104], which corresponds to what is nowadays called the random walk algorithm. We also present in Section 1.2.2.2 a more recent transition kernel, leading to the so-called Metropolis adjusted Langevin dynamics. Another popular option is the Hamiltonian Monte Carlo algorithm [43] for which the proposals are generated by a numerical integration of the Hamiltonian dynamics.

Random Walk. The random walk Metropolis algorithm uses a Gaussian centered on the current state as a proposal distribution, namely the proposal probability kernel is given by

$$T(\theta, d\theta') = \frac{1}{(\sigma\sqrt{2\pi})^d} \exp\left(-\frac{|\theta' - \theta|^2}{2\sigma^2}\right) d\theta',$$

where $\sigma > 0$ is a given parameter. In this case, the proposed moves are

$$\theta' = \theta + \sigma G,$$

where G is a standard d-dimensional Gaussian random variables. The (one-step) irreducibility of the associated chain is easily proved for connected domains. We refer to [130] and references therein for insights on the optimal choice of σ and optimal acceptance rate, in the simple situation where the target measure is tensorized. In this very specific case, the optimal acceptance/rejection rate should 0.234 and σ scales as the inverse of the dimension.

1.2.2 Stochastic differential equations

In this section, we start by giving some elements on stochastic differential equations (SDE) in Section 1.2.2.1. We then recall various SDEs used to sample from probability measure, namely the overdamped Langevin dynamics (Section 1.2.2.2), the Langevin dynamics (Section 1.2.2.3) and the adaptive Langevin dynamics (Section 1.2.2.4).

1.2.2.1 Some elements on stochastic differential equations

We describe in this section some general features of stochastic differential equations, elements on discretization and weak errors taken from Chapter 2.

We consider a stochastic differential equation of the general form [114, 74, 49]

$$\begin{cases} d\theta_t = b(\theta_t)\, dt + \sigma(\theta_t) dW_t, \\ \theta_0 \quad \text{given}, \end{cases} \tag{1.19}$$

where $\theta_t \in \mathbb{R}^d$, $b : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^d$, $\sigma : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^{d \times d'}$ and $W_t \in \mathbb{R}^{d'}$ is a standard d-dimensional Wiener process. We define the evolution operator of the Markov process (1.19) as follows: for a given test function ϕ,

$$P_t\phi(\theta) = \mathbb{E}[\phi(\theta_t)|\theta_0 = \theta],$$

where the expectation is over all realizations of (1.19) starting from the initial condition θ. We recall that the generator of the semi-group satisfies

$$\lim_{t \to 0} \frac{P_t - \mathrm{Id}}{t} \phi = \mathcal{L}\phi.$$

One can prove that the generator of (1.19) is the differential operator

$$\mathcal{L} = b \cdot \nabla_\theta + \frac{1}{2}\sigma\sigma^T : \nabla^2_\theta = \sum_{i=1}^d b_i \partial_{\theta_i} + \frac{1}{2} \sum_{i,j=1}^d (\sigma\sigma^T)_{i,j} \partial_{\theta_i}\partial_{\theta_j},$$

where ∇_θ stands for the gradient with respect to the variable θ, ∇^2_θ stands for the Hessian with respect to the variable θ and : stands for the Frobenius inner product. An appropriate domain for the unbounded operator $\mathcal{L}$ is for instance $C^2(\mathbb{R}^d)$. In particular, using the Itô formula, one can show that the generator satisfies, for $\phi \in C^2(\mathbb{R}^d)$ with compact support,

$$\frac{d}{dt}\mathbb{E}[\phi(\theta_t)|\theta_0 = \theta] = \mathcal{L}\phi(\theta). \tag{1.20}$$

Invariant probability measure. Let us denote by $\psi(t,\theta)$ the law of the process defined by (1.19) at time t. The Fokker Planck equation (which can be derived from (1.20)) characterizes the evolution of the law of the process:

$$\partial_t \psi = \mathcal{L}^\dagger \psi, \tag{1.21}$$

where $\mathcal{L}^\dagger$ is the adjoint of $\mathcal{L}$ on $L^2(\mathbb{R}^d)$. From this equality follows an important characterization of an invariant probability measure. For a given test function ϕ, μ is invariant for the SDE (1.19) if

$$\int_{\mathbb{R}^d} \mathcal{L}\phi \, d\mu = 0. \tag{1.22}$$

This equality can be obtained by noticing that for μ an invariant probability measure, $\partial_t \mu = 0$ so that $\mathcal{L}^\dagger \mu = 0$ by (1.21). A useful property to characterize an invariant probability measure is reversibility. The SDE (1.19) is said to be reversible with respect to the probability measure μ if the generator is self-adjoint on $L^2(\mu)$, *i.e.* for two test functions ϕ_1, ϕ_2,

$$\int_{\mathbb{R}^d} \phi_1(\mathcal{L}\phi_2) \, d\mu = \int_{\mathbb{R}^d} (\mathcal{L}\phi_1)\phi_2 \, d\mu. \tag{1.23}$$

We can recover (1.22) by setting $\phi_1 = \mathbf{1}$ in (1.23). Under appropriate conditions on b and σ (for example $d = d'$ and σ is of full rank), one can prove pathwise ergodicity [73]: for a given test function ϕ and an initial condition θ_0,

$$\lim_{t\to+\infty} \widehat{\phi}_t = \lim_{t\to+\infty} \frac{1}{t} \int_0^t \phi(\theta_s) ds = \mathbb{E}_\mu[\phi(\theta)] \qquad \text{a.s.} \tag{1.24}$$

A central limit theorem for the time averages $\widehat{\phi}_t$ holds as soon as the Poisson equation $-\mathcal{L}\Phi = \phi - \mathbb{E}_\mu[\phi]$ such that $\int_{\mathbb{R}^d} \Phi \, d\mu$ has a solution in $L^2(\mu)$ for an observable $\phi \in L^2(\mu)$ (see [14]). Using Itô's formula, one can show that the asymptotic variance in the central limit theorem can be written as

$$\sigma_\phi^2 = -2 \int_{\mathbb{R}^d} \Phi\phi \, d\mu$$

Discretization and error estimates. In general, SDEs such as (1.19) cannot be solved exactly and need to be discretized. Denote by $(\theta^m)_{m\geqslant 0}$ a time discretization of the SDE with a fixed time step Δt (so that θ^m is an approximation of $\theta_{m\Delta t}$). We assume that the Markov chain corresponding to the time discretization of the SDE admits a unique invariant probability measure, denoted by $\mu_{\Delta t}$. This is for instance the case for Langevin-type dynamics when the drift of the dynamics is globally Lipschitz or when Lyapunov conditions are satisfied [101]. For a given observable ϕ, the target expectation

$$\mathbb{E}_\mu(\phi) = \int_{\mathbb{R}^d} \phi(\theta)\mu(\theta) \, d\theta$$

is approximated by $\mathbb{E}_{\mu_{\Delta t}}(\phi)$, which is itself typically estimated by the trajectory average

$$\widehat{\phi}_{\Delta t, N_{\text{iter}}} = \frac{1}{N_{\text{iter}}} \sum_{m=1}^{N_{\text{iter}}} \phi(\theta^m).$$

The total error on averages with respect to μ can then be written as:

$$\widehat{\phi}_{\Delta t, N_{\text{iter}}} - \mathbb{E}_\mu(\phi) = (\mathbb{E}_{\mu_{\Delta t}}(\phi) - \mathbb{E}_\mu(\phi)) + \left(\widehat{\phi}_{\Delta t, N_{\text{iter}}} - \mathbb{E}_{\mu_{\Delta t}}(\phi)\right).$$

The first term on the right hand side corresponds to the bias on the invariant probability measure resulting from taking finite step sizes. The second term in the error has two origins:

(i) a bias coming from the initial distribution of θ^0 when this random variable is not distributed according to $\mu_{\Delta t}$; (ii) a statistical error, which is dictated by the central limit theorem for $N_{\rm iter}$ large.

We typically focus in this manuscript on the bias on the invariant probability measure, which can be bounded using the weak order of the scheme, provided some ergodicity conditions are satisfied. Recall that a numerical scheme is of weak order s if for any smooth and compactly supported function ϕ and final time $T > 0$, there exists $C \in \mathbb{R}_+$ such that

$$\forall m \in \{1, \dots, \lceil T/\Delta t \rceil\}, \qquad |\mathbb{E}[\phi(\theta_{m\Delta t})] - \mathbb{E}[\phi(\theta^m)]| \leqslant C\Delta t^s. \tag{1.25}$$

When this condition holds, and under appropriate ergodicity conditions (see [146] for a pioneering work, as well as [145, 101, 2, 87, 22] for subsequent works on Langevin-like dynamics), the following bound is obtained on the bias on the invariant probability measure of the numerical scheme: For any smooth and compactly supported function ϕ, there exists $\Delta t_\star > 0$ and L such that

$$\forall \Delta t \in (0, \Delta t_\star], \qquad |\mathbb{E}_{\mu_{\Delta t}}(\phi) - \mathbb{E}_\mu(\phi)| \leqslant L\Delta t^s. \tag{1.26}$$

In order to write a sufficient local consistency condition to obtain an estimate such as (1.26), we introduce the evolution operator $P_{\Delta t}$ associated with the numerical scheme at hand, defined as follows: For any smooth and compactly supported function ϕ,

$$\left(P_{\Delta t}\phi\right)(\theta) = \mathbb{E}\left[\phi\left(\theta^{m+1}\right) \,\middle|\, \theta^m = \theta\right].$$

Under appropriate technical conditions, including moment conditions on the iterates of the numerical scheme (see [107, Theorem 2.1] for a precise statement), a sufficient but not necessary condition for (1.26) to hold is

$$P_{\Delta t} = \mathrm{e}^{\Delta t\mathcal{L}} + \mathcal{O}(\Delta t^{s+1}), \tag{1.27}$$

where $(\mathrm{e}^{t\mathcal{L}}\phi)(\theta) = \mathbb{E}[\phi(\theta_t) \,|\, \theta_0 = \theta]$ is the evolution operator associated with the underlying SDE. Let us emphasize that (1.27) is not a necessary condition for (1.26) to hold. For instance, some of the schemes suggested in [3] or the so–called Geometric Langevin algorithm [22] are of weak order 1 but satisfy (1.26) with $s \geqslant 2$. Here and in all this manuscript, the above equality has to be understood as follows: For any smooth and compactly supported function ϕ, there exist $\Delta t_\star > 0$ and $K \in \mathbb{R}_+$ such that, for any $\Delta t \in (0, \Delta t_\star]$, there is a function $R_{\phi,\Delta t}$ for which

$$P_{\Delta t}\phi = \mathrm{e}^{\Delta t\mathcal{L}}\phi + \Delta t^{s+1} R_{\phi,\Delta t}, \qquad \sup_{\Delta t \in (0,\Delta t_\star]} \sup_{\theta \in \mathbb{R}^d} |R_{\phi,\Delta t}(\theta)| \leqslant K;$$

see for instance [94, Section 3.3] for a more precise discussion of this point.

Remark 1.7. *Let us emphasize that one can also analyze the strong error which characterizes the error on the trajectory between the discretized and the continuous processes. However, we focus in this thesis on weak error estimates instead of strong errors since we are interested in averages with respect to the invariant measure.*

1.2.2.2 Overdamped Langevin dynamics

The overdamped Langevin dynamics is introduced in the molecular dynamics context to describe the evolution of positions $\theta \in \mathbb{R}^d$ of a system. It is given by

$$d\theta_t = -\nabla V(\theta_t)\, dt + \sqrt{2\beta^{-1}}\, dW_t, \tag{1.28}$$

where W_t is a standard d-dimensional Wiener process. We refer to [93, 86, 118, 94] for more details about the overdamped Langevin dynamics. We recall here some important properties. The generator associated with (1.28) reads

$$\mathcal{L} = \frac{1}{\beta}\Delta - \nabla V \cdot \nabla. \tag{1.29}$$

A simple calculation shows that the Boltzmann–Gibbs distribution $\mu(d\theta) = Z^{-1}\exp(-\beta V(\theta))\,d\theta$ is invariant under the dynamics (1.28). This can be seen as a consequence of the fact that the generator (1.29) is self adjoint on $L^2(\mu)$: for any smooth functions ϕ_1, ϕ_2 with compact support, it holds

$$\int_{\mathbb{R}^d} \phi_1(\mathcal{L}\phi_2)\,\mathrm{e}^{-\beta V} = \frac{1}{\beta}\int_{\mathbb{R}^d} \phi_1 \mathrm{div}\left(\mathrm{e}^{-\beta V}\nabla\phi_2\right) = -\frac{1}{\beta}\int_{\mathbb{R}^d} \nabla\phi_1^T\nabla\phi_2\,\mathrm{e}^{-\beta V}.$$

Using the fact that the generator is elliptic, one can prove the pathwise ergodicity expressed as (1.24). We refer to [10, 94] for mathematical properties of the dynamics, in particular the convergence in $L^2(\mu)$ of the semi-group $\mathrm{e}^{t\mathcal{L}}$. In practice, the overdamped Langevin dynamics is often discretized using the well-known Euler–Maruyama numerical scheme with a fixed time step $\Delta t > 0$, which reads

$$\theta^{m+1} = \theta^m - \nabla V(\theta^m)\Delta t + \sqrt{2\Delta t}G^m, \tag{1.30}$$

where G^m is a vector of i.i.d. standard d-dimensional Gaussian random variables. One can prove that the discretization of the overdamped Langevin dynamics has a weak error of order 1, see [74, 107]. As a consequence, assuming that the dynamics (1.30) admits an invariant probability measure $\mu_{\Delta t}$ (see for instance [102]), and under some technical conditions, there exists, for any test function ϕ, a constant $C_\phi \in \mathbb{R}^+$ such that

$$\left|\int_\Theta \phi(\theta)\mu_{\Delta t}(d\theta) - \int_\Theta \phi(\theta)\mu(d\theta)\right| \leqslant C_\phi \Delta t.$$

Metropolis Adjusted Langevin algorithm. One way to avoid the numerical bias due to the finiteness of the time step is to use a Metropolis–Hastings acceptance/rejection at each time-step [23]. The resulting algorithm is known as Metropolis adjusted Langevin algorithm (MALA) [131]. In this case, the Markov chain admits μ as an invariant probability measure whatever the choice of time step $\Delta t > 0$. This can be seen as a Metropolis–Hastings algorithm for which the transition kernel T is given by

$$T(\theta,\, d\theta') = \left(\frac{\beta}{4\pi\Delta t}\right)^{d/2} \exp\left(-\beta\frac{\|\theta' - \theta + \Delta t\nabla_\theta V(\theta)\|^2}{4\Delta t}\right) d\theta'.$$

In this case, the acceptance/rejection ratio is given by

$$r(\theta,\theta') = \exp(\beta(V(\theta) - V(\theta'))).$$

It is clear that if the proposed move θ' is such that $V(\theta') \leqslant V(\theta)$, the movement is always accepted. Other ways transition are less likely. In [130] (see also references therein), provided that we start under the stationary distribution, and that it is tensorized and satisfies some technical conditions, the authors give insights on the optimal choice of Δt and rules about the acceptance rate. Concretely, one should aim for a rate 0.574 and Δt scales as $d^{-1/3}$, leading to a faster convergence to stationarity than the random walk because the optimal time step can be chosen much larger than for the random walk Metropolis algorithm as the dimension increases.

1.2.2.3 Langevin dynamics

It has been observed in practice that a better sampling of the Boltzmann–Gibbs probability measure μ is provided by the Langevin dynamics [26]. This dynamics introduces some extra inertia in the evolution. It is formulated for an extended configuration space with a momentum vector p conjugated to θ. Let

$$\tau(dq\,dp) = Z^{-1}\exp(-\beta H(\theta,p))d\theta\,dp = \mu(d\theta)\kappa(dp),$$

where

$$\kappa(dp) = \left(\frac{\beta}{2\pi}\right)^{d/2} \exp\left(-\beta \frac{p^T M^{-1} p}{2}\right) dp.$$

A motivation to sample from τ is that its marginal in the variable θ is μ. The marginal in the variable p is a standard normal distribution from which one can easily sample. Langevin dynamics can be seen as a perturbation of Hamiltonian dynamics. The latter describe the time evolution of mechanical systems as

$$\begin{cases} \dfrac{d\theta}{dt} = \nabla_p H(\theta(t), p(t)), \\ \dfrac{dp}{dt} = -\nabla_\theta H(\theta(t), p(t)), \end{cases} \tag{1.31}$$

where H is the Hamiltonian function which corresponds to the total energy of the system. It is typically the sum of the standard kinetic energy and the potential energy V:

$$H(\theta, p) = \frac{p^T M^{-1} p}{2} + V(\theta),$$

where $M \in \mathbb{R}^{d\times d}$ is called the mass matrix. We say in this case that the Hamiltonian is separable. One of the properties of (1.31) is that the Hamiltonian is preserved, in other words $H(\theta_0, p_0) = H(\theta_t, p_t)$ for all $t \geqslant 0$. We refer to for example to [89, 55] for more details on properties of Hamiltonian dynamics and its discretization. In order to sample the probability measure τ with density proportional to $\mathrm{e}^{-\beta H(q,p)}$, which admits μ as marginal distribution in the variable θ, we recall the Langevin dynamics, which is a perturbation of the Hamiltonian dynamics where some fluctuation/dissipation mechanism is added to the evolution of the momenta. It is given by

$$\begin{cases} d\theta_t = p_t\, dt, \\ dp_t = -\nabla V(\theta_t)\, dt - \Gamma M^{-1} p_t\, dt + \sqrt{\dfrac{2}{\beta}} \Gamma^{1/2}\, dW_t, \end{cases} \tag{1.32}$$

where $\Gamma \in \mathbb{R}^{d\times d}$ is a positive definite symmetric matrix which represents the friction parameter. We refer to [118, 86, 94] for more details and a complete analysis of the dynamics. We recall here some important properties of the dynamics. From now on, we set $M = \mathrm{I}_d$ to simplify the presentation. The generator of the dynamics (1.32) can be written as

$$\mathcal{L}_{\mathrm{lan}} = \mathcal{L}_{\mathrm{ham}} + \gamma \mathcal{L}_{\mathrm{FD}}\ , \tag{1.33}$$

where

$$\mathcal{L}_{\mathrm{ham}} = -\nabla V^T \nabla_p + p^T \nabla_\theta, \qquad \mathcal{L}_{\mathrm{FD}} = -p^T \Gamma \nabla_p + \Gamma : \nabla_p^2. \tag{1.34}$$

A simple calculation reveals that

$$\mathcal{L}^*_{\mathrm{ham}} = \frac{1}{\beta} \sum_{i=1}^{d} \partial_{\theta_i} \partial^*_{p_i} - \partial_{p_i} \partial^*_{\theta_i} = -\mathcal{L}_{\mathrm{ham}},$$

$$\mathcal{L}^*_{\mathrm{FD}} = -\frac{1}{\beta} \sum_{i=1}^{d} \partial^*_{p_i} \partial_{p_i} = \mathcal{L}_{\mathrm{FD}},$$

when all operators are considered as operators on $L^2(\tau)$. The probability measure τ is then invariant for the dynamics (1.32) in view of (1.23). Pathwise ergodicity is proved for τ using the fact that the generator is hypoelliptic [73]. This allows to deduce that for a given test function ϕ,

$$\lim_{t\to\infty} \frac{1}{t} \int_0^t \phi(\theta_s)\, ds = \int_{\mathbb{R}^d} \phi(\theta) \mu(\theta)\, d\theta, \quad \text{a.s.}$$

Remark 1.8. *Let us mention that overdamped Langevin dynamics can be obtained from Langevin dynamics by two limiting processes: (i) when the friction* $\Gamma = \gamma \mathrm{I}_d$ *and* $\gamma \to \infty$ *with time rescaled as* γt*; (ii) when* $M = m\mathrm{I}_d$ *and* $m \to 0$*. See [93, Section 2.2] for precisions.*

Discretization of the dynamics. To discretize Langevin dynamics, one can use splitting schemes [3, 87, 86]. The idea is to split the generator into elementary parts whose associated elementary evolution dynamics can be analytically integrated. The weak error order of numerical schemes is determined by establishing estimates such as (1.27). The Baker–Campbell–Hausdorff formula [55] is generally used to obtain such estimations. For two operators A, B, this formula reads

$$\mathrm{e}^{\Delta t A}\mathrm{e}^{\Delta t B} = \mathrm{e}^{\Delta t C_\Delta t}$$

where

$$C_\Delta t = \Delta t(A+B) + \frac{\Delta t^2}{2}[A,B] + \mathcal{O}(\Delta t^3),$$

with

$$[A,B] = AB - BA$$

the commutator between the operators A and B. For the Langevin dynamics, the generator can be separated into three parts as

$$\mathcal{L}_{\mathrm{lan}} = \mathcal{L}_1 + \mathcal{L}_2 + \mathcal{L}_3, \tag{1.35}$$

with

$$\mathcal{L}_1 = -\nabla_\theta V^T \nabla_p, \qquad \mathcal{L}_2 = p^T \nabla_\theta, \qquad \mathcal{L}_3 = -p^T \Gamma \nabla_p + \frac{1}{\beta}\Gamma : \nabla^2. \tag{1.36}$$

The elementary generator $\mathcal{L}_1$ is associated with the elementary differential equation $dp_t = -\nabla_\theta V(\theta_t)\,dt$, and can be analytically integrated over a time Δt as

$$p^{m+1} = p^m - \Delta t \nabla V\left(\theta^m\right).$$

The elementary generator $\mathcal{L}_2$ is associated with the elementary differential equations $d\theta_t = p_t\,dt$ and can be analytically integrated over a time Δt as

$$\theta^{m+1} = \theta^m + \Delta t\, p^m.$$

The elementary generator $\mathcal{L}_3$ is associated with the Ornstein–Uhlenbeck process $dp_t = -\Gamma p_t\,dt + \sqrt{2\beta^{-1}}\Gamma^{1/2}\,dW_t$, and can also be analytically integrated over a time t as

$$p_t = \mathrm{e}^{-\Gamma t}p_0 + \sqrt{\frac{2}{\beta}}\int_0^t \mathrm{e}^{-(t-s)\Gamma}\Gamma^{1/2}\,dW_s \sim \mathcal{N}\left(\alpha_t p_0, \frac{\mathrm{I}_d - \alpha_{2t}}{\beta}\right), \qquad \alpha_t = \mathrm{e}^{-\Gamma t}.$$

First order schemes are based on a Lie-Trotter splitting, encoded by evolution operators such as

$$P_{\Delta t} = \mathrm{e}^{\Delta t \mathcal{L}_3}\mathrm{e}^{\Delta t \mathcal{L}_2}\mathrm{e}^{\Delta t \mathcal{L}_1}. \tag{1.37}$$

Of course other choices of orderings of $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_3$ are possible. The numerical scheme associated with (1.37) reads

$$\begin{cases} p^{m+\frac{1}{2}} = p^m - \Delta t \nabla V\left(\theta^m\right), \\ \theta^{m+1} = \theta^m + \Delta t\, p^{m+\frac{1}{2}}, \\ p^{m+1} = \alpha_{\Delta t}\, p^{m+\frac{1}{2}} + \left(\dfrac{\mathrm{I}_d - \alpha_{2\Delta t}}{\beta}\right)^{1/2} G^m, \end{cases} \tag{1.38}$$

where G^m is a vector of i.i.d. standard d-dimensional Gaussian random variables. Second-order schemes are obtained by a Strang splitting of the elementary evolutions, such as

$$P_{\Delta t} = \mathrm{e}^{\Delta t \mathcal{L}_3/2} \mathrm{e}^{\Delta t \mathcal{L}_2/2} \mathrm{e}^{\Delta t \mathcal{L}_1} \mathrm{e}^{\Delta t \mathcal{L}_2/2} \mathrm{e}^{\Delta t \mathcal{L}_3/2}. \tag{1.39}$$

Here as well, various other choices of orderings can be considered, see for instance [85, 87, 86] for a detailed analysis of the merits of the various options. The numerical scheme associated with (1.39) reads

$$\begin{cases} p^{m+\frac{1}{3}} = \alpha_{\Delta t/2}\, p^m + \left(\dfrac{\mathrm{I}_d - \alpha_{\Delta t}}{\beta}\right)^{1/2} G^m, \\ \theta^{m+\frac{1}{2}} = \theta^m + \dfrac{\Delta t}{2}\, p^{m+\frac{1}{3}}, \\ p^{m+\frac{2}{3}} = p^{m+\frac{1}{3}} - \Delta t \nabla V\left(\theta^{m+\frac{1}{2}}\right), \\ \theta^{m+1} = \theta^{m+\frac{1}{2}} + \dfrac{\Delta t}{2} p^{m+\frac{2}{3}}, \\ p^{m+1} = \alpha_{\Delta t/2}\, p^{m+\frac{2}{3}} + \left(\dfrac{\mathrm{I}_d - \alpha_{\Delta t}}{\beta}\right)^{1/2} G^{m+\frac{1}{2}}, \end{cases}$$

where $G^{m+\frac{1}{2}}$ and G^m are vectors of i.i.d. standard d-dimensional Gaussian random variables.

Remark 1.9. *In fact, one can reach second order accuracy on the invariant measure using a first order splitting between the Hamiltonian part (for which one uses a Verlet scheme [152]) and the Ornstein–Uhlenbeck part [22, 3]. These schemes, known as Geometric Langevin Algorithm, are encoded by evolution operators such as* $P_{\Delta t} = \mathrm{e}^{\Delta t \mathcal{L}_1/2} \mathrm{e}^{\Delta t \mathcal{L}_2} \mathrm{e}^{\Delta t \mathcal{L}_1/2} \mathrm{e}^{\Delta t \mathcal{L}_3}$. *As pointed out before, such schemes do not satisfy (1.27).*

1.2.2.4 Adaptive Langevin dynamics

We recall in this section the adaptive Langevin dynamics (AdL) [65, 39] which allows to sample from the Boltzmann–Gibbs distribution when the magnitude of diffusion in the Brownian motion is unknown. The extended version of Adaptive Langevin dynamics introduced in [137] (see Chapter 2) is build upon AdL.

AdL was first introduced in [65] to address the problem of sampling the canonical measure in computational statistical physics, *i.e.* the Boltzmann-Gibbs measure (1.12), while correcting for extra fluctuation terms leading to a spurious heating of the system. It is a combination of Nosé–Hoover (NH) dynamics [112, 63] and Langevin dynamics. We first recall NH dynamics, then motivate AdL and discuss some of its properties, and finally give elements on its numerical discretization.

Nosé–Hoover dynamics. Let us first introduce and recall some important properties of NH dynamics. The NH thermostat allows to dissipate the extra kinetic energy of the system, or on the contrary increase the missing kinetic energy, by adding a control variable $\xi \in \mathbb{R}$ that follows a negative feedback loop. This variable can be interpreted as some variable friction coefficient (which can take negative values). More precisely, the Nosé–Hoover dynamics is given by

$$\begin{cases} d\theta_t = p_t\, dt, \\ dp_t = -\nabla V(\theta_t)\, dt - \xi_t p_t\, dt, \\ d\xi_t = \dfrac{1}{\eta}\left(|p|^2 - \dfrac{d}{\beta}\right) dt. \end{cases} \tag{1.40}$$

One way to interpret the dynamics (1.40) is to see it as the superposition of a Hamiltonian dynamics to which an (anti)friction term is added to the evolution of momenta thanks to the

control variable ξ. The friction is adjusted following a negative feedback loop to preserve the Boltzmann–Gibbs measure. Namely, if $|p|^2 > d/\beta$, the kinetic energy is larger than what it should be in average under the probability measure (1.12), so the friction is increased, which ends up decreasing p and decreasing the kinetic energy. We can use the same line of argument for the opposite case when the kinetic energy is smaller than the target average value, in which case antifriction makes sure that the kinetic energy increases.

In order to make the above formal discussion more rigorous, we prove using the generators of (1.40) that the Nosé–Hoover dynamics preserves the Boltzmann–Gibbs distribution, as made precise in the following result.

Lemma 1.10. *The probability measure*

$$\nu_{\mathrm{NH}}(d\theta\, dp\, d\xi) = \mu(\theta)\kappa(dp)\varrho(d\xi)\, d\theta, \tag{1.41}$$

with κ is redefined as

$$\kappa(dp) = \left(\frac{\beta}{2\pi}\right)^{d/2} \exp\left(-\beta\frac{|p|^2}{2}\right) dp,$$

and

$$\varrho(d\xi) = \sqrt{\frac{\eta}{2\pi}} \exp\left(-\frac{\eta\xi^2}{2}\right) d\xi,$$

is invariant by the Nosé–Hoover dynamics (1.40).

The marginal of ν_{NH} in the variable θ is μ, so that the Nosé–Hoover dynamics preserves the Boltzmann–Gibbs measure. Let us emphasize that, although the dynamics leaves the probability measure (1.41) invariant, it is in general not ergodic for this measure, see [65, 83]. We give some elements of the proof following the approach used in [90, Section 2], remaining however brief since a very similar proof is written with more details in Section 2.3.2.1 of Chapter 2. To write the proof of lemma 1.10, we introduce the generator of the dynamics, which can be decomposed as

$$\mathcal{L}_{\mathrm{NHd}} = \mathcal{L}_{\mathrm{ham}} + \mathcal{L}_{\mathrm{NH}}, \tag{1.42}$$

where $\mathcal{L}_{\mathrm{ham}}$ is given in (1.34), and

$$\mathcal{L}_{\mathrm{NH}} = -p^T\xi\nabla_p + \frac{1}{\eta}\left(|p|^2 - \frac{d}{\beta}\right)\partial_\xi.$$

Proof. We use the characterization of invariant probability measure given by (1.22). It suffices to show that, for all smooth and compactly supported functions ϕ,

$$\int_\Theta \mathcal{L}_{\mathrm{NHd}}\phi\, d\nu_{\mathrm{NHd}} = \int_\Theta \phi\,(\mathcal{L}^*_{\mathrm{NHd}}\mathbf{1})\, d\nu_{\mathrm{NHd}} = 0, \tag{1.43}$$

and a similar equality with $\mathcal{L}_{\mathrm{NHd}}$ replaced by $\mathcal{L}_{\mathrm{ham}}$. In these equalities, $\mathcal{L}^*_{\mathrm{NHd}}$ is the adjoint of $\mathcal{L}_{\mathrm{NHd}}$ on $L^2(\nu_{\mathrm{NH}})$. Simple computations based on integration by parts show that $\partial^*_{\theta_i} = -\partial_{\theta_i} - \partial_{\theta_i}V$, $\partial^*_{p_i} = -\partial_{p_i} + p_i$, $\partial^*_\xi = -\partial_\xi + \eta\xi$. It can then be easily shown that $\mathcal{L}_{\mathrm{ham}}$ and $\mathcal{L}_{\mathrm{NHd}}$ are antisymmetric. Moreover, $\mathcal{L}_{\mathrm{NHd}}\mathbf{1} = \mathcal{L}_{\mathrm{ham}}\mathbf{1} = 0$. The invariance of ν_{NH} therefore follows from (1.43). □

General formulation and motivation for Adaptive Langevin dynamics. Langevin dynamics given by (1.32) is ergodic with respect to the Boltzmann–Gibs measure but cannot dissipate excess kinetic energy, or add missing kinetic energy, because the friction coefficient is fixed. The idea in [65] was to combine Langevin dynamics (with a scalar friction, namely

$\Gamma = \gamma \mathrm{I}_d$, to simplify the discussion) and a Nosé–Hoover thermostat to sample from μ with a variable friction. AdL as introduced in [65] reads

$$\begin{cases} d\theta_t = p_t\, dt, \\ dp_t = -\nabla V(\theta_t)\, dt - \xi_t p_t\, dt + \sqrt{2\sigma}\, dW_t, \\ d\xi_t = \dfrac{1}{\eta}\left(|p|^2 - \dfrac{d}{\beta}\right) dt. \end{cases} \tag{1.44}$$

One important property of AdL, which motivates its use in machine learning context [39], is that one can sample from the Boltzmann–Gibbs distribution even when the diffusion term σ is unknown (as discussed below). This is useful when the computation of the gradient is corrupted in a way that extra noise of unknown magnitude is introduced. This extra noise can be combined with the Brownian motion in Langevin dynamics as a single Brownian motion of unknown magnitude σ. This explains why AdL is sometimes called stochastic gradient Nosé–Hoover thermostat [39] in the machine learning literature. In this manuscript, we consider a matricial version of AdL, where $\xi_t \in \mathbb{R}^{d\times d}$, namely

$$\begin{cases} d\theta_t = p_t\, dt, \\ dp_t = -\nabla V(\theta_t)\, dt - \xi_t p_t\, dt + \sqrt{2} A^{1/2}\, dW_t, \\ d[\xi_t]_{i,j} = \dfrac{1}{\eta}\left(p_{i,t}p_{j,t} - \dfrac{\delta_{i,j}}{\beta}\right) dt, \quad 1 \leqslant i,j \leqslant d. \end{cases} \tag{1.45}$$

with $A \in \mathbb{R}^{d\times d}$ an unknown positive definite symmetric matrix.

Invariant probability measure. The generator of the Adaptive Langevin dynamics (1.45) can be decomposed as

$$\mathcal{L}_{\mathrm{AdL,A}} = \mathcal{L}_{\mathrm{ham}} + \mathcal{L}_{\mathrm{FD}} + \mathcal{L}_{\mathrm{NH}}, \tag{1.46}$$

where $\mathcal{L}_{\mathrm{ham}}$ and $\mathcal{L}_{\mathrm{FD}}$ are given by (1.34) (upon replacing Γ by A in $\mathcal{L}_{\mathrm{FD}}$) and

$$\mathcal{L}_{\mathrm{NH}} = -p^T(\xi - A)\nabla_p + \frac{1}{\eta}\sum_{1\leqslant i,j\leqslant d}\left(p_i p_j - \frac{\delta_{i,j}}{\beta}\right)\partial_{\xi_{i,j}}.$$

An easy adapation of Lemma 1.10 allows to prove that the AdL dynamics (1.45) admits the following invariant probability measure (see [65, 39] for a proof, as well as Section 2.3.2.1 in Chapter 2):

$$\nu(d\theta\, dp\, d\xi) = \mu(\theta)\kappa(dp)\rho(d\xi), \tag{1.47}$$

where ρ has a form similar to ϱ in (1.41) (upon shifting ξ by A):

$$\rho(d\xi) = \prod_{1\leqslant i,j\leqslant d}\sqrt{\frac{\eta}{2\pi}}\exp\left(-\frac{\eta}{2}(\xi_{i,j} - A_{i,j})^2\right) d\xi_{i,j}.$$

Here as well, the marginal probability measure of ν in the variable θ is μ. In contrast to Nosé–Hoover dynamics, ergodicity can be proved here. Pathwise ergodicity holds by the results of [73] since the generator (1.46) can be shown to be hypoelliptic. The mathematical properties of AdL are investigated in [90] in the scalar case (namely, for the dynamics (1.44)). The main result of the latter work is the exponential convergence in $L^2(\nu)$ of the semi–group $\mathrm{e}^{t\mathcal{L}_{\mathrm{AdL,A}}}$. Namely, under some growth assumptions on derivatives of the potential energy function V, and assuming that μ satisfies Poincaré inequality, the following result holds: there exist $\lambda > 0$ (depending on η and A) and a constant $C \in \mathbb{R}_+$ such that, for any $\phi \in L^2(\nu)$,

$$\forall t \geqslant 0, \qquad \left\| \mathrm{e}^{t\mathcal{L}_{\mathrm{AdL,A}}}\phi - \int \phi\, d\nu \right\|_{L^2(\nu)} \leqslant C\mathrm{e}^{-t\lambda}\left\| \phi - \int \phi\, d\nu \right\|_{L^2(\nu)}.$$

This implies in particular the uniquess of the invariant probability measure, and allows to derive a central limit theorem for time averages along one realization of the dynamics with bounds on the asymptotic variance of order $\mathcal{O}(\max(\sigma, \sigma^{-1}, \sigma\eta, \sigma^{-1}\eta^{-1}))$ for the scalar case of AdL (1.44).

Discretization of the dynamics. We now describe how to construct a numerical scheme to approximate the solution of AdL when A is known. We refer to Chapter 2 for numerical schemes in the more interesting case when A is not known, for instance when the computation of the gradient of the potential function is polluted by random errors. We consider the symmetric splitting scheme introduced in [90], which is based on the decomposition of (1.46) into four elementary operators that can be analytically integrated. More precisely, we decompose the generator as

$$\mathcal{L}_{\mathrm{AdL,A}} = \mathcal{L}_1 + \mathcal{L}_2 + \mathcal{T}_3 + \mathcal{T}_4,$$

where $\mathcal{L}_1$ and $\mathcal{L}_2$ are given in (1.36), and

$$\mathcal{T}_3 = \frac{1}{\eta} \sum_{1 \leqslant i,j \leqslant d} \left(p_i p_j - \frac{\delta_{i,j}}{\beta} \right) \partial_{[\xi]_{i,j}}, \qquad \mathcal{T}_4 = -p^T \xi \nabla_p + A : \nabla^2.$$

We consider the following evolution operator

$$P_{\Delta t} = \mathrm{e}^{\Delta t \mathcal{T}_4/2} \mathrm{e}^{\Delta t \mathcal{T}_3/2} \mathrm{e}^{\Delta t \mathcal{L}_2/2} \mathrm{e}^{\Delta t \mathcal{L}_1} \mathrm{e}^{\Delta t \mathcal{L}_2/2} \mathrm{e}^{\Delta t \mathcal{T}_3/2} \mathrm{e}^{\Delta t \mathcal{T}_4/2}.$$

When $A = a\mathrm{I}_d$, the numerical scheme is given by

$$\begin{cases} p^{m+\frac{1}{2}} = \mathrm{e}^{-\Delta t \xi^m/2} p^m + \left[a(\xi^m)^{-1} \left(\mathrm{I}_d - \mathrm{e}^{-\Delta t \xi^m} \right) \right]^{1/2} G^m, \\ \xi^{m+\frac{1}{2}} = \xi^m + \dfrac{\Delta t}{2\eta} \left(p^{m+\frac{1}{2}} \left(p^{m+\frac{1}{2}} \right)^T - \mathrm{I}_d \right), \\ \theta^{m+\frac{1}{2}} = \theta^m + \dfrac{\Delta t}{2} p^{m+\frac{1}{2}}, \\ \widetilde{p}^{m+\frac{1}{2}} = p^{m+\frac{1}{2}} - \Delta t \nabla V \left(\theta^{m+\frac{1}{2}} \right), \\ \theta^{m+1} = \theta^{m+\frac{1}{2}} + \dfrac{\Delta t}{2} \widetilde{p}^{m+\frac{1}{2}}, \\ \xi^{m+1} = \xi^{m+\frac{1}{2}} + \dfrac{\Delta t}{2\eta} \left(\widetilde{p}^{m+\frac{1}{2}} \left(\widetilde{p}^{m+\frac{1}{2}} \right)^T - \mathrm{I}_d \right), \\ p^{m+1} = \mathrm{e}^{-\Delta t \xi^{m+1}/2} \widetilde{p}^{m+\frac{1}{2}} + \left[a(\xi^{m+1})^{-1} \left(\mathrm{I}_d - \mathrm{e}^{-\Delta t \xi^{m+1}} \right) \right]^{1/2} G^{m+\frac{1}{2}}, \end{cases} \tag{1.48}$$

where $G^{m+\frac{1}{2}}$ and G^m are vectors of i.i.d. standard d-dimensional Gaussian random variables.

Remark 1.11. *Let us emphasize that the term* $(\xi^{m+1})^{-1} \left(\mathrm{I}_d - \mathrm{e}^{-\Delta t \xi^{m+1}} \right)$ *in the integration of the Ornstein–Uhlenbeck process is only valid when* ξ *is invertible. When* ξ *is singular or close to singular, this should be understood through a limiting procedure as explained for the scalar case in [90] and in Chapter 2. The procedure can be generalized to the matricial case by spectral calculus.*

Remark 1.12. *Using the matricial version of AdL requires evolving a matricial friction in* $\mathbb{R}^{d \times d}$*. This can be computationally expensive when* d *is large, especially since one needs to compute the inverse of this matrix at each time step.*

1.2.3 Other algorithms

The problem of sampling from a given probability measure is still a very active research area, and a myriad of other algorithms than Metropolis–Hastings and discretization of SDEs exist. Other MCMC algorithms include for example Gibbs samplers introduced in [52]. It is based on updating separately the components of the Markov chain. The main disadvantage of this method is that one needs to be able to sample from some marginals distributions of the target probability measure. Another popular algorithm is the Hamiltonian Monte Carlo method (initially named Hybrid Monte Carlo), introduced in [43], which uses a numerical integration of the Hamiltonian dynamics as proposal for the Metropolis–Hastings algorithm. The momenta are sampled at each iteration from the standard normal distribution. We refer for example to [13, 26, 96, 45] for a mathematical analysis of the algorithm.

Another class of algorithms is variational inference [66], see also [17] for a review. The main idea of variational inference algorithms is to approximate the target probability measure by a simpler one, from which one can easily sample. The sampling problem becomes an optimization problem. These types of algorithms are generally faster than MCMC methods, but give no guarantee that one is sampling from the exact desired distribution. A less explored area is to mix MCMC methods and variational inference algorithms, see for example [135]. Variational inference is also popular for to generative methods in machine learning community. This is discussed in Chapter 4, where we use variational autoencoders as a generative method to produce transition paths as discussed in 1.4.3.

1.3 Sampling methods in the large data context

All the methods introduced in Section 1.2 require the computation of either V or ∇V at each iteration. In the Bayesian context where $V = -\log \pi$, this means that either π and/or its gradient need to be estimated at each iteration. The cost of computing these quantities scales as the number of data points $\mathcal{O}(N_{\text{data}})$, and is often the computational bottleneck in implementations of MCMC algorithms. In this section, we recall some methods from the literature to reduce the computational time of MCMC methods in the Bayesian context when the number of data points is large. The aim is to cut the cost of one iteration from $\mathcal{O}(N_{\text{data}})$ to only a fraction of this cost. We first recall in Section 1.3.1 the minibatching method used for MCMC methods based on the discretization of SDEs, and recall some important results on the error introduced by minibatching on the target measure. We then briefly review in Section 1.3.2 some methods to reduce the computational cost of Metropolis–Hastings algorithms.

1.3.1 Methods based on estimators of the log-likelihood gradient

Using MCMC methods based on the discretization of SDEs to sample from the posterior probability measure (as those presented in Section 1.2.2) requires, at each time step, the computation of $\nabla_\theta \log \pi(\cdot|\mathbf{x})$. For π given by (1.1) or (1.9), the gradient reads

$$\nabla_\theta \left[\log \pi(\theta|\mathbf{x})\right] = \nabla_\theta \left[\log P_{\text{prior}}(\theta)\right] + \sum_{i=1}^{N_{\text{data}}} \nabla_\theta \left[\log P_{\text{elem}}(x_i|\theta)\right]. \tag{1.49}$$

The evaluation of this gradient has a computational cost scaling as $\mathcal{O}(N_{\text{data}})$, which is computationally demanding when N_{data} is large. We present an approach called minibatching, to limit the cost to a fraction of $\mathcal{O}(N_{\text{data}})$ by approximating the gradient; and then discuss the bias due to minibatching induced on the invariant probability measure actually sampled by the numerical methods.

1.3.1.1 Minibatching

One way to limit the cost of evaluating (1.49) is to use an unbiased estimator based on a mini-batch [126] of the complete data set to approximate the exact gradient. More precisely, considering a minibatch of n points out of N_{data}, the gradient of the posterior distribution is approximated by

$$\widehat{F}_n(\theta) = \nabla_\theta \left[\log P_{\text{prior}}(\theta)\right] + \frac{N_{\text{data}}}{n} \sum_{i \in I_n} \nabla_\theta \left[\log P_{\text{elem}}(x_i|\theta)\right], \tag{1.50}$$

where I_n is a random subset of size n generated by sampling uniformly indices from $\{1, ..., N_{\text{data}}\}$. There are various strategies to do so, which we now make precise. In the discussion below, all expectations are with respect to realizations of I_n, the values of the parameters θ being fixed.

Sampling with replacement. If I_n is generated by sampling indices with replacement from the set $\{1, ..., N_{\text{data}}\}$, the computation of the variance of the estimator of the gradient is straightforward: denoting by $g_i = \nabla_\theta \left[\log P_{\text{elem}}(x_i|\theta)\right]$, it follows that

$$\text{cov}\left(\widehat{F_n}(\theta)\right) = \frac{N_{\text{data}}^2}{n^2} \text{cov}\left(\sum_{i \in I_n} \nabla_\theta \left[\log P_{\text{elem}}(x_i|\theta)\right]\right) = \frac{N_{\text{data}}(N_{\text{data}} - 1)}{n} \text{cov}(g_1). \tag{1.51}$$

When sampling with replacement, it can occur that the same data point is used twice in the computations of the gradient estimator $\widehat{F}_n$. In this case, even for $n = N_{\text{data}}$ the gradient estimator is noisy.

Sampling without replacement. If I_n is generated by sampling n indices without replacement from the set $\{1, ..., N_{\text{data}}\}$, one can show that

$$\text{cov}\left(\widehat{F_n}(\theta)\right) = \frac{N_{\text{data}}(N_{\text{data}} - n)}{n} \text{var}(g_1). \tag{1.52}$$

When sampling without replacement, no data point can be used twice in the computations of the gradient estimator $\widehat{F}_n$. In this case, for $n = N_{\text{data}}$ the covariance matrix is zero and the gradient estimator is exact. To prove (1.52), we introduce the random variable $\mathbf{1}_{i \in I_n}$ which represents whether i is in I_n or not. A simple computation shows that

$$\text{var}(\mathbf{1}_{i \in I_n}) = \frac{n}{N_{\text{data}}} - \frac{n^2}{N_{\text{data}}^2}.$$

For $i \neq j$, the random variable $\mathbf{1}_{i \in I_n} \mathbf{1}_{j \in I_n}$ corresponds to the probability of drawing the set of indices $\{i, j\}$ among the n indices. Events where it has value 1 can be obtained by computing the probability of drawing i among the n indices out of N_{data} points, and then the probability to have j among the remaining $n-1$ indices out of the remaining $N_{\text{data}}-1$ points. Therefore, $\mathbb{E}[\mathbf{1}_{i \in I_n} \mathbf{1}_{j \in I_n}] = \frac{n(n-1)}{N_{\text{data}}(N_{\text{data}}-1)}$, so that

$$\text{cov}(\mathbf{1}_{i \in I_n}, \mathbf{1}_{j \in I_n}) = \frac{n(n-1)N_{\text{data}} - n^2(N_{\text{data}} - 1)}{N_{\text{data}}^2(N_{\text{data}} - 1)} = -\frac{n(N_{\text{data}} - n)}{N_{\text{data}}^2(N_{\text{data}} - 1)}.$$

Using these equalities, one can then compute the covariance of the estimator of the gradient of the log-likelihood as

$$\begin{aligned}\operatorname{cov}\left(\widehat{F_n}(\theta)\right) &= \frac{N_{\text{data}}^2}{n^2}\operatorname{cov}\left(\sum_{i=1}^{N_{\text{data}}} \nabla_\theta \left[\log P_{\text{elem}}(x_i|\theta)\right] \mathbf{1}_{i\in I_n}\right)\\ &= \frac{N_{\text{data}}^2}{n^2}\sum_{i=1}^{N_{\text{data}}} \operatorname{var}\Big(\nabla_\theta \left[\log P_{\text{elem}}(x_i|\theta)\right] \mathbf{1}_{i\in I_n}\Big)\\ &+ \frac{2N_{\text{data}}^2}{n^2}\sum_{1\leqslant i<j\leqslant N_{\text{data}}} \operatorname{cov}\Big(\nabla_\theta \left[\log P_{\text{elem}}(x_i|\theta)\right] \mathbf{1}_{i\in I_n}, \nabla_\theta \left[\log P_{\text{elem}}(x_j|\theta)\right] \mathbf{1}_{j\in I_n}\Big).\end{aligned}$$

Denoting by $g = \frac{1}{N_{\text{data}}}\sum_{i=1}^{N_{\text{data}}} g_i$ to alleviate the notation, it follows that

$$\begin{aligned}\operatorname{cov}\left(\widehat{F_n}(\theta)\right) &= \frac{N_{\text{data}}^2}{n^2}\left(\sum_{i=1}^{N_{\text{data}}} g_i g_i^T \operatorname{var}\left(\mathbf{1}_{i\in I_n}\right) + 2\sum_{1\leqslant i<j\leqslant N_{\text{data}}} g_i g_j^T \operatorname{cov}(\mathbf{1}_{i\in I_n}, \mathbf{1}_{j\in I_n})\right)\\ &= \frac{N_{\text{data}}^2}{n^2}\left(\sum_{i=1}^{N_{\text{data}}} g_i g_i^T \frac{n(N_{\text{data}}-n)}{N_{\text{data}}^2} - 2\sum_{1\leqslant i<j\leqslant N_{\text{data}}} g_i g_j^T \frac{n(N_{\text{data}}-n)}{N_{\text{data}}^2(N_{\text{data}}-1)}\right)\\ &= \frac{N_{\text{data}}^2}{n^2}\frac{n(N_{\text{data}}-n)}{N_{\text{data}}^2}\left(\sum_{i=1}^{N_{\text{data}}} g_i g_i^T - \frac{2}{N_{\text{data}}-1}\sum_{1\leqslant i<j\leqslant N_{\text{data}}} g_i g_j^T\right)\\ &= \frac{N_{\text{data}}-n}{n}\left(\left(1+\frac{1}{N_{\text{data}}-1}\right)\sum_{i=1}^{N_{\text{data}}} g_i g_i^T - \frac{1}{N_{\text{data}}-1}\sum_{i,j=1}^{N_{\text{data}}} g_i g_j^T\right)\\ &= \frac{N_{\text{data}}-n}{n}\left(\frac{N_{\text{data}}}{N_{\text{data}}-1}\sum_{i=1}^{N_{\text{data}}} g_i g_i^T - \frac{N_{\text{data}}^2}{N_{\text{data}}-1} g g^T\right)\\ &= \frac{N_{\text{data}}-n}{n}\frac{N_{\text{data}}^2}{N_{\text{data}}-1}\left(\frac{1}{N_{\text{data}}}\sum_{i=1}^{N_{\text{data}}} g_i g_i^T - g g^T\right)\\ &= \frac{N_{\text{data}}(N_{\text{data}}-n)}{n}\operatorname{var}(g_1).\end{aligned}$$

Reshuffling. A popular way to perform minibatching while ensuring that all data points are considered is reshuffling [12]. In this framework, the algorithm proceeds by epochs (cycles). Each epoch is composed of N_{data}/n iterations of the numerical method (assuming that N_{data}/n is an integer), so that all data points are seen once during each epoch. At each epoch e, one draws a permutation σ_e of $\{1, ..., N_{\text{data}}\}$. At each iteration $\ell \in \{1, \dots, N_{\text{data}}/n\}$, the estimator of the gradient is computed as

$$\widehat{F}_n(\theta) = \nabla_\theta \left[\log P_{\text{prior}}(\theta)\right] + \frac{N_{\text{data}}}{n}\sum_{i=n\ell+1}^{n(\ell+1)} \nabla_\theta(\log P_{\text{elem}}(x_{\sigma_e(i)}|\theta)).$$

This method is popular in the machine learning community, in particular for training neural networks. We will use it in Chapter 3. However, it does not easily allow to discuss the statistical properties of the gradient estimator, in contrast to estimators based on sampling with or without replacement (see the next paragraph), which is why we restrict ourselves to the latter case in the mathematical analysis we perform in Chapter 2.

Properties of minibatching estimators when sampling with(out) replacement. When sampling with or without replacement, a straightforward computation shows that $\widehat{F}_n$ is a unbiased estimator of $\nabla_\theta \log \pi(\cdot|\mathbf{x})$. One can rewrite (1.50) as (see Section 2.2.2 in Chapter 2 for more details and comments on the computations)

$$\widehat{F_n}(\theta) = \nabla_\theta(\log \pi(\theta|\mathbf{x})) + \sqrt{\varepsilon(n)}\Sigma_{\mathbf{x}}^{\frac{1}{2}}(\theta) Z_{\mathbf{x},N_{\text{data}},n}, \tag{1.53}$$

where $Z_{\mathbf{x},N_{\text{data}},n}$ is a centered random variable with covariance I_d, and $\Sigma_{\mathbf{x}}(\theta)$ is the empirical covariance of the gradient estimator for $n = 1$:

$$\Sigma_{\mathbf{x}}(\theta) = \mathrm{cov}_{\mathcal{I}}\left[\nabla_\theta(\log P_{\text{elem}}(x_{\mathcal{I}}|\theta))\right], \tag{1.54}$$

where the expectation is taken over $\mathcal{I}$ uniformly distributed in $\{1, ..., N_{\text{data}}\}$. The term $\varepsilon(n)$ in (1.53) represents the noise magnitude. Its expression depend on the way the sampling is performed: for sampling with replacement, it holds, in view of [30, 154] (recall (1.51)),

$$\varepsilon(n) = \frac{N_{\text{data}}(N_{\text{data}} - 1)}{n}, \tag{1.55}$$

while, for sampling without replacement (see [30, 154]) (recall (1.52)),

$$\varepsilon(n) = \frac{N_{\text{data}}(N_{\text{data}} - n)}{n}. \tag{1.56}$$

When N_{data} and n are sufficiently large, the random variable $Z_{\mathbf{x},N_{\text{data}},n}$ approximately follows a centered reduced normal distribution; but this may not be the case in the nonasymptotic regime, see Section 2.2.2 for more details.

1.3.1.2 Minibatching error for stochastic differential equations

Using minbatching to sample from a probability measure induces an extra bias due to the noise term in (1.53). When using a gradient estimator $\widehat{F}_n$ in conjunction with a discretization of overdampled Langevin dynamics, one obtains the the so-called Stochastic Gradient Langevin Dynamics (SGLD) introduced in [157] and inspired by the Stochastic Gradient Dynamics (SGD), used for optimization [126]. In [157] the authors suggest to use decreasing time step to remove the extra bias; see [147] for a mathematical analysis. For fixed time steps, an analysis of the asymptotic and non-asymptotic bias is provided in [154]. The results show that SGLD has a weak error of order one, similarly to the Euler–Maruyama discretization of overdamped Langevin dynamics, but with an extra term in the leading error arising from the mini-batching procedure.

Quantifying the bias due to minibatching, and reducing it as much as possible, is one important contribution of this thesis. We refer to Section 1.4.1 for a summary of the contributions in this direction, and to Chapter 2 for: (i) a more detailed literature review; (ii)refinements on the analysis of the bias for SGLD and the discretization of Langevin dynamics with mini-batching; (iii) an analysis of the (reduced) minibatching bias when using AdL.

1.3.2 Metropolis–Hastings based algorithms for large data sets

A key step in the Metropolis–Hastings algorithm is the computation of the ratio r in (1.17) at each time step. In the context of Bayesian inference, this ratio involves the computation of

$$\frac{T(\tilde{\theta}_{k+1}, d\theta_k)\pi(d\tilde{\theta}_{k+1})}{T(\theta_k, d\tilde{\theta}_{k+1})\pi(d\theta_k)} = \frac{T(\tilde{\theta}_{k+1}, d\theta_k)P_{\text{prior}}(d\tilde{\theta}_{k+1})}{T(\theta_k, d\tilde{\theta}_{k+1})P_{\text{prior}}(d\theta_k)} \prod_{i=1}^{N_{\text{data}}} \frac{P_{\text{elem}}(x_i|\tilde{\theta}_{k+1})}{P_{\text{elem}}(x_i|\theta_k)}. \tag{1.57}$$

The evaluation of this quantity (or its logarithm) has a computational cost scaling as $\mathcal{O}(N_{\text{data}})$. When the number of data points is large, one may be willing to introduce some bias on the

invariant probability measure which is sampled in order to reduce the cost of an iteration, and hence increase the number of iterations N_{iter}, as a way of reducing the variance of the average quantities which are computed (in the same spirit as mininatching techniques).

The key step to ensure the unbiasedness of MH algorithms is the acceptance/rejection decision. When using all the data points to compute the ratio (1.57), the decision which is taken is always correct. The idea behind most modified MH in the large data setting is to take decisions which are not always correct, but are correct with a probability as close to 1 as possible. This leads to a bias on the target measure which is effectively sampled, which is bias which however decreases as the number of data points used in the decision increases.

Let us describe more precisely one way to do so. In [75], the acceptance probability r given by (1.17) is approximated by a (random) acceptance probability r^*, computed as (1.57) but with only a random subset of the data, obtained in practice by sampling without replacement. The decision is then taken using some sequential hypothesis test. One fixes a threshold that should separate r^* and the uniform random variable U used in the acceptance test. As long as $|r^* - U|$ is too small compared to the estimated standard deviation of r^* (obtained by an empirical estimate based on a rewriting of (1.57) in a logarithmic form), more data points are added in order to refine the estimate of r^*, until $|r^* - U|$ is sufficiently large for a decision to be unambiguously taken. An important aspect of the above algorithm is to quantify the threshold on $|r^* - U|$ to decide when to take the decision. This can be estimated using concentration inequalities as in [11], although, from a technical viewpoint, one drawback of these inequalities is that they require bounds on the random variables, or on their (exponential) moments, which cannot always be guaranteed. In any case, in all these approaches, the stopping time before a decision is taken is not controlled, so that one can end up making a number of computations similar to the one of the standard MH algorithm.

1.4 Contributions

The goal of this final section of the introduction is to briefly present the main results and contributions of this PhD work.

1.4.1 Removing the mini-batching error in Bayesian inference using Adaptive Langevin dynamics

This work, presented in Chapter 2, is preprinted in [137] and submitted to *The Journal of Machine Learning Research*.

As pointed in Section 1.3.1, minibatching is a simple way to effectively reduce the computational time of MCMC based on the discretization of SDEs. The noise introduced by the gradient estimator is encoded in the term (see (1.53)) $\sqrt{\varepsilon(n)}\Sigma_{\mathbf{x}}^{\frac{1}{2}}(\theta)Z_{\mathbf{x},N_{\text{data}},n}$. For our analysis, we place ourselves in the case where $Z_{\mathbf{x},N_{\text{data}},n}$ is not necessarily a standard Gaussian random variable. Chapter 2 is first dedicated to the numerical analysis of minibatching error in Bayesian inference for Langevin-like algorithms. The first contribution is a refinement of first order weak error estimates on the invariant probability measure in term of $\varepsilon(n)$ and $\Sigma_{\mathbf{x}}$. In particular, we show that the bias on the invariant probability measure is of order $\mathcal{O}((1+\varepsilon(n))\Delta t)$ for SGLD and for of order $\mathcal{O}((\Delta t+\varepsilon(n))\Delta t)$ for second order discretizations of Langevin dynamics.

The second part of Chapter 2 is dedicated to the adaptive Langevin dynamics already introduced in Section 1.2.2.4. We first numerically challenge the hypothesis of $\Sigma_{\mathbf{x}}$ being constant, in which case minibatching procedure induces no bias on the posterior distribution when using the discretization of AdL. While one can analytically prove that $\Sigma_{\mathbf{x}}$ is constant in the Gaussian case, we demonstrate by numerical computations that $\Sigma_{\mathbf{x}}$ genuinely depends on θ in some situations (namely when the elementary likelihood is given by a mixture of

Gaussians). We provide an analysis of minibatching error for the discretization of AdL in the case where $\Sigma_{\mathbf{x}}$ depends on θ, showing that the error is again linear in $\varepsilon(n)$ provided that $\varepsilon(n)\Delta t$ is small enough. An upper bound for of the error can at dominant order be summarized as

$$\varepsilon(n)\Delta t \|\Sigma_{\mathbf{x}} - S^*\|_{L^2(\pi)},$$

where

- $S^* = \frac{1}{d}\overline{\Sigma}_{\mathbf{x}} = \int_\Theta \Sigma_{\mathbf{x}}(\theta)\pi(\theta|\mathbf{x})\, d\theta$ for matricial AdL;
- S^* is a diagonal matrix with entries $\int_\Theta [\Sigma_{\boldsymbol{x}}(\theta)]_{i,i}\, \pi(\theta|\boldsymbol{x})\, d\theta$, where $1 \leqslant i \leqslant d$ for diagonal AdL;
- $S^* = \frac{1}{d}\mathrm{Tr}(\overline{\Sigma}_{\mathbf{x}})\mathrm{I}_d$ for scalar AdL.

The last contribution of the work is the introduction of an extended version of AdL that allows to systemically reduce the bias. Under the assumption that the covariance of the force estimator can be decomposed on a finite basis of functions, the presented dynamics allows to fully remove the minibatching bias. We show that in this case S^* corresponds to the $L^2(\pi)$ projection of $\Sigma_{\mathbf{x}}$ onto the vector space of symmetric matrices generated by the basis. We finally give some insights about the choice of the basis functions.

The results of the Chapter 2 are illustrated by various numerical examples: (i) the Gaussian case; (ii) the case of mixture of Gaussians; (iii) Bayesian logistic regression.

1.4.2 Bayesian neural networks

This work started at the University of Edinburgh (School of Mathematics) during a two months PhD mobility program funded by the doctoral school MSTIC, through interactions with Ben Leimkuhler and Tiffany Vlaar. In Chapter 3, we present the preliminary results of this work on which we are currently still working.

We tackle the sampling of Bayesian neural network posterior distributions. Following results from Chapter 2, we start by numerically analyzing the structure of the covariance matrix of the estimator of the force in this context. For that, we consider two numerical models: (i) a toy classification model; (ii) a toy spiral data [88]. The main conclusions obtained by extensive numerical simulations on this models can be summarized as follows.

- The mean of $\Sigma_{\mathbf{x}}$ seems to be roughly isotropic, suggesting that the scalar, diagonal and matricial versions of AdL will exhibit the same bias on the invariant probability measure that they effectively sample;
- The covariance matrix is of low rank, suggesting that it can efficiently be approximated in an unexpensive way. This opens the way to an alternative strategy to the full matricial AdL, where instead of learning the mean of $\Sigma_{\mathbf{x}}$, one would learn the first eigenvalues of it, thus reducing the bias without resorting to a matricial friction (which is infeasible in neural networks framework).
- The particular shape of the covariance matrix associated with the last layer suggests to use AdL only for this particular layer. Note that sampling the last layer only has already been suggested in [77].

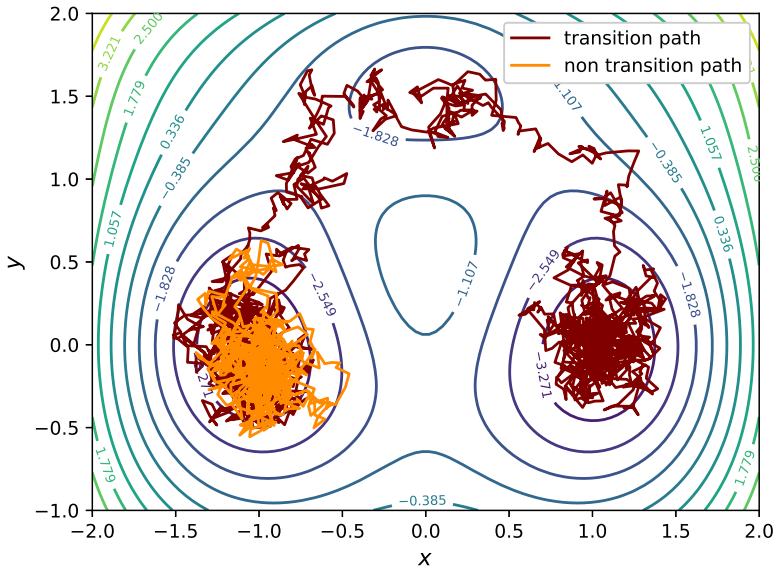


Figure 1.1 – Transition path and non transition path for the 2-dimensional potential given by (4.3).

1.4.3 Generative methods for sampling transition paths in molecular dynamics

The material for the Chapter 4 has been preprinted in [92] (submitted to *ESAIM Proceedings* in the special issue gathering contributions from the research projects initiated during the 6 week long summer research school CEMRACS 2021).

We consider the Euler–Maruyama discretization of the overdamped Langevin dynamics in the context of molecular dynamics. We place ourselves in the case where the potential V has many local minimas. One main issue in this situation is metastability: the trajectory remains trapped in restricted subdomains of the phase space, preventing an accurate sampling. Generating transition paths, that we define as trajectories which, from a fixed initial condition q_0 located in an initial potential well A reach a prespecified set $B \in \mathbb{R}^d$ before time $T \geqslant 0$, is of particular interest. Take for example the Figure 1.1 where we plot two typical trajectories using the Euler–Maruyama discretization of the overdamped Langevin dynamics with potential function (4.3) introduced in Chapter 4. The orange trajectory is not a transition path as it remains trapped in the first well A; whereas the red one jumps from the well A to B and therefore is a transition path. Such trajectories are rare, and even more rare as the dimension increases.

The goal of Chapter 4 is to use generative models from the machine learning literature to generate transition paths. We start by considering data driven generative approaches, namely variational autoencoders, first with a bi-dimensional embedding space, which turned out to be too small to account for the fluctuations of the whole trajectory, leading to too smooth trajectories (either reconstructed or generated). Incorporating the temporal aspect in the embedding space allows for better reconstructed trajectories. However generating new ones requires a refined modeling of the distribution on the latent space, and lead to poor results. We therefore turn to data free approaches, using reinforcement learning techniques. This allows to learn a perturbation on the drift of overdamped Langevin dynamics by maximizing some reward function, making transition from one well to another more likely. The results obtained this way are more realistic than with data based generative methods.

CHAPTER 2

REMOVING THE MINI-BATCHING ERROR IN BAYESIAN INFERENCE USING ADAPTIVE LANGEVIN DYNAMICS

Contents

The material for this chapter has been released in [137] and is currently under review.

Abstract. Bayesian inference allows to obtain useful information on the parameters of models, either in computational statistics or more recently in the context of Bayesian Neural Networks. The computational cost of usual Monte Carlo methods for sampling a posteriori laws in Bayesian inference scales linearly with the number of data points. One option to reduce it to a fraction of this cost is to resort to mini-batching in conjunction with unadjusted discretizations of Langevin dynamics, in which case only a random fraction of the data is used to estimate the gradient. However, this leads to an additional noise in the dynamics and hence a bias on the invariant measure which is sampled by the Markov chain. We advocate using the so-called Adaptive Langevin dynamics, which is a modification of standard inertial Langevin dynamics with a dynamical friction which automatically corrects for the increased noise arising from mini-batching. We investigate the practical relevance of the assumptions underpinning Adaptive Langevin (constant covariance for the estimation of the gradient), which are not satisfied in typical models of Bayesian inference, and quantify the bias induced by minibatching in this case. We also show how to extend AdL in order to systematically reduce the bias on the posterior distribution by considering a dynamical friction depending on the current value of the parameter to sample.

2.1 Introduction

Bayesian modeling allows to determine the distribution of parameters in statistical models, and hence to estimate functions of these parameters and their uncertainties. The Bayesian approach to neural networks in particular has recently gained some attention from the Machine Learning community; see for instance [155, 64] and references therein. Since the distributions of parameters are given by (possibly very) high dimensional probability measures, Markov Chain Monte Carlo (MCMC) techniques [128, 23] are the default method to sample these target measures. In this work, we only consider probability distributions having a density with respect to the Lebesgue measure. Quantities of interest are then expectations with respect to the target distribution, which are approximated by Monte Carlo estimates based on ergodicity results for the Markov chains under consideration. Let us however mention, that, beyond using MCMC methods to sample from distributions of parameters, these methods can also be used for optimization when run at small target temperature, which may be beneficial to explore complex non-convex energy landscapes (see for instance [99, 88, 38]), or ensure some form of regularization in an attempt to avoid overfitting (as already noted in [157]).

Two major classes of MCMC techniques can be distinguished. The first one gathers methods based on the Metropolis–Hastings algorithm [103, 58]. The second class of MCMC techniques relies on discretizations of stochastic differential equations (SDEs) which are ergodic with respect to the target measure. The SDEs used to sample from a probability distribution were originally introduced in molecular dynamics [48, 148, 86, 5], where atomic configurations in a system are typically distributed according to the Boltzmann–Gibbs distribution $\pi(d\theta) = Z^{-1} \exp(-V(\theta))\, d\theta$, where Z is the normalization constant and V the potential energy function of the system. Two prominent examples of such dynamics are Langevin dynamics (which we consider in this work as being the underdamped, or kinetic version) and its overdamped limit. In practice, methods from the two classes can be blended, as for the Metropolis-adjusted Langevin algorithm [113, 132, 129], which is a Metropolis–Hastings algorithm whose proposal function is provided by a Euler–Maruyama discretization of overdamped Langevin dynamics. We focus in this work on the second class of techniques, namely the discretization of SDEs. Langevin-like dynamics are gaining increasing popularity in Machine Learning and related application fields (see for instance [31, 109, 54] to quote just a few works).

Both Metropolis–Hastings methods and discretization of SDEs can be computationally expensive in the context of Bayesian inference since the log-likelihood and/or its gradient have to be evaluated at each step, either to perform a Metropolis test or to compute the forces in the dynamics. The cost of computing the log-likelihood and its gradient scales as $\mathcal{O}(N_{\text{data}})$, where N_{data} is the size of the data set, which may be large. Some variations of Metropolis-like algorithms, based on estimates of the log-likelihood obtained from a random subsample of the data, have been introduced to reduce the computational cost of one iteration; see for example [75, 11, 122]. These methods however require some prior knowledge on the target measure, and/or introduce some bias on the measure actually sampled by the algorithm. A similar approach was proposed for discretizations of SDEs by Welling and Teh in [157], who suggested to use a mini-batch of the data to construct an estimator of the gradient of the log-likelihood, leading to the so-called Stochastic Gradient Langevin Dynamics (SGLD).

SGLD however also induces biases on the sampled invariant measure, which have two origins: the finiteness of the time step and the fact that the mini-batch size n is smaller than N_{data}. It is possible to remove the bias by using decreasing time steps, as initially suggested in [157] and mathematically analyzed in [147], but this is not practical for the convergence of longtime averages, in particular when there is some metastability in the system (*i.e.* in situations when the posterior probability measure is multimodal, and the numerical methods remain temporarily stuck in one of the modes before finding their way to another one). An analysis of the asymptotic bias of SGLD for fixed step sizes is provided in [154]

(together with some analysis of the non-asymptotic bias, following the approach developed in [102]). The results show that SGLD has a weak error of order one, similarly to the Euler-Maruyama discretization of overdamped Langevin dynamics, but with an extra term in the leading error arising from the mini-batching procedure. In fact, the magnitude of this term makes it the dominant one, unless n is very close to N_{data}.

Various extensions and refinements of SGLD were proposed, in an attempt to improve the performance of the method and/or to reduce its bias. A first trend is to apply the mini-batching philosophy to dynamics which are more efficient in terms of sampling than overdamped Langevin dynamics, in particular Langevin dynamics [100], Hybrid Monte Carlo algorithms [31] and piecewise deterministic Markov processes [116, 15]. A second trend is to rely on control variate techniques to reduce the covariance of the stochastic estimator of the gradient of the log-likelihood, computed in practice by Gaussian approximations of the modes of the target probability measure; see for instance [110, 24, 9]. Of course, both trends can (should) be combined.

Our emphasis in this work is on the adaptive Langevin dynamics (AdL), introduced in [65, 39], which provides a way to reduce, and possibly even remove the bias arising from mini-batching in SGLD. This dynamics is a modification of Langevin dynamics where the friction is considered as a dynamical variable that adjusts itself so that the distribution of the velocities is correct. In addition to correcting for the mini-batching bias, it can also be used to train neural networks [88]. AdL was mathematically studied in [90], and further tested from a numerical viewpoint in [138]. The method completely removes the mini-batching bias when the covariance matrix $\Sigma_{\mathbf{x}}(\theta)$ of the estimator of the gradient is constant in the range of parameters explored by the method. This assumption is satisfied for Gaussian a posteriori distributions, as obtained when considering a Gaussian prior and Gaussian likelihoods, or when the number of data points is large enough and the a posteriori distribution concentrates around a Gaussian distribution according to the Bernstein–von Mises theorem (see for instance [150, Section 10.2]). There are however situations where this assumption does not hold (as we highlight on a numerical example in Section 2.3.3.2), in which case AdL fails to correct for the bias arising from mini-batching.

In this paper, we consider the case of extreme mini-batching procedures, corresponding to n as small as 1, as in initial works on stochastic approximation [126]. In this situation, no central limit theorem can be invoked to precisely characterize the statistical properties of the stochastic gradient estimator. This is in contrast with various works which assume the mini-batching noise to be Gaussian. This is however not needed to make precise the weak error of numerical methods, and hence the bias on the invariant probability measure, as we show in our estimates on the bias for SGLD and Langevin dynamics with stochastic gradient estimators. One of our contributions is to precisely quantify how the bias on the invariant measure depends on the type (with or without replacement) and amount of mini-batching, through some key parameter $\varepsilon(n)$ defined in Section 2.2.2. We also carefully study the covariance matrix of the stochastic gradient estimator in illustrative numerical examples, highlighting that this matrix may exhibit substantial variations in the range of parameters explored by the dynamics under consideration, so that the key assumption underlying AdL is not satisfied in general. Nonetheless, our numerical analysis explains why AdL still succeeds to substantially reduce the bias compared to SGLD and Langevin dynamics. We finally introduce an extended version of AdL allowing to even further reduce the bias incurred by non constant covariance matrices. One of our main contributions is to prove that, for all the Langevin-type SDEs considered in this paper, the bias introduced by the minibatching procedure is controlled by

$$\varepsilon(n)\Delta t \min_{S\in\mathscr{S}} \|\Sigma_{\mathbf{x}} - S\|_{L^2(\pi)} = \varepsilon(n)\Delta t \|\Sigma_{\mathbf{x}} - S^*\|_{L^2(\pi)}, \tag{2.1}$$

where Δt is the time step used to discretize the SDE at hand, and π is the target posterior

distribution. The vector space of matrix valued functions $\mathscr{S}$ depends on the chosen dynamics. For discretizations of standard Langevin dynamics, $S^* = 0$ (see Sections 2.2.3.2 and 2.2.4.2), whereas S^* corresponds to the average of $\Sigma_{\mathbf{x}}$ with respect to π for AdL (see Section 2.3.3.1) and to the $L^2(\pi)$-projection of $\Sigma_{\mathbf{x}}$ onto the vector space of symmetric matrices generated by some basis of functions for the extended version of AdL we introduce (see Section 2.4.2). We verify that the reduction in the bias is proportional to the quality of the approximation of the covariance matrix on a basis of functions (*e.g.* piecewise constant functions). In its simplest forms, our extension of AdL is very similar to the original AdL method, but allows for some extra flexibility which can dramatically reduce the bias.

The paper is organized as follows. We start in Section 2.2 by reviewing various results related to the numerical analysis of SDEs and the quantification of the bias on the invariant measure sampled by SGLD and Langevin dynamics with stochastic gradient estimators. We next turn to AdL in Section 2.3: we illustrate that some residual bias remains present due to the fact that the covariance of the gradient estimator is not constant in general, and we quantify it. Section 2.4 is dedicated to the introduction of an extended version of AdL that allows to accomodate non constant covariance matrices for the gradient estimator. The theoretical predictions made in Sections 2.3 and 2.4 are illustrated in Section 2.5 on a model of Gaussian mixtures and logistic regression for the classification of MNIST data. Some conclusions and perspectives are gathered in Section 2.6.

2.2 Stochastic gradient Markov Chain Monte Carlo

We consider a Bayesian inference problem where we denote by $\mathbf{x} = (x_i)_{i=1,\dots,N_{\text{data}}} \in \mathcal{X}^{N_{\text{data}}}$ a set of N_{data} data points, with $\mathcal{X} \subset \mathbb{R}^{d_{\text{data}}}$. The data points are assumed to be independent and identically distributed (i.i.d.) with respect to an elementary likelihood probability measure $P_{\text{elem}}(\cdot|\theta)$, parameterized by $\theta \in \Theta = \mathbb{R}^d$. The likelihood of $\mathbf{x}$ is then given by

$$P_{\text{likelihood}}(\mathbf{x}|\theta) = \prod_{i=1}^{N_{\text{data}}} P_{\text{elem}}(x_i|\theta).$$

In the Bayesian framework, a prior distribution P_{prior} is considered on the parameters. The aim is to sample from the posterior distribution

$$\pi(\theta|\mathbf{x}) \propto P_{\text{prior}}(\theta) P_{\text{likelihood}}(\mathbf{x}|\theta), \tag{2.2}$$

in order, for instance, to compute expectations with respect to this distribution. Sampling is usually done with MCMC methods. This however requires, at each iteration, to compute either the log-likelihood $\pi(\cdot|\mathbf{x})$ or its gradient

$$\nabla_\theta(\log \pi(\theta|\mathbf{x})) = \nabla_\theta(\log P_{\text{prior}}(\theta)) + \sum_{i=1}^{N_{\text{data}}} \nabla_\theta(\log P_{\text{elem}}(x_i|\theta)). \tag{2.3}$$

The cost of computing these quantities scales as $\mathcal{O}(N_{\text{data}})$, and is usually the computational bottleneck in implementations of MCMC algorithms. As discussed in the introduction, we focus here on stochastic gradient dynamics, which are discretizations of stochastic dynamics admitting $\pi(\cdot|\mathbf{x})$ as invariant probability measure, and where the cost of evaluating (2.3) is reduced by approximating the gradient through some estimator based on a mini-batch of the complete data set.

We first review in Section 2.2.1 elements on the error analysis of discretization of SDEs, with some emphasis on the bias induced on the invariant measure, as these results will be repeatedly used throughout this paper, and are key to understanding the performance and limitations of all the methods we discuss. We then recall in Section 2.2.2 the mini-batching

method [157] and focus on the case where the size of mini-batch is small, possibly limited to a single point (as done in stochastic approximation algorithms [126]). We then recall two classical SDEs upon which our method is based: overdamped Langevin dynamics in Section 2.2.3 (known as SGLD when using mini-batching), and then Langevin dynamics in Section 2.2.4. In each of these sections, we analyze how mini-batching affects the posterior distribution and quantify the bias incurred on it by studying the associated effective dynamics. These results are illustrated by numerical examples in Section 2.2.5.

2.2.1 Some elements on error analysis for discretizations of SDEs

Consider a SDE $(\theta_t)_{t\geqslant 0} \subset \Theta$ with generator $\mathcal{L}$ admitting $\pi(\cdot|\mathbf{x})$ as invariant probability measure: For any smooth function ϕ with compact support,

$$\int_\Theta \mathcal{L}\phi \, d\pi(\cdot|\mathbf{x}) = 0.$$

Denote by $(\theta^m)_{m\geqslant 0}$ a time discretization of the SDE with a fixed time step Δt (so that θ^m is an approximation of $\theta_{m\Delta t}$). We assume that the Markov chain corresponding to the time discretization of the SDE admits a unique invariant probability measure, denoted by $\pi_{\Delta t}$. This is for instance the case for Langevin-type dynamics when the drift of the dynamics is globally Lipschitz or when Lyapunov conditions are satisfied [101]. For a given observable ϕ, the target expectation

$$\mathbb{E}_\pi(\phi) = \int_\Theta \phi(\theta)\pi(\theta|\mathbf{x}) \, d\theta$$

is approximated by $\mathbb{E}_{\pi_{\Delta t}}(\phi)$, which is itself typically estimated by the trajectory average

$$\widehat{\phi}_{\Delta t, N_{\rm iter}} = \frac{1}{N_{\rm iter}} \sum_{m=1}^{N_{\rm iter}} \phi(\theta^m).$$

The total error on averages with respect to $\pi(\cdot|\mathbf{x})$ can then be written as:

$$\widehat{\phi}_{\Delta t, N_{\rm iter}} - \mathbb{E}_\pi(\phi) = (\mathbb{E}_{\pi_{\Delta t}}(\phi) - \mathbb{E}_\pi(\phi)) + \left(\widehat{\phi}_{\Delta t, N_{\rm iter}} - \mathbb{E}_{\pi_{\Delta t}}(\phi)\right).$$

The first term on the right hand side corresponds to the bias on the invariant probability measure resulting from taking finite step sizes. The second term in the error has two origins: (i) a bias coming from the initial distribution of θ^0 when this random variable is not distributed according to $\pi_{\Delta t}$; (ii) a statistical error, which is dictated by the central limit theorem for $N_{\rm iter}$ large.

We focus in this work on the bias on the invariant probability measure, which can be bounded using the weak order of the scheme, provided some ergodicity conditions are satisfied. Recall that a numerical scheme is of weak order s if for any smooth and compactly supported function ϕ and final time $T > 0$, there exists $C \in \mathbb{R}_+$ such that

$$\forall m \in \{1, \ldots, \lceil T/\Delta t \rceil\}, \qquad |\mathbb{E}[\phi(\theta_{m\Delta t})] - \mathbb{E}[\phi(\theta^m)]| \leqslant C\Delta t^s. \tag{2.4}$$

When this condition holds, and under appropriate ergodicity conditions (see [146] for a pioneering work, as well as [145, 101, 2, 87, 22] for subsequent works on Langevin-like dynamics), the following bound is obtained on the bias on the invariant probability measure of the numerical scheme: For any smooth and compactly supported function ϕ, there exists $\Delta t_\star > 0$ and L such that

$$\forall \Delta t \in (0, \Delta t_\star], \qquad |\mathbb{E}_{\pi_{\Delta t}}(\phi) - \mathbb{E}_\pi(\phi)| \leqslant L\Delta t^s. \tag{2.5}$$

In order to write a sufficient local consistency condition to obtain an estimate such as (2.4), we introduce the evolution operator $P_{\Delta t}$ associated with the numerical scheme at hand, defined as follows: For any smooth and compactly supported function ϕ,

$$(P_{\Delta t}\phi)(\theta) = \mathbb{E}\left[\phi\left(\theta^{m+1}\right) \,\middle|\, \theta^m = \theta\right].$$

Under appropriate technical conditions, including moment conditions on the iterates of the numerical scheme (see [107, Theorem 2.1] for a precise statement), a sufficient condition for (2.4) to hold is

$$P_{\Delta t} = \mathrm{e}^{\Delta t \mathcal{L}} + \mathcal{O}(\Delta t^{s+1}), \tag{2.6}$$

where $(\mathrm{e}^{t\mathcal{L}}\phi)(\theta) = \mathbb{E}[\phi(\theta_t) \,|\, \theta_0 = \theta]$ is the evolution operator associated with the underlying SDE. Here and in the sequel, the above equality has to be understood as follows: For any smooth and compactly supported function ϕ, there exist $\Delta t_\star > 0$ and $K \in \mathbb{R}_+$ such that, for any $\Delta t \in (0, \Delta t_\star]$, there is a function $R_{\phi,\Delta t}$ for which

$$P_{\Delta t}\phi = \mathrm{e}^{\Delta t \mathcal{L}}\phi + \Delta t^{s+1} R_{\phi,\Delta t}, \qquad \sup_{\Delta t \in (0,\Delta t_\star]} \sup_{\theta \in \Theta} |R_{\phi,\Delta t}(\theta)| \leqslant K;$$

see for instance [94, Section 3.3] for a more precise discussion of this point.

Let us conclude this section by introducing the concept of effective dynamics (also called modified SDEs in [139, 163]), a tool inspired by backward numerical analysis. The idea is to construct a continuous dynamics, parameterized by the time step, which better coincides with the numerical scheme than the reference dynamics. More precisely, given a numerical scheme of weak order s, characterized by the local consistency condition (2.6), we look for a modified SDE with generator $\mathcal{L}_{\mathrm{mod},\Delta t}$ such that

$$P_{\Delta t} - \mathrm{e}^{\Delta t \mathcal{L}_{\mathrm{mod},\Delta t}} = \mathcal{O}(\Delta t^{s+2}). \tag{2.7}$$

An alternative reformulation is the following: denoting by $(\widetilde{\theta})_{t \geqslant 0}$ the solution to the SDE with generator $\mathcal{L}_{\mathrm{mod},\Delta t}$, then

$$\mathbb{E}[\phi(\theta^1) \,|\, \theta_0 = \theta] = \mathbb{E}[\phi(\widetilde{\theta}_t) \,|\, \theta_0 = \theta] + \mathrm{O}(\Delta t^3),$$

while

$$\mathbb{E}[\phi(\theta^1) \,|\, \theta_0 = \theta] = \mathbb{E}[\phi(\theta_t) \,|\, \theta_0 = \theta] + \mathrm{O}(\Delta t^2).$$

The prime interest of effective dynamics in our work is to provide some interpretation of the behavior of numerical schemes. Following the standard philosophy of backward analysis, it is indeed possible to obtain information on the behavior of numerical schemes by studying the properties of the effective dynamics, in particular their invariant probability measures. This is also useful for instance to understand the impact of mini-batching on SGLD and Langevin dynamics (see respectively [154] and Section 2.2.4), and is also the foundation of Adaptive Langevin dynamics. In particular, when (2.7) holds with $\mathcal{L}_{\mathrm{mod},\Delta t} = \mathcal{L} + \Delta t^s \mathcal{A}_{\mathrm{mod}}$, then (2.5) can be rewritten more precisely in the form of a Talay–Tubaro expansion [146]:

$$\forall \Delta t \in (0, \Delta t_\star], \qquad |\mathbb{E}_{\pi_{\Delta t}}(\phi) - \mathbb{E}_\pi(\phi) - \Delta t^s \mathbb{E}_\pi(f\phi)| \leqslant \widetilde{L} \Delta t^{s+1}, \tag{2.8}$$

with $f = (-\mathcal{L}^*)^{-1} \mathcal{A}^*_{\mathrm{mod}} \mathbf{1}$, adjoints being taken on $L^2(\pi)$ (provided the inverse of $\mathcal{L}$ can be defined on an appropriate subspace of $L^2(\pi)$, see again [94, Section 3.3]).

2.2.2 Mini-Batching procedure

The idea of using mini-batching in the context of SDEs has first been introduced through SGLD in [157] to reduce the calculation time of one step of the discretization of the over-damped Langevin dynamics (see (2.16) below) from $\mathcal{O}(N_{\mathrm{data}})$ to a fraction of this cost. It is

inspired from the classical stochastic gradient descent method [126] in the sense that a consistent approximation of $\nabla_\theta(\log \pi(\cdot|\mathbf{x}))$ is used in the discretization of overdamped Langevin dynamics; see Section 2.2.3 below.

Mini-batching relies on computing at each iteration an approximation of the gradient of the log-likelihood using a random subset of size n of the data points, which reduces the cost from $\mathcal{O}(N_{\text{data}})$ to $\mathcal{O}(n)$. More precisely, the gradient of the posterior distribution is approximated by

$$\widehat{F_n}(\theta) = \nabla_\theta(\log P_{\text{prior}}(\theta)) + \frac{N_{\text{data}}}{n} \sum_{i \in I_n} \nabla_\theta(\log P_{\text{elem}}(x_i|\theta)), \tag{2.9}$$

where I_n is a random subset of size n generated by sampling uniformly indices from $\{1, ..., N_{\text{data}}\}$, with or without replacement. Sampling with or without replacement produces similar results when $n \ll N_{\text{data}}$ with N_{data} large enough. Sampling with replacement however leads to estimators (2.9) with larger variances for a given value of n (as quantified by (2.12) below), and should therefore be avoided when larger batches are considered. In any case, it is easily shown that $\widehat{F_n}(\theta)$ in (2.9) is a consistent approximation of $\nabla_\theta(\log \pi(\cdot|\mathbf{x}))$ given by (2.3).

Properties of $\widehat{F_n}(\theta)$. In view of (2.9), the covariance matrix of the stochastic gradient, which is nonnegative and symmetric, is given by:

$$\text{cov}\left(\widehat{F_n}(\theta)\right) = \varepsilon(n)\Sigma_{\mathbf{x}}(\theta), \tag{2.10}$$

with $\Sigma_{\mathbf{x}}(\theta)$ the empirical covariance of the gradient estimator for $n = 1$ (*i.e.* with expectations computed with respect to the random variable $\mathcal{I}$ uniformly distributed in $\{1, ..., N_{\text{data}}\}$):

$$\begin{aligned}\Sigma_{\mathbf{x}}(\theta) &= \text{cov}_{\mathcal{I}}\left[\nabla_\theta(\log P_{\text{elem}}(x_{\mathcal{I}}|\theta))\right] \\ &= \frac{1}{N_{\text{data}} - 1} \sum_{i=1}^{N_{\text{data}}} \left[\nabla_\theta(\log P_{\text{elem}}(x_i|\theta)) - \mathscr{F}_{\mathbf{x}}(\theta)\right] \left[\nabla_\theta(\log P_{\text{elem}}(x_i|\theta)) - \mathscr{F}_{\mathbf{x}}(\theta)\right]^T,\end{aligned} \tag{2.11}$$

where uv^T is the matrix $(u_i v_j)_{1 \leqslant i,j \leqslant d} \in \mathbb{R}^{d \times d}$ for $u, v \in \mathbb{R}^d$, the average force reads

$$\mathscr{F}_{\mathbf{x}}(\theta) = \frac{1}{N_{\text{data}}} \sum_{i=1}^{N_{\text{data}}} \nabla_\theta(\log P_{\text{elem}}(x_i|\theta)),$$

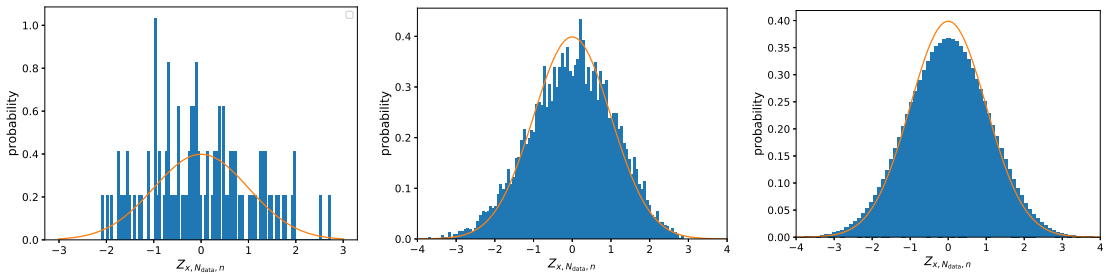
and

$$\varepsilon(n) = \begin{cases} \dfrac{N_{\text{data}}(N_{\text{data}} - 1)}{n}, & \text{for sampling with replacement,} \\ \dfrac{N_{\text{data}}(N_{\text{data}} - n)}{n}, & \text{for sampling without replacement.} \end{cases} \tag{2.12}$$

We refer for instance to [30] for more details. We will see in the next sections that, for all the methods we consider, the bias due to mini-batching is controlled by $\varepsilon(n)$ whether sampling is performed with our without replacement. It is really this parameter (in fact, $\varepsilon(n)\Delta t$) which needs to be small in order for the asymptotic analysis on the bias to be correct. Note that $\varepsilon(n) \geqslant N_{\text{data}} - 1$ for sampling with replacement, which means that a bias is necessarily observed in this case. Values of $\varepsilon(n)$ of order N_{data} are obtained only when the mini-batch size n is a fraction of the total number of data points N_{data}. To obtain values of order 1, sampling has to be done without replacement with n close to N_{data}. In contrast, $\varepsilon(n)$ is of order N_{data}^2 when the mini-batch size is small, whether sampling is performed with or without replacement.

The above definitions allow to rewrite (2.9) as

$$\widehat{F_n}(\theta) = \nabla_\theta(\log \pi(\theta|\mathbf{x})) + \sqrt{\varepsilon(n)}\Sigma_{\mathbf{x}}^{\frac{1}{2}}(\theta) Z_{\mathbf{x}, N_{\text{data}}, n}, \tag{2.13}$$



(a) $n = 1$ (b) $n = 2$ (c) $n = 30$

Figure 2.1 – Histogram of $Z_{\mathbf{x},N_{\text{data}},n}$ when sampling is performed without replacement for the Gaussian model of Section 2.2.5.1, with $N_{\text{data}} = 100$, $\sigma_\theta = 1$, $\sigma_x = 1$, $\mu = 0$, $\theta = 0.5$. The reference standard Gaussian distribution is superimposed as a continuous line.

where $Z_{\mathbf{x},N_{\text{data}},n}$ is by construction a centered random variable with identity covariance. If n and N_{data} are large enough, and $n \ll N_{\text{data}}$, the central limit theorem holds so that $Z_{\mathbf{x},N_{\text{data}},n}$ asymptotically follows a centered reduced normal distribution (as assumed for instance in [31, 4, 162]). Most of the works in the literature consider the regime where the central limit theorem holds. This is however unnecessary when it comes to quantifying the weak error of the numerical scheme and hence the error on the invariant measure. In our work, we consider explicitly the case when n is of order 1 and N_{data} is (moderately) large, so that $Z_{\mathbf{x},N_{\text{data}},n}$ is not necessarily close to a Gaussian distribution. This is illustrated in Figures 2.1 and 2.2, where we plot the distribution of $Z_{\mathbf{x},N_{\text{data}},n}$ for various values of n for the models introduced in Section 2.2.5, where the elementary likelihood is either a Gaussian or a mixture of Gaussians. It is clear that the distribution of $Z_{\mathbf{x},N_{\text{data}},n}$ is far from being Gaussian for small values of n, but is nonetheless centered and with covariance I_d. Distributions close to Gaussian are obtained for $n = 20 - 30$ for the situation considered in Figure 2.2.

2.2.3 Stochastic Gradient Langevin Dynamics

We introduce in this section the SGLD algorithm (see Section 2.2.3.1) and analyze the bias due to replacing the gradient (2.3) by the stochastic estimator (2.9) using the effective dynamics associated with the numerical method at hand (see Section 2.2.3.2).

2.2.3.1 Description of the method

A popular SDE to sample from $\pi(\cdot|\mathbf{x})$ is the overdamped Langevin dynamics:

$$d\theta_t = \nabla_\theta(\log \pi(\theta_t|\mathbf{x}))\, dt + \sqrt{2}\, dW_t, \tag{2.14}$$

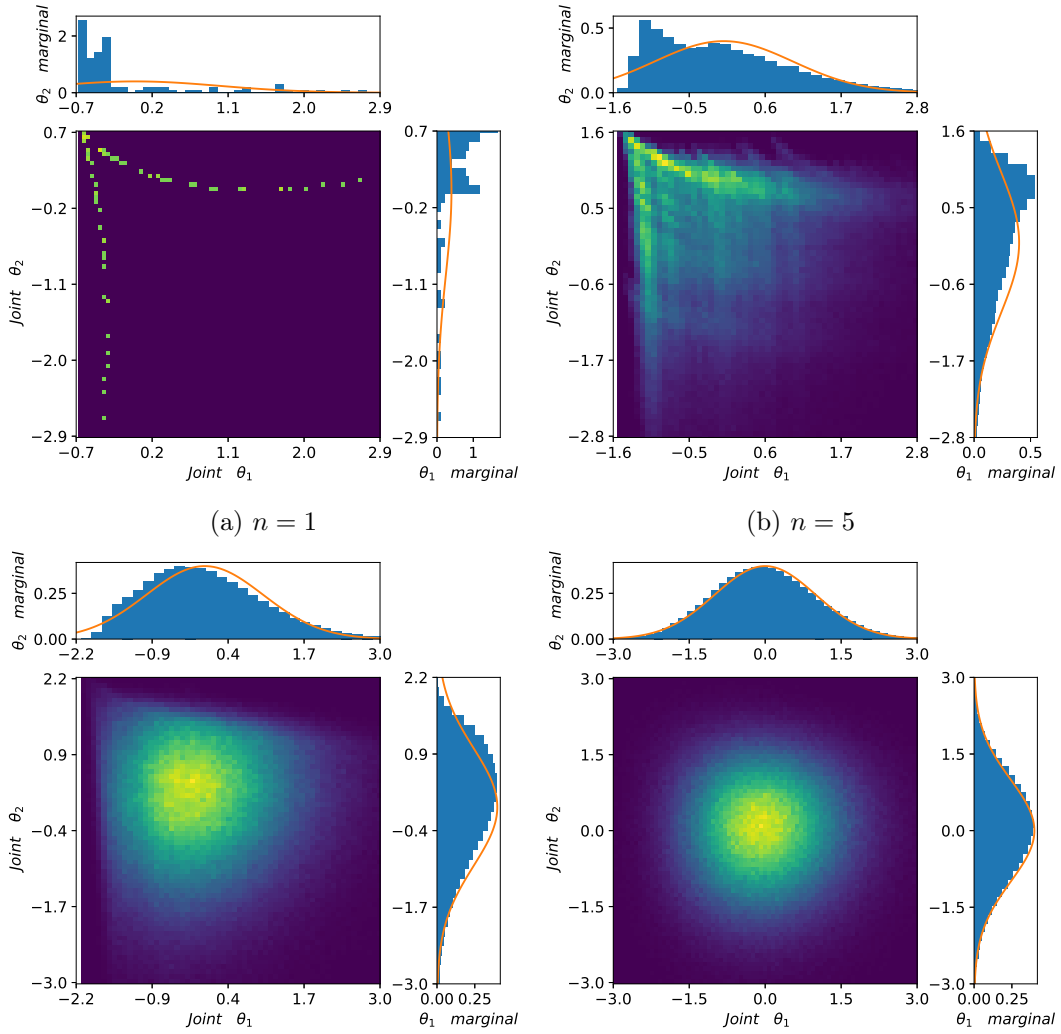
where W_t is a standard d-dimensional Wiener process. Its generator is given by

$$\mathcal{L}_{\text{ovd}} = \nabla_\theta(\log \pi(\cdot|\mathbf{x}))^T \nabla_\theta + \Delta_\theta. \tag{2.15}$$

It is well known that the process (2.14) is irreducible and admits $\pi(\cdot|\mathbf{x})$ as a unique invariant probability measure, so that expectations can be approximated by trajectory averages (see for instance [73]). In general, one cannot directly simulate the overdamped Langevin dynamics (2.14). To numerically approximate the solution, the widely used Euler-Maruyama scheme can be considered:

$$\theta^{m+1} = \theta^m + \Delta t \nabla_\theta(\log \pi(\theta^m|\mathbf{x})) + \sqrt{2\Delta t}\, G^m, \tag{2.16}$$

where $\Delta t > 0$ is the step size, and $(G^m)_{m\geqslant 0}$ is a vector of i.i.d. standard d-dimensional Gaussian random variables. It can be shown that the bias $\mathbb{E}_{\pi_{\Delta t}}(\phi) - \mathbb{E}_\pi(\phi)$ for overdamped



(a) $n = 1$

(b) $n = 5$

(c) $n = 10$

(d) $n = 30$

Figure 2.2 – Histogram of $Z_{\mathbf{x},N_{\text{data}},n}$ for $\theta = (1, 0.2)$, when sampling is performed without replacement for the model of Section 2.2.5.2 with $N_{\text{data}} = 100$ data points sampled from a mixture of Gaussians with parameters $\sigma_1 = \sigma_2 = 0.4$, $w = 0.5$, $\theta_1 = 0.2$ and $\theta_2 = 1$. The reference standard Gaussian distribution is superimposed as a continuous line on the marginal distributions in θ_1, θ_2.

Langevin dynamics is of order $\mathcal{O}(\Delta t)$; see for instance [144] and [101, Theorem 7.3] for pioneering works on ergodic properties of discretization of SDEs and error estimation for globally Lipschitz vector fields.

Using the stochastic estimator (2.9), SGLD corresponds to the following numerical scheme:

$$\theta^{m+1} = \theta^m + \Delta t \widehat{F_n}(\theta^m) + \sqrt{2\Delta t}\, G^m. \tag{2.17}$$

We assume in the sequel that this Markov chain admits a unique invariant probability measure $\pi_{\Delta t,n}$ (we omit the dependence on $\mathbf{x}$ to simplify the notation).

Remark 2.1. *It is suggested in [157] to use decreasing time steps as this allows to ultimately eliminate the bias without resorting to a Metropolis Hastings scheme. However, considering decreasing time steps means that we need more iterations for a fixed final time T, which increases the computational cost of the algorithm and results in a possibly large variance in the estimation of averages at fixed computational cost. In practice, it is more customary to use a small finite time step. As discussed in [154], the bias arising from mini-batching dominates the one resulting from the time step discretization. Our focus in this work is therefore on the bias arising from mini-batching and not on the bias coming from the use of finite time steps.*

2.2.3.2 Effective SGLD

We recall in this section how to prove that the bias between the invariant measure $\pi_{\Delta t,n}$ of SGLD (2.17) and $\pi(\cdot|\mathbf{x})$ is of order $\mathcal{O}((1+\varepsilon(n))\Delta t)$ – by which we mean that error estimates such as (2.5) hold with Δt^s replaced by $(1+\varepsilon(n))\Delta t$ on the right hand side. Such error estimates were already obtained in [154]. This analysis is important since we will repeatedly rely on it to quantify the bias on the invariant probability measures of numerical schemes, in particular for Adaptive Langevin dynamics and their extensions, which is why we insist on presenting the overall strategy.

From a technical viewpoint, the analysis relies on effective dynamics, discussed at the end of Section 2.2.1. To derive an effective dynamics for the SGLD, we follow the general framework of backward error analysis for SDEs considered in [139, 163, 37, 1], which is based on building a perturbation to the generator associated with the original dynamics (2.15), constructed so that SGLD evolved over one step is closer to the SDE associated with the modified generator than to the original dynamics. The results of [154] show that (see also Appendix 2.A.a for a short proof allowing to make precise the dependence of the remainder term on n)

$$\widehat{P}_{\Delta t,n} = \mathrm{e}^{\Delta t(\mathcal{L}_{\rm ovd}+\Delta t\mathcal{A}_{{\rm ovd},n})} + \mathcal{O}\left((1+\varepsilon(n)^{3/2})\Delta t^3\right), \qquad \mathcal{A}_{{\rm ovd},n} = \varepsilon(n)\mathcal{A}_{\rm mb} + \mathcal{A}_{\rm disc}, \tag{2.18}$$

where $\widehat{P}_{\Delta t,n}$ is the evolution operator associated with the SGLD scheme (2.17), and the operators $\mathcal{A}_{\rm mb}$ and $\mathcal{A}_{\rm disc}$ respectively encode the perturbations arising from mini-batching and time discretization:

$$\mathcal{A}_{\rm mb}\phi = \frac{1}{2}\Sigma_{\mathbf{x}} : \nabla_\theta^2\phi,$$

$$\mathcal{A}_{\rm disc}\phi = -\nabla_\theta^2(\log\pi(\cdot|\mathbf{x})) : \nabla_\theta^2\phi - \frac{1}{2}\nabla_\theta\Delta(\log\pi(\cdot|\mathbf{x}))^T\nabla_\theta\phi - \frac{1}{2}\nabla_\theta(\log\pi(\cdot|\mathbf{x}))^T\nabla_\theta^2(\log\pi(\cdot|\mathbf{x}))\nabla_\theta\phi.$$

In the latter expressions, we denote by : the Frobenius inner product of two square matrices, *i.e.*

$$\forall M^1, M^2 \in \mathbb{R}^{d\times d}, \qquad M^1 : M^2 = \sum_{i,j=1}^d M^1_{i,j}M^2_{i,j}.$$

In fact, as discussed in Remark 2.15 of Appendix 2.A.a, the error estimate in (2.18) can be improved to $\mathcal{O}((1+\varepsilon(n))\Delta t^3)$ when $\mathbb{E}[Z^3_{\mathbf{x},N_{\rm data},n}] = 0$. Using [94, Remark 5.5], or results

from [154], we deduce that the bias between $\pi_{\Delta t,n}$ and π is of order $\mathcal{O}\left((1+\varepsilon(n))\Delta t\right)$ provided that Δt and $\varepsilon(n)\Delta t$ are small enough. Recalling (2.8) and the expression of $\mathcal{A}_{\mathrm{mb}}$, one can see that the bias is indeed larger when $\Sigma_{\mathbf{x}}$ is larger, although this statement is not as clear cut as for Langevin dynamics since the correction function f which appears in (2.8) for SGLD also involves derivatives of $\Sigma_{\mathbf{x}}$, in contrast to estimates obtained for underdamped and adaptive Langevin dynamics.

Note that the bias can be decreased by either decreasing Δt or increasing n, hence decreasing $\varepsilon(n)$ in view of (2.12). However, as already discussed in [154], the mini-batching error usually dominates the discretization error by orders of magnitude since $\varepsilon(n)$ is proportional to N_{data}^2 unless n is a fraction of N_{data}, in which case it scales as N_{data}.

Remark 2.2. *To reduce the mini-batching bias, it is suggested in [154] to renormalize the magnitude of the injected noise by a quantity involving $\Sigma_{\mathbf{x}}(\theta)$ (the so-called modified SGLD scheme). Since $\Sigma_{\mathbf{x}}(\theta)$ is usually unknown, this requires estimating this matrix, which is however computationally expensive since the estimation has to be repeated for each new value of θ, and may cancel the gain provided by mini-batching in the first place. Our focus in this work is to reduce the mini-batching error without the need to estimate $\Sigma_{\mathbf{x}}(\theta)$.*

2.2.4 Langevin dynamics with mini-batching

It has been observed in practice that a better sampling of probability measures can be provided by Langevin dynamics, both in the literature on computational statistical physics (see for instance [26]) and more recently in the machine learning literature [35]. We first present in Section 2.2.4.1 the numerical scheme obtained by discretizing Langevin dynamics with a splitting scheme and replacing the gradient of the log-likelihood by its estimator (2.9), and then analyze the bias induced on the invariant probability measure of the numerical scheme in Section 2.2.4.2.

2.2.4.1 Standard Langevin dynamics

Langevin dynamics introduces some inertia in the evolution of θ, through an extended configuration space with a momentum vector p conjugated to θ. It can be seen as a perturbation of the Hamiltonian dynamics where some fluctuation/dissipation mechanism is added to the evolution of the momenta, and reads

$$\begin{cases} d\theta_t = p_t\, dt, \\ dp_t = \nabla_\theta(\log \pi(\theta_t|\mathbf{x}))\, dt - \Gamma p_t\, dt + \sqrt{2}\Gamma^{1/2}\, dW_t, \end{cases} \tag{2.19}$$

where $\Gamma \in \mathbb{R}^{d\times d}$ is a positive definite symmetric matrix. It would be possible, as in molecular dynamics, to attach a mass to each degree of freedom, but we set here for simplicity this mass to 1, the generalization to non trivial mass matrices being straightforward. The generator of Langevin dynamics (2.19) is given by

$$\mathcal{L}_{\mathrm{lan}} = \nabla_\theta(\log \pi(\cdot|\mathbf{x}))^T \nabla_p + p^T \nabla_\theta - p^T \Gamma \nabla_p + \Gamma : \nabla_p^2. \tag{2.20}$$

The diffusion constant $\sqrt{2}\Gamma^{1/2}$ in front of the Wiener process ensures that the following probability distribution is invariant:

$$\mu(d\theta\, dp|\mathbf{x}) = \pi(\theta|\mathbf{x})\tau(dp)\, d\theta,$$

where

$$\tau(dp) = (2\pi)^{-d/2} \mathrm{e}^{-p^2/2}\, dp, \tag{2.21}$$

see for instance [118, 86, 94]. In particular, the marginal distribution of $\mu(\cdot|\mathbf{x})$ in the θ variable is indeed the target distribution (2.2). It can even be shown that time averages along

solutions of (2.19) almost surely converge to averages with respect to μ since the generator of the dynamics is hypoelliptic [73].

To numerically approximate the solution of Langevin dynamics, we use a numerical integrator based on a second order Strang splitting, as encoded by the following evolution operator (although there are various other choices of orderings, see for instance [85, 87, 86]):

$$P_{\Delta t} = \mathrm{e}^{\Delta t \mathcal{L}_3/2} \mathrm{e}^{\Delta t \mathcal{L}_2/2} \mathrm{e}^{\Delta t \mathcal{L}_1} \mathrm{e}^{\Delta t \mathcal{L}_2/2} \mathrm{e}^{\Delta t \mathcal{L}_3/2}, \tag{2.22}$$

where

$$\mathcal{L}_1 = \nabla_\theta \log(\pi(\cdot|\mathbf{x}))^T \nabla_p, \qquad \mathcal{L}_2 = p^T \nabla_\theta, \qquad \mathcal{L}_3 = -p^T \Gamma \nabla_p + \Gamma : \nabla^2. \tag{2.23}$$

The elementary generators $\mathcal{L}_1$ and $\mathcal{L}_2$ are respectively associated with the elementary differential equations $d\theta_t = p_t\, dt$ and $dp_t = \nabla_\theta \log(\pi(\theta_t|\mathbf{x})\, dt$, and can be analytically integrated. The elementary generator $\mathcal{L}_3$ is associated with the Ornstein–Uhlenbeck process $dp_t = -\Gamma p_t\, dt + \sqrt{2}\Gamma^{1/2}\, dW_t$, and can also be analytically integrated as:

$$p_t = \mathrm{e}^{-\Gamma t} p_0 + \sqrt{2} \int_0^t \mathrm{e}^{-(t-s)\Gamma} \Gamma^{1/2}\, dW_s \sim \mathcal{N}(\alpha_t p_0, \mathrm{I}_d - \alpha_{2t}), \qquad \alpha_t = \mathrm{e}^{-\Gamma t}. \tag{2.24}$$

Finally, the numerical scheme encoded by (2.22) reads:

$$\begin{cases} p^{m+\frac{1}{3}} = \alpha_{\Delta t/2}\, p^m + (\mathrm{I}_d - \alpha_{\Delta t})^{1/2}\, G^m, \\ \theta^{m+\frac{1}{2}} = \theta^m + \dfrac{\Delta t}{2}\, p^{m+\frac{1}{3}}, \\ p^{m+\frac{2}{3}} = p^{m+\frac{1}{3}} + \Delta t \nabla_\theta \left[\log \pi \left(\theta^{m+\frac{1}{2}} \middle| \mathbf{x} \right) \right], \\ \theta^{m+1} = \theta^{m+\frac{1}{2}} + \dfrac{\Delta t}{2} p^{m+\frac{2}{3}}, \\ p^{m+1} = \alpha_{\Delta t/2}\, p^{m+\frac{2}{3}} + (\mathrm{I}_d - \alpha_{\Delta t})^{1/2}\, G^{m+\frac{1}{2}}, \end{cases} \tag{2.25}$$

where $(G^m)_{m \geqslant 0}$ and $(G^{m+\frac{1}{2}})_{m \geqslant 0}$ are two independent families of i.i.d. standard d-dimensional Gaussian random variables. Moreover, it is proved in [87, 2, 44] that the Markov chain generated by (2.25) admits a unique invariant probability measure $\mu_{\Delta t}$ and that there exists $C \in \mathbb{R}+$ such that, for any smooth function ϕ with compact support,

$$\left| \int_\Theta \phi(\theta, p) \mu_{\Delta t}(d\theta\, dp) - \int_\Theta \phi(\theta, p) \mu(d\theta\, dp|\mathbf{x}) \right| \leqslant C \Delta t^2.$$

The key element to prove this statement is the fact that the numerical scheme (2.25) is an approximation of weak order 2 of Langevin dynamics (see [87, 2])

$$P_{\Delta t} = \mathrm{e}^{\Delta t \mathcal{L}_{\mathrm{lan}}} + \mathcal{O}(\Delta t^3). \tag{2.26}$$

2.2.4.2 Error estimates for Langevin dynamics with mini-batching

In order to analyze the error on the posterior measure sampled by a discretization of Langevin dynamics used in conjunction with mini-batching, we derive here the effective dynamics associated with the numerical method (2.25) when the gradient of the log-likelihood is replaced by its stochastic estimator (2.9) (similar results are obtained for other Strang splittings). This

corresponds to the following numerical scheme:

$$\begin{cases} p^{m+\frac{1}{3}} = \alpha_{\Delta t/2}\, p^m + (\mathrm{I}_d - \alpha_{\Delta t})^{1/2}\, G^m, \\ \theta^{m+\frac{1}{2}} = \theta^m + \dfrac{\Delta t}{2}\, p^{m+\frac{1}{3}}, \\ p^{m+\frac{2}{3}} = p^{m+\frac{1}{3}} + \Delta t \widehat{F_n}\left(\theta^{m+\frac{1}{2}}\right), \\ \theta^{m+1} = \theta^{m+\frac{1}{2}} + \dfrac{\Delta t}{2} p^{m+\frac{2}{3}}, \\ p^{m+1} = \alpha_{\Delta t/2}\, p^{m+\frac{2}{3}} + (\mathrm{I}_d - \alpha_{\Delta t})^{1/2}\, G^{m+\frac{1}{2}}. \end{cases} \tag{2.27}$$

When using mini-batching, we are in fact replacing $\mathrm{e}^{\Delta t \mathcal{L}_1}$ in (2.22) by the elementary evolution operator $Q_{\Delta t}^{\mathcal{L}_1}$ acting as

$$\left(Q_{\Delta t}^{\mathcal{L}_1}\varphi\right)(\theta, p) = \mathbb{E}\left[\varphi\left(\theta, p + \Delta t \widehat{F_n}(\theta)\right)\right].$$

A simple computation provided in Appendix 2.A.b shows that

$$Q_{\Delta t}^{\mathcal{L}_1} = \mathrm{e}^{\Delta t \mathcal{L}_1} + \mathcal{O}(\varepsilon(n)\Delta t^2). \tag{2.28}$$

Denoting by $\widehat{P}_{\Delta t,n}$ the evolution operator of the numerical scheme (2.27), and by $\mu_{\Delta t,n}$ its invariant probability measure (assuming that it exists and it is unique), it can then be proved that (see Appendix 2.A.b)

$$\widehat{P}_{\Delta t,n} = \mathrm{e}^{\Delta t \mathcal{L}_{\mathrm{lan}}} + \mathcal{O}\left((\varepsilon(n) + \Delta t)\Delta t^2\right). \tag{2.29}$$

In view of this equality, the bias between μ and $\mu_{\Delta t,n}$ is of order $\mathcal{O}\left((\varepsilon(n) + \Delta t)\Delta t\right)$ (by which we mean that error estimates such as (2.5) hold with $(\varepsilon(n) + \Delta t)\Delta t$ on the right hand side instead of Δt^s). It is clear that the error $\varepsilon(n)\Delta t$ coming from mini-batching dominates the error Δt^2 due to time discretization since $\varepsilon(n)$ is much larger than Δt unless n is very close to N_{data}. Comparing this equality to (2.26) highlights the fact that mini-batching degrades the consistency estimate (2.26) by one order in Δt with respect to Langevin dynamics (2.19).

Remark 2.3. *To remove the bias on the invariant probability measure at dominant order in $\varepsilon(n)\Delta t$, it is suggested in [100] to modify the integration of the Ornstein–Uhlenbeck part on the momenta. This requires however an estimation of the covariance matrix, which can again be computationally prohibitive (as discussed in Remark 2.2).*

2.2.4.3 Effective dynamics for Langevin dynamics with mini-batching

We now construct an effective dynamics which coincides at order 3 in Δt over one time step with the numerical scheme (2.27) even when using mini-batching, in order to obtain an equality similar to (2.26). This effective dynamics is the key building block to understand Adaptive Langevin dynamics in Section 2.3.1. A straightforward computation, following the lines of [100], shows that (see Appendix 2.A.b)

$$\widehat{P}_{\Delta t,n} = \mathrm{e}^{\Delta t(\mathcal{L}_{\mathrm{lan}} + \Delta t \varepsilon(n)\mathcal{A}_{\mathrm{lan}})} + \mathcal{O}\left((1 + \varepsilon(n)^{3/2})\Delta t^3\right), \qquad \mathcal{A}_{\mathrm{lan}} = \frac{1}{2}\Sigma_{\mathbf{x}} : \nabla_p^2. \tag{2.30}$$

The effective dynamics associated with $\mathcal{L}_{\mathrm{lan}} + \Delta t \varepsilon(n)\mathcal{A}$ is

$$\begin{cases} d\widetilde{\theta}_t = \widetilde{p}_t\, dt, \\ d\widetilde{p}_t = \nabla_\theta \left[\log \pi\left(\widetilde{\theta}_t \middle| \mathbf{x}\right)\right] dt - \Gamma \widetilde{p}_t\, dt + \left(2\Gamma + \varepsilon(n)\Delta t \Sigma_{\mathbf{x}}(\widetilde{\theta}_t)\right)^{1/2} dW_t, \end{cases} \tag{2.31}$$

where $\Gamma+\varepsilon(n)\Delta t\Sigma_{\mathbf{x}}(\theta)$ is a positive definite matrix since $\Sigma_{\mathbf{x}}$ is positive (even though the latter matrix is unknown). We deduce from (2.8) that, thanks to estimates on $\mathcal{L}_{\rm lan}$ obtained from hypocoercive estimates [61, 41, 42], the bias on the invariant probability measure sampled by the numerical scheme is, at dominant order, of order $\varepsilon(n)\Delta t\|\mathcal{A}_{\rm lan}^*\mathbf{1}\|_{L^2(\mu)}$, which, by a simple computation, is itself bounded up to a multiplicative factor by $\varepsilon(n)\Delta t\|\Sigma_{\mathbf{x}}\|_{L^2(\pi)}$, in accordance with the general estimate (2.1) (choosing $\mathscr{S}=\{0\}$ hence $S^*=0$). Here and in the sequel, we assume implicitly that $\Sigma_{\mathbf{x}}$ has all its entries in $L^2(\pi)$.

Remark 2.4. *Consider the simple case when the covariance of the gradient estimator is contant, namely $\Sigma_{\mathbf{x}}=\sigma^2\mathrm{I}_d\in\mathbb{R}^{d\times d}$, and the friction is isotropic, namely $\Gamma=\gamma\mathrm{I}_d\in\mathbb{R}^{d\times d}$, with $\sigma,\gamma>0$. In this situation, the modified Langevin dynamics (2.31) samples the invariant probability measure proportional to $\mu^{\beta_{\rm eff}}$, with*

$$\beta_{\rm eff}=\left(1+\frac{\varepsilon(n)\sigma^2\Delta t}{2\gamma}\right)^{-1}<1. \tag{2.32}$$

2.2.5 Numerical illustration

We illustrate in this section the results on the bias of the posterior measure introduced by mini-batching when the elementary likelihoods are given by either a Gaussian or a mixture of Gaussians. The aim is to numerically quantify the bias in the non asymptotic regime $n=\mathcal{O}(1)$ (for which $Z_{\mathbf{x},N_{\rm data},n}$ is not Gaussian), and to have a benchmark on the bias to compare with AdL and its extension.

2.2.5.1 Gaussian posterior

We first suppose that the elements of the data set are normally distributed, namely $x_i|\theta\sim\mathcal{N}(\theta,\sigma_x^2)$, where θ is the parameter to estimate. The prior distribution on θ is a centered normal distribution with variance σ_θ^2. In this case, simple computations show that the posterior distribution on θ is also Gaussian with mean $\mu_{\rm post}$ and variance $\sigma_{\rm post}^2$, where (see for instance [154])

$$\mu_{\rm post}=\left(\frac{\sigma_x^2}{\sigma_\theta^2}+N_{\rm data}\right)^{-1}\sum_{i=1}^{N_{\rm data}}x_i,\qquad \sigma_{\rm post}^2=\left(\frac{1}{\sigma_\theta^2}+\frac{N_{\rm data}}{\sigma_x^2}\right)^{-1}. \tag{2.33}$$

Moreover,

$$\widehat{F_n}(\theta)=-\frac{\theta}{\sigma_\theta^2}+\frac{N_{\rm data}}{n}\sum_{i\in I_n}\frac{x_i-\theta}{\sigma_x^2}.$$

The variance of $\widehat{F_n}(\theta)$ as a function of θ can therefore be analytically computed using (2.11) (see again [154] for instance):

$$\Sigma_{\mathbf{x}}(\theta)=\mathrm{var}_{\mathcal{I}}\left[\nabla_\theta(\log P_{\rm elem}(x_{\mathcal{I}}|\theta))\right]=\frac{\mathrm{var}(\mathbf{x})}{\sigma_x^4}, \tag{2.34}$$

where $\mathcal{I}$ is a random variable uniformly distributed in $\{1,...,N_{\rm data}\}$, and

$$\mathrm{var}(\mathbf{x})=\frac{1}{N_{\rm data}-1}\sum_{i=1}^{N_{\rm data}}\left(x_i-\frac{1}{N_{\rm data}}\sum_{j=1}^{N_{\rm data}}x_j\right)^2$$

is the empirical variance of the data. In this case, the covariance of the force $\Sigma_{\mathbf{x}}$ is constant and does not depend on the parameter θ.

Let us first discuss SGLD. As shown in [154, Section 2.1], there is no bias on the mean of the posterior distribution $\pi_{\Delta t,n}$ of the Markov chain associated with SGLD. The latter distribution

is however not Gaussian, because the random variable $Z_{\mathbf{x},N_{\text{data}},n}$ is not Gaussian. In order to characterize the shape of this distribution at dominant order, we write the effective dynamics with generator $\mathcal{L}_{\text{ovd}} + \Delta t \mathcal{A}_{\text{ovd},n}$ in (2.18), which turns out to be an Ornstein–Uhlenbeck process:

$$d\theta_t = \left[-\left(1 + \frac{\Delta t}{2\sigma_{\text{post}}^2}\right) \frac{\theta - \mu_{\text{post}}}{\sigma_{\text{post}}^2} \right] dt + \sqrt{2} \left(1 + \frac{\Delta t \varepsilon(n)}{2\sigma_x^4} \text{var}(\mathbf{x}) + \frac{\Delta t}{\sigma_{\text{post}}^2} \right)^{1/2} dW_t.$$

The probability measure of the latter process is a Gaussian distribution with mean 0 and variance

$$\sigma_{\text{post}}^2 \left(1 + \frac{\Delta t}{2\sigma_{\text{post}}^2}\right)^{-1} \left(1 + \frac{\varepsilon(n)\Delta t}{2\sigma_x^4} \text{var}(\mathbf{x}) + \frac{\Delta t}{\sigma_{\text{post}}^2} \right).$$

This shows that the invariant probability measure of SGLD is, at dominant order in Δt and $\varepsilon(n)\Delta t$, a Gaussian distribution with mean 0 and variance

$$\sigma_{\text{post}}^2 \left[1 + \frac{\Delta t}{2} \left(\frac{\varepsilon(n)}{\sigma_x^4} \text{var}(\mathbf{x}) + \frac{1}{\sigma_{\text{post}}^2} \right) \right] + \mathcal{O}\left((1 + \varepsilon(n)^2) \Delta t^2 \right). \tag{2.35}$$

For Langevin dynamics with mini-batching, there is (as for SGLD) no bias on the mean of the posterior distribution $\pi_{\Delta t,n}$, as proved in Appendix 2.B. The latter distribution is however not Gaussian when the random variable $Z_{\mathbf{x},N_{\text{data}},n}$ is not Gaussian. It is however close to a Gaussian distribution since the marginal posterior distribution in the θ variable obtained for the effective Langevin dynamics is, according to Remark 2.4, a Gaussian distribution with mean μ_{post} and variance $\sigma_{\text{post}}^2/\beta_{\text{eff}}$. This means that the variance for the discretization of Langevin dynamics (2.27) is, at dominant order in Δt and $\varepsilon(n)\Delta t$, and when $\Gamma = \gamma \text{I}_d$,

$$\sigma_{\text{post}}^2 \left(1 + \frac{\varepsilon(n)\Delta t}{2\gamma\sigma_x^4} \text{var}(\mathbf{x}) \right) + \mathcal{O}\left((1 + \varepsilon(n)^2) \Delta t^2 \right). \tag{2.36}$$

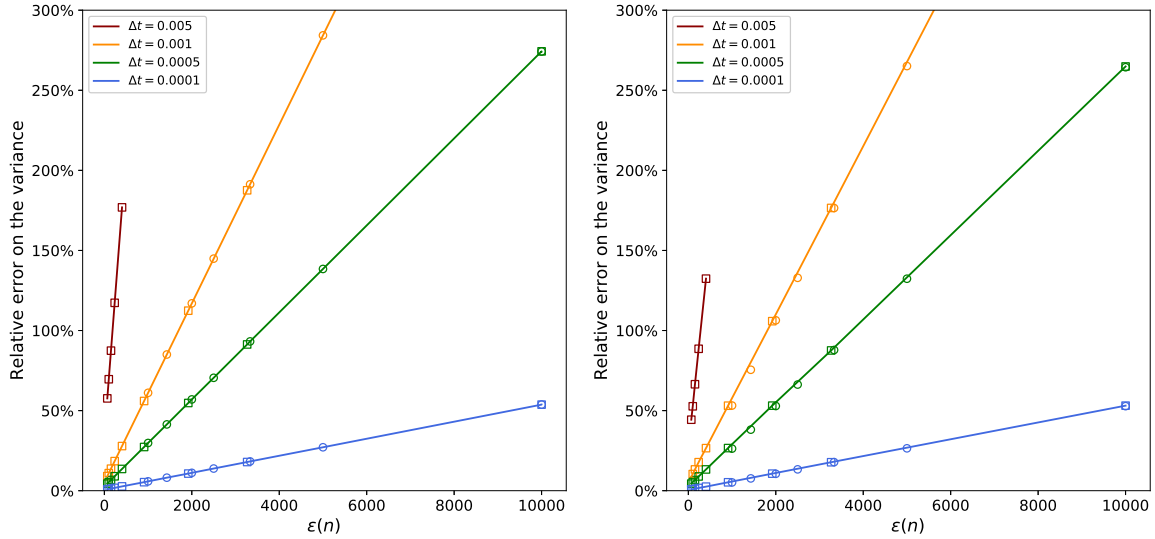
Note that this expression coincides with (2.35) when $\gamma = 1$ and $\varepsilon(n) \gg 1$.

To perform the numerical experiments, we generate a dataset of $N_{\text{data}} = 100$ according to a Gaussian distribution with mean $\theta_0 = 0$ and variance $\sigma_x = 1$. We also set $\sigma_\theta = 1$ in the prior distribution. We run the SGLD scheme (2.17) and Langevin dynamics with the numerical scheme (2.27) with $\Gamma = 1$ for a final time $T = 10^6$ and various values of Δt (which corresponds to $N_{\text{iter}} = T/\Delta t$ time steps). We also consider various values of n, the subsampling of the data points being done with and without replacement. We report in Figure 2.3 the relative bias on the variance, given in the limit $N_{\text{iter}} \to +\infty$ by the ratio of

$$\left| \left[\int_{\mathbb{R}} \theta^2 \pi_{\Delta t,n}(\theta|\mathbf{x})\, d\theta - \left(\int_{\mathbb{R}} \theta \pi_{\Delta t,n}(\theta|\mathbf{x})\, d\theta \right)^2 \right] - \left[\int_{\mathbb{R}} \theta^2 \pi(\theta|\mathbf{x})\, d\theta - \left(\int_{\mathbb{R}} \theta \pi(\theta|\mathbf{x})\, d\theta \right)^2 \right] \right|,$$

and σ_{post}^2, where $\pi_{\Delta t,n}(\cdot|\mathbf{x})$ denotes the invariant measure in the θ variable of the numerical scheme under consideration (for (2.27), this corresponsd to the marginal measure of $\mu_{\Delta t,n}(\theta, p|\mathbf{x})$ in the θ variable).

Several conclusions can be drawn from the results presented in Figure 2.3. First, note that the bias is determined by the value of $\varepsilon(n)$ irrespectively of the fact that sampling is performed with or without replacement. Second, the bias is indeed affine in $\varepsilon(n)$, with a slope which proportional to Δt. In fact, all curves would be superimposed if we were plotting the error as a function of $\varepsilon(n)\Delta t$, which demonstrates that the error is indeed determined at dominant order by this parameter. We do not report results for $\varepsilon(n) = 0$, which corresponds to $n = N_{\text{data}}$ and sampling without replacement, since the corresponding error is anyway much smaller than the one arising from mini-batching. Finally, errors are very similar for SGLD and the scheme (2.27) associated with Langevin dynamics, as anticipated from the comparison of (2.35) and (2.36) when choosing $\gamma = 1$.



(a) SGLD (b) Langevin dynamics

Figure 2.3 – Relative error on the variance of the posterior distribution for various values of Δt and n when the elementary likelihood is a Gaussian distribution, when sampling with (circles) and without replacement (squares).

2.2.5.2 Mixture of Gaussians

We next consider a more realistic case where the data points are distributed according to a mixture of Gaussians:

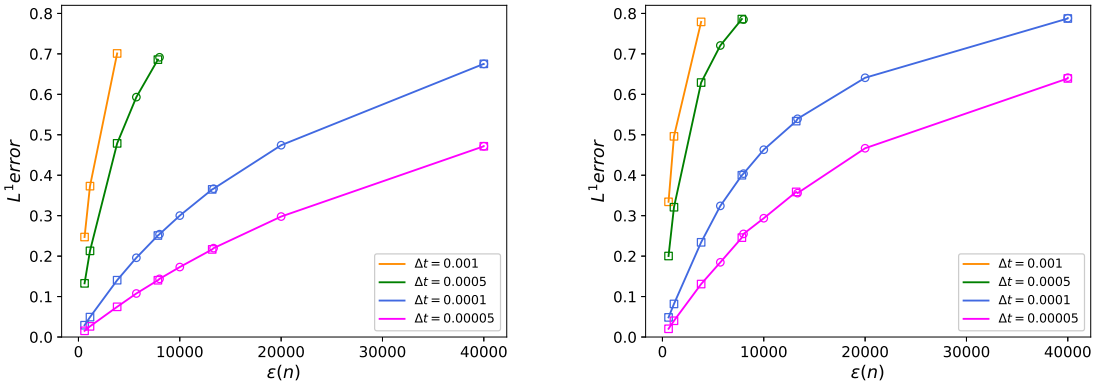
$$P_{\text{elem}}(x_i|\theta) = \frac{w}{\sqrt{2\pi}\sigma_1} \exp\left(-\frac{(x_i-\mu_1)^2}{2\sigma_1^2}\right) + \frac{1-w}{\sqrt{2\pi}\sigma_2} \exp\left(-\frac{(x_i-\mu_2)^2}{2\sigma_2^2}\right), \tag{2.37}$$

where $w \in [0,1]$, $\mu_1, \mu_2 \in \mathbb{R}$ and $\sigma_1, \sigma_2 \in (0, +\infty)$. We consider the case when the parameters to estimate are the centers of the Gaussians $\theta = (\mu_1, \mu_2)$, whereas σ_1, σ_2 and w are given. The prior distribution on the vector of parameters θ is chosen to be a centered normal distribution with covariance matrix I_2. To perform numerical simulations, we fix $\mu_1 = 1$, $\mu_2 = 0.5$, $\sigma_1 = \sigma_2 = 0.4$, $w = 0.4$ and $N_{\text{data}} = 200$ to generate the dataset according to (2.37). We run again the numerical schemes (2.17) and (2.27), with $\Gamma = \mathrm{I}_2$ and an integration time $T = 10^6$. We compute the L^1 error on the marginal distribution in the $\theta_1 = \mu_1$ variable, given in the limit $N_{\text{iter}} \to +\infty$ by

$$\int_{\mathbb{R}} \left| \int_{\mathbb{R}} \pi_{\Delta t,n}(\theta|\mathbf{x})\, d\theta_2 - \int_{\mathbb{R}} \pi(\theta|\mathbf{x})\, d\theta_2 \right| d\theta_1. \tag{2.38}$$

We plot this error with respect to $\varepsilon(n)$ for various values of Δt in Figure 2.4. We use numerical quadratures to approximate the integral with respect to θ_1 in (2.38) (approximating the marginals as piecewise constant functions over a grid of 500 bins over the interval [0,1.4]), and also to compute the integral with respect to θ_2 for $\pi(\cdot|\mathbf{x})$.

The interpretation of the results presented in Figure 2.4 is quite similar to the discussion of the results of Figure 2.3. They confirm that the bias is determined by the value of $\varepsilon(n)$ irrespectively of the fact that sampling is performed with or without replacement. Note also that the errors are quite similar for SGLD and Langevin dynamics with $\Gamma = \mathrm{I}_2$. Moreover, the L^1 error (2.38) is affine in $\varepsilon(n)$ provided $\varepsilon(n)\Delta t$ is sufficiently small so that the asymptotic analysis of the bias is indeed valid. From a quantitative viewpoint, the affine regime is observed for L^1 errors below 0.1. This regime is relevant for time steps $\Delta t \leqslant 10^{-4}$ for the values of n considered in our simulations, but not for simulations with larger time steps.



(a) SGLD (b) Langevin dynamics

Figure 2.4 – L^1 error on the θ_1 marginal of the posterior distribution for various values of Δt and n when the elementary likelihoods are mixtures of Gaussians, when sampling with (circles) and without replacement (squares).

2.3 Adaptive Langevin dynamics

Using SGLD or Langevin dynamics with mini-batching introduces a bias on the posterior distribution. We recall in Section 2.3.1 the Adaptive Langevin (AdL) dynamics [65, 39], whose aim is to remove the bias due to mini-batching. Under the key assumption that the covariance matrix $\Sigma_{\mathbf{x}}(\theta)$ defined in (2.11) is constant (but unknown), it can indeed be proved that Adaptive Langevin dynamics samples the target distribution (see Section 2.3.2). However, we demonstrate in Section 2.3.3 that this assumption may not hold in practice, *e.g.* for models for which the elementary likelihood is a mixture of Gaussians, which motivates the construction of an extended version of AdL to tackle such cases. We also quantify the bias on the invariant measure due to the fact that $\Sigma_{\mathbf{x}}$ is not constant by deriving an error estimate involving (2.1), henceforth explaining why AdL performs better than SGLD or Langevin dynamics with minibatching.

2.3.1 General formulation of Adaptive Langevin dynamics

AdL was initially introduced to address the issue of the gradient of the energy not being exactly computed in molecular dynamics [65]. It was then considered for Bayesian inference in [39] where it allows to remove the bias arising from mini-batching under the assumption that the covariance $\Sigma_{\mathbf{x}}$ is constant. As discussed in Section 2.2.4, the effect of the stochastic estimator (2.13) of the force in Langevin dynamics is, at dominant order, to add an unknown contribution to the diffusion coefficient in front of the Brownian motion, which we denote by $\sqrt{2}A_{\Delta t,n}(\theta)^{1/2}$, with

$$A_{\Delta t,n}(\theta) = \Gamma + \frac{\varepsilon(n)\Delta t}{2}\Sigma_{\mathbf{x}}(\theta). \tag{2.39}$$

Note that $A_{\Delta t,n}(\theta) \in \mathbb{R}^{d\times d}$ is an unknown positive definite symmetric matrix. The idea behind AdL is to modify the effective Langevin dynamics so that it admits $\pi(\cdot|\mathbf{x})$ as an invariant probability measure (more precisely, that it admits an invariant probability measure whose marginal distribution in the θ variable is $\pi(\cdot|\mathbf{x})$). The friction matrix, denoted here by $\xi \in \mathbb{R}^{d\times d}$, is no longer a constant, but a dynamical variable that adjusts itself to the effective noise resulting from mini-batching. Denoting by $[\xi]_{i,j}$ the components of ξ, AdL can

be written as

$$\begin{cases} d\theta_t = p_t\, dt, \\ dp_t = (\nabla(\log \pi(\theta_t|\mathbf{x})) - \xi_t p_t)\, dt + \sqrt{2} A_{\Delta t,n}(\theta_t)^{1/2} dW_t, \\ d[\xi_t]_{i,j} = \dfrac{1}{\eta}\left(p_{i,t}p_{j,t} - \delta_{i,j}\right) dt, \quad 1 \leqslant i,j \leqslant d, \end{cases} \tag{2.40}$$

where η is a positive scalar which sets the timescale for the evolution of the friction matrix (see Remark 2.6 below for more general choices). Let us emphasize once again that we consider that each degree of freedom has mass 1 for simplicity, but our analysis can be easily generalized to non trivial mass matrices, or even to non-quadratic kinetic energies.

Remark 2.5. *If the initial condition ξ_0 is symmetric, then the matrix ξ_t is symmetric for all t. In any case, the quantities $\xi_{i,j,t} - \xi_{j,i,t}$ are constant. This shows that it suffices to introduce $(d+1)d/2$ new variables $([\xi]_{i,j})_{1\leqslant i\leqslant j\leqslant d}$ to simulate AdL. However, for Lemma 2.7 below, it is more convenient to write out statements and proofs with all variables $([\xi]_{i,j})_{1\leqslant i,j\leqslant d}$.*

Remark 2.6. *It is possible to formulate AdL for with different timescale parameters for the evolution of the components of the friction matrix, as defined by a symmetric matrix $\eta = \eta^T \in \mathbb{R}^{d\times d}$ with positive entries. In this case, each element i,j of ξ follows the dynamics*

$$d[\xi_t]_{i,j} = \frac{1}{\eta_{i,j}}\left(p_{i,t}p_{j,t} - \delta_{i,j}\right) dt.$$

Let us conclude this general presentation of AdL by recalling the motivation for the dynamics on the friction variable. We assume for this discussion that $d = 1$ in order to simplify the presentation. One way to understand the intuition behind AdL is to see (2.40) as the superposition of a Hamiltonian dynamics (which preserves the energy $-\log \pi(\theta|\mathbf{x}) + |p|^2/2$) and the following elementary dynamics:

$$\begin{cases} dp_t = -\xi_t p_t\, dt + \sqrt{2} A_{\Delta t,n}^{1/2} dW_t, \\ d\xi_t = \dfrac{1}{\eta}\left(p_t^2 - 1\right) dt. \end{cases}$$

Note that we write $A_{\Delta t,n}^{1/2}$ here instead of $A_{\Delta t,n}(\theta)^{1/2}$ since the value of θ does not change for this subdynamics. Knowing that the marginal distribution over the variable p of the invariant probability measure should be a Gaussian of variance 1 (see [39]), the idea is to keep the right balance between the friction ξ and the fluctuation $A_{\Delta t,n}$. Given that the strength of the fluctuation is unknown, the friction is adjusted so that the average kinetic energy is fixed to its target value, *i.e.* $\mathbb{E}(|p|^2) = 1$. More precisely, if $|p|^2 > 1$, the kinetic energy is larger than what it should be, so the friction is increased, which ends up decreasing p. We can use the same line of argument for the opposite case. In fact, the dynamics of the additional variable ξ follows a negative feedback loop control as in the Nosé–Hoover thermostat [112, 63].

2.3.2 Adaptive Langevin dynamics for gradient estimators with constant covariance

We discuss more precisely in this section the properties of AdL under the crucial assumption that the covariance of the gradient estimator is constant :

$$\Sigma_{\mathbf{x}}(\theta) = \Sigma_{\mathbf{x}}. \tag{2.41}$$

This implies in particular that the matrix $A_{\Delta t,n}$ defined in (2.39) is constant. The limitations of this assumption are discussed more precisely in Section 2.3.3.

We start by discussing invariant probability measures of AdL in Section 2.3.2.1. We next present in Section 2.3.2.2 numerical schemes to integrate AdL, together with error estimates on the bias on their invariant measures. These results are numerically illustrated in Section 2.3.2.3 for the Gaussian model of Section 2.2.5.1, for which the fundamental assumption (2.41) is satisfied. Let us emphasize here again that, in contrast to various works in the literature, we do not necessarily assume that n is much larger than 1 and that the random variable in (2.13) follows a Gaussian distribution.

2.3.2.1 Invariant probability measure of AdL

The generator of the stochastic dynamics (2.40) can be decomposed as

$$\mathcal{L}_{\mathrm{AdL},\Sigma_{\mathbf{x}}} = \mathcal{L}_{\mathrm{ham}} + \mathcal{L}_{\mathrm{FD}} + \mathcal{L}_{\mathrm{NH}},$$

where

$$\mathcal{L}_{\mathrm{NH}} = -p^T(\xi - A_{\Delta t,n})\nabla_p + \frac{1}{\eta}\sum_{1\leqslant i,j\leqslant d}(p_i p_j - \delta_{i,j})\partial_{[\xi]_{i,j}},$$

and

$$\mathcal{L}_{\mathrm{ham}} = p^T\nabla_\theta + \nabla_\theta(\log\pi(\cdot|\mathbf{x}))^T\nabla_p, \qquad \mathcal{L}_{\mathrm{FD}} = A_{\Delta t,n} : \nabla_p^2 - p^T A_{\Delta t,n}\nabla_p.$$

We recall the following result on the invariant probability measure of (2.40) (see [90, Section 2]).

Lemma 2.7. *Suppose that (2.41) holds. Then the dynamics (2.40) admits the following invariant probability measure*

$$\nu(d\theta\, dp\, d\xi) = \pi(\theta|\mathbf{x})\tau(dp)\rho(d\xi)\, d\theta, \tag{2.42}$$

where $\tau(dp)$ is defined in (2.21) and

$$\rho(d\xi) = \prod_{1\leqslant i,j\leqslant d}\sqrt{\frac{\eta}{2\pi}}\exp\left(-\frac{\eta}{2}(\xi_{i,j} - [A_{\Delta t,n}]_{i,j})^2\right) d[\xi]_{i,j},$$

where $[A_{\Delta t,n}]_{i,j}$ is the (i,j) component of $A_{\Delta t,n}$.

Lemma 2.7 suggests that, as long as assumption (2.41) is satisfied, sampling a probability measure using Adaptive Langevin dynamics is not affected by the mini-batching procedure to estimate the gradient of the log-likelihood in the θ variable. The marginal distribution of (2.42) in the variable θ is indeed the target distribution π, whatever the value of $A_{\Delta t,n}$. This shows that AdL can indeed adjust the friction in order to compensate fluctuations of arbitrary constant magnitude. We recall the proof of Lemma 2.7 because we will use similar computations in the proof of Theorem 2.12 below.

Remark 2.8. *Let us emphasize that Lemma 2.7 states that ν is invariant by AdL. However, the dynamics cannot be ergodic for this measure since $[\xi_t]_{i,j} - [\xi_t]_{j,i}$ remains constant, whereas $[\xi]_{i,j}$ and $[\xi_t]_{j,i}$ are independent under the probability measure (2.42). The dynamics can therefore at best be ergodic for the restriction of ν onto the sub-manifold*

$$\mathscr{S}(\xi_0) = \Theta\times\mathbb{R}^d\times\left\{\xi\in\mathbb{R}^{d\times d}\,\middle|\,[\xi]_{i,j} - [\xi]_{j,i} = [\xi_0]_{i,j} - [\xi_0]_{j,i}\right\},$$

which is determined by the initial condition ξ_0. Such an ergodicity result is however not trivial at all since there are $d(d+1)$ independent degrees of freedom in the symmetric matrix ξ_t, while the noise acting on the momentum variable p is only of dimension d. The stochastic dynamics is therefore highly degenerate. In any case, the important point here is that the marginal in the θ-variable of the projected measure is $\pi(\cdot|\mathbf{x})$ whatever the distribution ρ in (2.42).

Proof. We follow the approach of [90, Section 2]. It suffices to show that, for all smooth and compactly supported functions ϕ,

$$\int_\Theta \mathcal{L}_{\mathrm{AdL},\Sigma_\mathbf{x}} \phi \, d\nu = \int_\Theta \phi \left(\mathcal{L}^*_{\mathrm{AdL},\Sigma_\mathbf{x}} \mathbf{1} \right) d\nu = 0, \tag{2.43}$$

where adjoints are taken in $L^2(\nu)$. Simple computations based on integration by parts show that $\partial^*_{\theta_i} = -\partial_{\theta_i} - \partial_{\theta_i}(\log \pi(\theta|\mathbf{x}))$, $\partial^*_{p_i} = -\partial_{p_i} + p_i$, $\partial^*_{[\xi]_{i,j}} = -\partial_{[\xi]_{i,j}} + \eta \left([\xi]_{i,j} - [A_{\Delta t,n}]_{i,j} \right)$. We can then rewrite the generators $\mathcal{L}_{\mathrm{ham}}$ and $\mathcal{L}_{\mathrm{FD}}$ as [90]:

$$\mathcal{L}_{\mathrm{ham}} = \sum_{i=1}^{d} \partial^*_{p_i} \partial_{\theta_i} - \partial^*_{\theta_i} \partial_{p_i}, \tag{2.44}$$

$$\mathcal{L}_{\mathrm{FD}} = -\nabla^*_p A_{\Delta t,n} \nabla_p = - \sum_{1 \leqslant i,j \leqslant d} [A_{\Delta t,n}]_{i,j} \partial^*_{p_i} \partial_{p_j}. \tag{2.45}$$

Moreover, for ϕ, ψ two smooth and compactly supported functions,

$$\begin{aligned}
\int_\Theta (\mathcal{L}_{\mathrm{NH}} \phi) \psi \, d\nu &= \sum_{1 \leqslant i,j \leqslant d} \int_\Theta -p_i ([\xi]_{i,j} - [A_{\Delta t,n}]_{i,j}) \left(\partial_{p_j} \phi \right) \psi + \frac{1}{\eta} (p_i p_j - \delta_{i,j}) \left(\partial_{[\xi]_{i,j}} \phi \right) \psi \, d\nu \\
&= \sum_{1 \leqslant i,j \leqslant d} \int_\Theta -([\xi]_{i,j} - [A_{\Delta t,n}]_{i,j}) \partial^*_{p_j} \left(p_i \psi \right) \phi + \frac{1}{\eta} (p_i p_j - \delta_{i,j}) \left(\partial^*_{[\xi]_{i,j}} \psi \right) \phi \, d\nu \\
&= - \int_\Theta (\mathcal{L}_{\mathrm{NH}} \phi) \psi \, d\nu.
\end{aligned} \tag{2.46}$$

It is then clear that $\mathcal{L}_{\mathrm{ham}}$ and $\mathcal{L}_{\mathrm{NH}}$ are antisymmetric, while $\mathcal{L}_{\mathrm{FD}}$ is symmetric. Moreover, $\mathcal{L}_{\mathrm{NH}} \mathbf{1} = \mathcal{L}_{\mathrm{ham}} \mathbf{1} = \mathcal{L}_{\mathrm{FD}} \mathbf{1} = 0$. The invariance of ν therefore follows from (2.43). □

The mathematical properties of AdL are investigated in [90] in the case when $\Sigma_\mathbf{x} = \sigma^2 \mathrm{I}_d$ with σ^2 constant, and the friction is scalar valued (in which case the invariant probability measure provided by Lemma 2.7 is in fact the only invariant probability measure). The main contributions of [90] are the following: (i) the exponential convergence of the law of the process encoded by the convergence of the semi-group $\mathrm{e}^{t\mathcal{L}}$ is proved using the hypocoercive approach of [61, 41, 42]; (ii) a central limit theorem for time averages along one realization of the dynamics is derived with bounds on the asymptotic variance depending on the parameters η and Γ of the dynamics. If $A_{\Delta t,n} \approx \Gamma$ (which is the case when Δt is sufficiently small and n is sufficiently large), the mathematical analysis suggests to fix $\Gamma = \mathcal{O}(1)$ and $\eta = \mathcal{O}(1)$.

2.3.2.2 Numerical scheme

We now construct a numerical scheme for which AdL in (2.40) is the effective dynamics at dominant order in Δt and $\varepsilon(n)\Delta t$. Concretely, we replace the matrix $A_{\Delta t,n}$ by its expression (2.39) in the SDE (2.40) and consider the symmetric splitting scheme introduced in [90], which is based on decomposing (2.40) into the following four elementary SDEs (the variables which are evolved are indexed by t, while the ones which remain constant do not have any subscript):

$$d\theta_t = p \, dt, \tag{2.47}$$

$$dp_t = -\xi p_t \, dt + \sqrt{2} \Gamma^{1/2} \, dW_{1,t}, \tag{2.48}$$

$$dp_t = \nabla_\theta (\log \pi(\theta|\mathbf{x})) \, dt + \sqrt{\varepsilon(n)\Delta t} \Sigma_\mathbf{x}(\theta)^{1/2} \, dW_{2,t}, \tag{2.49}$$

$$d[\xi_t]_{i,j} = \frac{1}{\eta} (p_i p_j - \delta_{i,j}) \, dt, \qquad 1 \leqslant i \leqslant j \leqslant d, \tag{2.50}$$

where $W_{1,t}, W_{2,t}$ are two independent standard d-dimensional Brownian motions. The elementary ordinary differential equations (2.47) and (2.50) can be trivially integrated. The elementary SDE (2.48) is an Ornstein–Uhlenbeck process that can be analytically integrated in law as

$$p^{m+1} = \mathrm{e}^{-\Delta t \xi} p^m + \sigma_{\xi,\Gamma,\Delta t} G^m, \qquad \sigma^2_{\xi,\Gamma,\Delta t} = 2 \int_0^{\Delta t} \mathrm{e}^{-s\xi} \Gamma \mathrm{e}^{-s\xi} \, ds,$$

where $(G^m)_{m \geqslant 0}$ is a family of i.i.d. standard d-dimensional Gaussian random variables. If there exists M such that $\xi M + M \xi = \Gamma$, then $\sigma^2_{\xi,\Gamma,\Delta t} = 2\left(M - \mathrm{e}^{-\Delta t \xi} M \mathrm{e}^{-\Delta t \xi}\right)$. In particular, for $\Gamma = \gamma \mathrm{I}_d$ and when ξ is invertible, one can choose $M = \gamma \xi^{-1}/2$ in which case $\sigma^2_{\xi,\Gamma,\Delta t} = \gamma \xi^{-1}(\mathrm{I}_d - \mathrm{e}^{-2\Delta t \xi})$. When ξ is singular or close to singular, the latter formula has to be understood through a limiting procedure based on spectral calculus, see [90].

The elementary SDE (2.49) is integrated as

$$p^{m+1} = p^m + \Delta t \widehat{F_n}(\theta) = p^m + \Delta t \nabla_\theta (\log \pi(\theta|\mathbf{x})) + \Delta t \sqrt{\varepsilon(n)} \Sigma_{\mathbf{x}}(\theta)^{1/2} Z_{\mathbf{x},N_{\mathrm{data}},n}. \tag{2.51}$$

Since the random variable $Z_{\mathbf{x},N_{\mathrm{data}},n}$ has mean 0 and identity covariance by construction, the equality (2.73) in Appendix 2.A.b shows that the numerical scheme (2.51) is weakly consistent with (2.49), with an error of order $(1+\varepsilon(n)^{3/2})\Delta t^3$ over one time step (even if $Z_{\mathbf{x},N_{\mathrm{data}},n}$ is not Gaussian). In the case where $\mathbb{E}[Z^3_{\mathbf{x},N_{\mathrm{data}},n}] = 0$, the error over one time step is of order $(1+\varepsilon(n))\Delta t^3$.

The numerical scheme we consider is finally obtained by a Strang splitting where (2.49) is updated in the central step of the algorithm, in order to compute the force only once per time step and avoid its storage. The order in which the remaining elementary dynamics are integrated is unimportant for our purposes, although some orderings may be better than others, in particular in some limiting regimes where one of the parameters goes to 0 or infinity; see [91] for an extensive discussion in the context of AdL. Fixing $\Gamma = \gamma \mathrm{I}_d$, the numerical scheme reads as follows:

$$\begin{cases} p^{m+\frac{1}{2}} = \mathrm{e}^{-\Delta t \xi^m/2} p^m + \left[\gamma (\xi^m)^{-1} \left(\mathrm{I}_d - \mathrm{e}^{-\Delta t \xi^m}\right)\right]^{1/2} G^m, \\ \xi^{m+\frac{1}{2}} = \xi^m + \dfrac{\Delta t}{2\eta}\left(p^{m+\frac{1}{2}} \left(p^{m+\frac{1}{2}}\right)^T - \mathrm{I}_d\right), \\ \theta^{m+\frac{1}{2}} = \theta^m + \dfrac{\Delta t}{2} p^{m+\frac{1}{2}}, \\ \widetilde{p}^{m+\frac{1}{2}} = p^{m+\frac{1}{2}} + \Delta t \widehat{F_n}\left(\theta^{m+\frac{1}{2}}\right), \\ \theta^{m+1} = \theta^{m+\frac{1}{2}} + \dfrac{\Delta t}{2} \widetilde{p}^{m+\frac{1}{2}}, \\ \xi^{m+1} = \xi^{m+\frac{1}{2}} + \dfrac{\Delta t}{2\eta}\left(\widetilde{p}^{m+\frac{1}{2}} \left(\widetilde{p}^{m+\frac{1}{2}}\right)^T - \mathrm{I}_d\right), \\ p^{m+1} = \mathrm{e}^{-\Delta t \xi^{m+1}/2} \widetilde{p}^{m+\frac{1}{2}} + \left[\gamma (\xi^{m+1})^{-1} \left(\mathrm{I}_d - \mathrm{e}^{-\Delta t \xi^{m+1}}\right)\right]^{1/2} G^{m+\frac{1}{2}}, \end{cases} \tag{2.52}$$

where $(G^m)_{m \geqslant 0}$ and $(G^{m+\frac{1}{2}})_{m \geqslant 0}$ are two independent families of i.i.d. standard Gaussian random variables. Note that it is possible to work only with the additional variables $(\xi_{i,j})_{1 \leqslant i \leqslant j \leqslant d}$ since the updates of $\xi_{i,j}$ and $\xi_{j,i}$ in (2.40) are the same. The initial conditions for ξ^0 is $\gamma \mathrm{Id}$, while the components of θ^0, p^0 are initialized to 0. Of course, more educated choices can be considered depending on the system at hand (*e.g.* restarting from values sampled by SGLD or some Langevin-like dynamics).

Remark 2.9. *Other versions of the numerical scheme (2.52) can be considered, in particular the one used in [90], for which $\xi \in \mathbb{R}$. In this case, (2.50) should be replaced by*

$$d\xi_t = \frac{1}{\eta}\left(p^T p - d\right) dt. \tag{2.53}$$

Another option to reduce the number of additional variables is to consider the variable ξ as a diagonal matrix with entries ξ_i. In this case, (2.50) has to be replaced with

$$d[\xi_t]_i = \frac{1}{\eta}\left(p_i^2 - 1\right) dt, \qquad 1 \leqslant i \leqslant d. \tag{2.54}$$

The numerical schemes for these two cases can be obtained by an obvious modification of the right hand side of the update formulas for $\xi^{m+\frac{1}{2}}$ and ξ^{m+1} in (2.52).

Using the Baker–Campbell–Hausdorff formula (see for instance [55, Section III.4.2]), as done in [87, 91] for instance, one can prove that the evolution operator $\widehat{P}_{\Delta t,n}$ of the numerical scheme (2.52) satisfies (see Appendix 2.A.c)

$$\widehat{P}_{\Delta t,n} = \mathrm{e}^{\Delta t \mathcal{L}_{\mathrm{AdL},\Sigma_{\mathbf{x}}}} + \mathcal{O}\left((1+\varepsilon(n)^{3/2})\Delta t^3\right). \tag{2.55}$$

Assume next that (2.40) and (2.52) both admit a unique invariant probability measure, respectively denoted by ν_{ξ_0} and $\nu_{\xi_0,\Delta t,n}$. Then, the following error estimate holds: for any smooth function φ with compact support, there exists $C \in \mathbb{R}_+$ such that

$$\left|\int_\Theta \varphi(\theta,p,\xi)\,\nu_{\xi_0,\Delta t,n}(d\theta\,dp\,d\xi) - \int_\Theta \varphi(\theta,p,\xi)\,\nu_{\xi_0}(d\theta\,dp\,d\xi)\right| \leqslant C\left(1+\varepsilon(n)^{3/2}\right)\Delta t^2. \tag{2.56}$$

Note that this error estimate holds even when $Z_{\mathbf{x},N_{\mathrm{data}},n}$ is not Gaussian (namely for small values of n). When $\mathbb{E}[Z^3_{\mathbf{x},N_{\mathrm{data}},n}] = 0$, the error estimate (2.56) holds with $C(1+\varepsilon(n))\Delta t^2$ on the right hand side. When $Z_{\mathbf{x},N_{\mathrm{data}},n}$ is Gaussian, the integration (2.51) is in fact exact in law. In any case, the analysis we made crucially depends on the fact that (2.41) holds. When it does not hold, there is an additional bias on the invariant measure due to mini-batching, as quantified in Section 2.3.3.

2.3.2.3 Numerical illustration for Gaussian likelihoods

We consider the same setting as in Section 2.2.5.1. The covariance $\Sigma_{\mathbf{x}}$, given by (2.34), is constant for this model, so that assumption (2.41) indeed holds. We run the numerical scheme (2.52) with $\gamma = 1$ for an integration time $T = 10^6$. Similar computations have already been performed in [39, 91]. In Figure 2.5, we plot the error on the variance of the marginal posterior distribution in the θ variable with respect to $\varepsilon(n)$. First, we note that, for Δt small enough (namely $\Delta t \leqslant 0.005$), the bias is not affected by the value of $\varepsilon(n)$ (even for the very large value N^2_{data} obtained when $n = 1$), and hence by the size of the mini-batch. The bias is therefore simply due to the discretization time step. For large values of Δt (namely $\Delta t \geqslant 0.008$ in our experiments), the error is affected by the value of n. This is probably due to the fact that this rather large value of Δt is out of the regime where the asymptotic analysis for the bias holds. In any case, a comparison with Figure 2.3 shows that the error obtained with AdL is much smaller than the one obtained with SGLD (2.17) or the numerical scheme (2.27) (which corresponds to Langevin dynamics with mini-batching), even with much larger time steps.

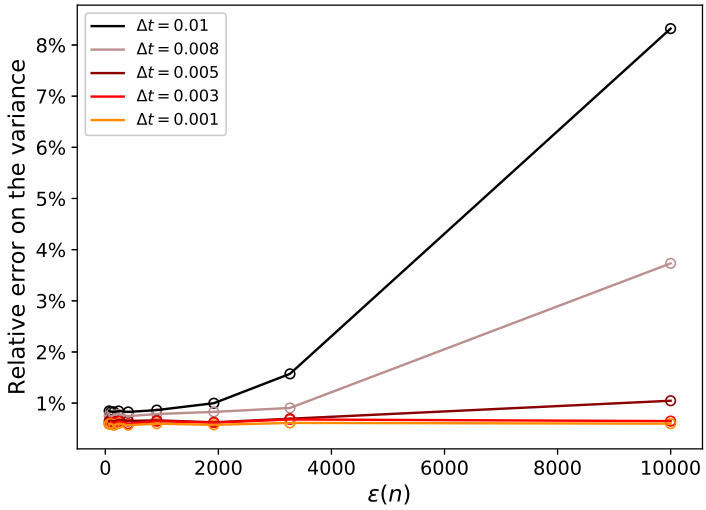


Figure 2.5 – Relative error on the variance of the posterior distribution for various values of n using AdL when the elementary likelihood is Gaussian, and sampling is performed without replacement.

2.3.3 Impact of a non constant covariance matrix

The bias analysis in the previous section does not hold in the case of a non constant covariance matrix $\Sigma_{\mathbf{x}}$. We first quantify in Section 2.3.3.1 the bias due to mini-batching when assumption (2.41) is not satisfied. We illustrate our analysis in Section 2.3.3.2 with a numerical example where the elementary likelihood is a mixture of Gaussians. In this case, $\Sigma_{\mathbf{x}}(\theta)$ genuinely depends on θ.

2.3.3.1 Mini-batching bias for Adaptive Langevin dynamics and non constant covariance

We consider the situation where $\Sigma_{\mathbf{x}}(\theta)$ genuinely depends on θ, and prove that the bias on the invariant measure is of order $\mathcal{O}(\varepsilon(n)\Delta t) + \mathcal{O}(\Delta t^2)$. Denote by

$$\overline{\Sigma}_{\mathbf{x}} = \int_\Theta \Sigma_{\mathbf{x}}(\theta)\pi(\theta|\mathbf{x})\, d\theta \tag{2.57}$$

the average covariance of the estimator of the force. The motivation for introducing this particular constant matrix is discussed after Lemma 2.10 below. The generator of (2.40) can then be written as

$$\mathcal{L}_{\mathrm{AdL},\Sigma_{\mathbf{x}}} = \mathcal{L}_{\mathrm{AdL},\overline{\Sigma}_{\mathbf{x}}} + \varepsilon(n)\Delta t\widetilde{\mathcal{L}}_{\Sigma_{\mathbf{x}}-\overline{\Sigma}_{\mathbf{x}}}, \qquad \widetilde{\mathcal{L}}_{\mathscr{M}} = \mathscr{M} : \nabla_p^2. \tag{2.58}$$

Since $\overline{\Sigma}_{\mathbf{x}}$ does not depend on θ, the probability measure ν defined in (2.42) is an invariant probability measure of the dynamics with generator $\mathcal{L}_{\mathrm{AdL},\overline{\Sigma}_{\mathbf{x}}}$. As in Sections 2.2.3 and 2.2.4, we next use [94, Remark 5.5] to prove that the extra bias due to mini-batching on the invariant measure of (2.40) when $\Sigma_{\mathbf{x}}$ is not constant is of order $\mathcal{O}(\varepsilon(n)\Delta t)$. More precisely, denoting by ν_{ξ_0} the invariant probability measure of the continuous dynamics (2.40) when $\Sigma_{\mathbf{x}}$ depend on the θ variable, and by $\overline{\nu}_{\xi_0}$ the invariant probability measure of the same dynamics (2.40) when the covariance is set to $\overline{\Sigma}_{\mathbf{x}}$ (recall Remark 2.8), it can be proved under appropriate ergodicity conditions that, for any smooth function φ with compact support, there exists $C \in \mathbb{R}_+$ such that

$$\left| \int_\Theta \varphi(\theta,p,\xi)\nu_{\xi_0}(d\theta\, dp\, \xi) - \int_\Theta \varphi(\theta,p,\xi)\left[1+\varepsilon(n)\Delta t f_{\mathrm{AdL},\Sigma_{\mathbf{x}}}(\theta,p,\xi)\right]\overline{\nu}_{\xi_0}(d\theta\, dp\, d\xi) \right| \leqslant C\varepsilon(n)^2\Delta t^2, \tag{2.59}$$

where (adjoints being taken on $L^2(\overline{\nu}_{\xi_0})$)

$$f_{\mathrm{AdL},\Sigma_{\mathbf{x}}} = \left(-\mathcal{L}^*_{\mathrm{AdL},\overline{\Sigma}_{\mathbf{x}}}\right)^{-1} \widetilde{\mathcal{L}}^*_{\Sigma_{\mathbf{x}}-\overline{\Sigma}_{\mathbf{x}}} \mathbf{1}.$$

This shows that the difference between ν_{ξ_0} and $\overline{\nu}_{\xi_0}$ is of order $\varepsilon(n)\Delta t$, with a magnitude related to the norm of $f_{\mathrm{AdL},\Sigma_{\mathbf{x}}}$ in $L^2(\overline{\nu}_{\xi_0})$. The latter quantity is estimated in the following lemma.

Lemma 2.10. *Assume that the marginal distributions of $\overline{\nu}_{\xi_0}$ in the θ and p variables are respectively $\pi(\cdot|\mathbf{x})$ and τ, and that the resolvent $\mathcal{L}^{-1}_{\mathrm{AdL},\overline{\Sigma}_{\mathbf{x}}}$ is bounded on the subspace of functions in $L^2(\overline{\nu}_{\xi_0})$ with average 0 with respect to $\overline{\nu}_{\xi_0}$. Then there exists $C \in \mathbb{R}_+$ such that*

$$\|f_{\mathrm{AdL},\Sigma_{\mathbf{x}}}\|_{L^2(\overline{\nu}_{\xi_0})} \leqslant C \left\|\Sigma_{\mathbf{x}} - \overline{\Sigma}_{\mathbf{x}}\right\|_{L^2(\pi)}.$$

The boundedness of the resolvent has been proved when ξ is scalar valued, see [90]. Lemma 2.10 provides a motivation on the choice of $\overline{\Sigma}_{\mathbf{x}}$. Indeed, the error estimate in Lemma 2.10 would be true with $\overline{\Sigma}_{\mathbf{x}}$ replaced by any constant, positive, symmetric matrix $M \in \mathbb{R}^{d\times d}$. The optimal choice is the $L^2(\pi)$-orthogonal projection $\overline{\Sigma}_{\mathbf{x}}$ of $\Sigma_{\mathbf{x}}$ onto constant matrices.

Proof. Note first that $\widetilde{\mathcal{L}}^*_{\Sigma_{\mathbf{x}}-\overline{\Sigma}_{\mathbf{x}}} \mathbf{1} = \left(\Sigma_{\mathbf{x}} - \overline{\Sigma}_{\mathbf{x}}\right) : \left(\nabla_p^2\right)^* \mathbf{1}$. The (i,j)-component of $\left(\nabla_p^2\right)^* \mathbf{1}$ is $\partial^*_{p_i}\partial^*_{p_j}\mathbf{1} = p_i p_j - \delta_{i,j}$. Since

$$\int_{\mathbb{R}^d} (p_i p_j - \delta_{i,j})^2 \, \tau(p)\, dp = 1 + \delta_{i,j},$$

we obtain, by a Cauchy–Schwarz inequality,

$$\left\|\widetilde{\mathcal{L}}^*_{\Sigma_{\mathbf{x}}-\overline{\Sigma}_{\mathbf{x}}}\mathbf{1}\right\|^2_{L^2(\overline{\nu}_{\xi_0})} \leqslant \left\|\Sigma_{\mathbf{x}} - \overline{\Sigma}_{\mathbf{x}}\right\|^2_{L^2(\pi)} \sum_{i,j=1}^d \int_{\mathbb{R}^d} (p_i p_j - \delta_{i,j})^2 \, \tau(p)\, dp = d(d+1) \left\|\Sigma_{\mathbf{x}} - \overline{\Sigma}_{\mathbf{x}}\right\|^2_{L^2(\pi)}.$$

The conclusion then follows from the assumed boundedness of the resolvent. □

We finally deduce from (2.56) and (2.59) that the total error between the invariant probability measures for (2.40) and (2.52) is of order $\mathcal{O}(\varepsilon(n)\Delta t) + \mathcal{O}(\Delta t^2)$. However, as motivated by Lemma 2.10, the prefactor for the error term $\varepsilon(n)\Delta t$ is much smaller than the one for SGLD and Langevin dynamics, and depends only on the deviation of the covariance of the gradient from its average.

Remark 2.11. *Using other versions of the numerical scheme (2.52) (see Remark 2.9) affects the prefactor for the error term $\varepsilon(n)\Delta t$. If we consider the case where the variable ξ is a scalar, in other words, if we consider (2.53) for the numerical scheme, $\overline{\Sigma}_{\mathbf{x}}$ in (2.57) should be replaced by $s^*\mathrm{Id}_d$, with*

$$s^* = \frac{1}{d}\int_\Theta \mathrm{Tr}\left(\Sigma_{\mathbf{x}}(\theta)\right) \pi(\theta|\mathbf{x})\, d\theta.$$

In this case, the prefactor in front of the minibatching error term is larger than $\|\Sigma_{\mathbf{x}} - \overline{\Sigma}_{\mathbf{x}}\|_{L^2(\pi)}$, as it is given by $\|\Sigma_{\mathbf{x}} - s^*\mathrm{I}_d\|_{L^2(\pi)}$. If we consider the variable ξ to be a diagonal matrix, i.e. *we consider (2.54) for the numerical scheme, $\overline{\Sigma}_{\mathbf{x}}$ in (2.57) should be replaced by a diagonal matrix with entries*

$$\int_\Theta [\Sigma_{\mathbf{x}}(\theta)]_{i,i}\, \pi(\theta|\mathbf{x})\, d\theta, \qquad 1 \leqslant i \leqslant d.$$

By interpreting the various $L^2(\pi)$-norms of the difference between $\Sigma_{\mathbf{x}}$ and a constant matrix as the distance to finite dimensional subspaces of $L^2(\pi)$ included in each other, one can conclude that the projection error obtained for diagonal matrices is between the error obtained with full matrices and isotropic ones.

2.3.3.2 Mixture of Gaussians

We illustrate here that the bias on the posterior distribution is indeed of order $\mathcal{O}(\varepsilon(n)\Delta t+\Delta t^2)$ when (2.41) does not hold. We consider to this end the model described in Section 2.2.5.1 with the same parameters as in that section. Assuming that the data points x_i are distributed according to $P_{\text{elem}}(\cdot|\theta_0)$ for some θ_0, the covariance $\Sigma_{\text{x}}(\theta)$ has the following limit when the number of data points N_{data} is large:

$$\begin{aligned}\Sigma(\theta) &:= \lim_{N_{\text{data}}\to\infty} \Sigma_{\mathbf{x}}(\theta) \\ &= \int_{\mathcal{X}} \nabla_\theta(\log P_{\text{elem}}(x|\theta))\nabla_\theta(\log P_{\text{elem}}(x|\theta))^T P_{\text{elem}}(x|\theta_0)\,dx \\ &- \left(\int_{\mathcal{X}} \nabla_\theta(\log P_{\text{elem}}(x|\theta))P_{\text{elem}}(x|\theta_0)dx\right)\left(\int_{\mathcal{X}} \nabla_\theta(\log P_{\text{elem}}(x|\theta))P_{\text{elem}}(x|\theta_0)dx\right)^T.\end{aligned} \tag{2.60}$$

This limit is well defined provided $\nabla_\theta(\log P_{\text{elem}}(\cdot|\theta)) \in L^2(P_{\text{elem}}(\cdot|\theta_0))$ for all $\theta \in \Theta$. Computing $\Sigma(\theta)$ in (2.60) is intractable in general for two reasons: (i) θ_0 is unknown and anyway the data points may even not be distributed according to $P_{\text{elem}}(\cdot|\theta_0)$; (ii) the integral over x is possibly a high dimensional one or requires an evaluation cost of $\mathcal{O}(N_{\text{data}})$ to approximate it. We nonetheless use the formula (2.60) to compute the elements of $\Sigma(\theta)$ in the low-dimensional case considered here to better understand the issues at stake. We plot in Figure 2.6 the components $\Sigma_{11}(\theta)$, $\Sigma_{12}(\theta)$ and $\Sigma_{22}(\theta)$ of the symmetric matrix $\Sigma(\theta)$ in (2.60). It is clear that the variance genuinely depends on the value of the parameter $\theta = (\theta_1, \theta_2)$, so that assumption (2.41) fails in this case.

To check whether the failure of assumption (2.41) has an impact on the properties of AdL, we use the same dataset as described in Section 2.2.5.2. We run the numerical scheme (2.52) for the following two cases: when ξ is a $d \times d$ matrix or when it is a scalar. We fix $T = 10^6$, $\gamma = 1$ and $\eta = 0.1$. This last choice is motivated by metastability issues we noticed while using AdL for larger values of η. We report in Figure 2.7 the L^1 error given by (2.38). Note first that using AdL greatly reduces the L^1 error compared to the results of Figure 2.4, obtained with SGLD (2.17) or the numerical scheme (2.27) (which corresponds to Langevin dynamics with mini-batching). However, AdL, in both cases, fails to completely correct the bias due to mini-batching. Consistently with the analysis of Section 2.3.3.1, we observe that the bias seems to scale linearly with $\varepsilon(n)\Delta t\|\Sigma_{\mathbf{x}}(\theta) - S^*\|_{L^2(\pi)}$ (where S^* is the $L^2(\pi)$-projection of $\Sigma_{\mathbf{x}}$ onto the set of admissible matrices) when this quantity is sufficiently small. We numerically compute

$$\left\|\Sigma_{\mathbf{x}}(\theta) - \frac{1}{d}\int_\Theta \text{Tr}\left(\Sigma_{\mathbf{x}}(\theta)\right)\pi(\theta|\mathbf{x})\,d\theta\right\|_{L^2(\pi)} \approx 1.75$$

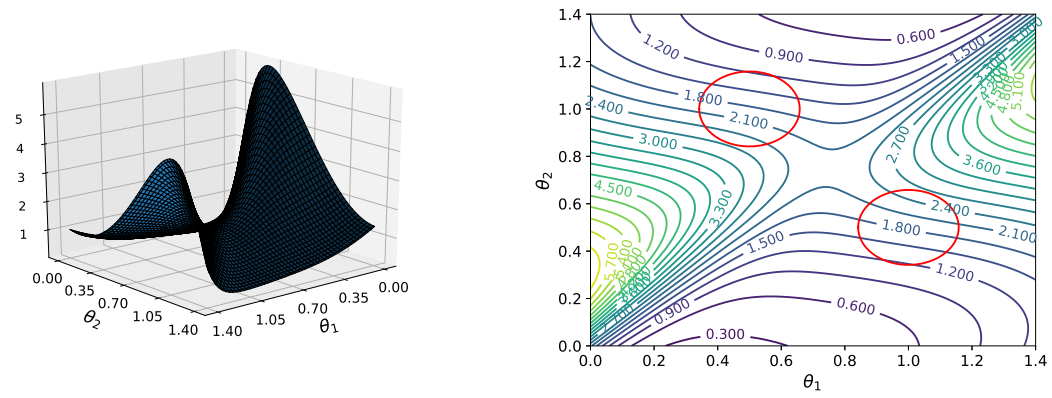
for the scalar case, and

$$\left\|\Sigma_{\mathbf{x}}(\theta) - \int_\Theta \Sigma_{\mathbf{x}}(\theta)\pi(\theta|\mathbf{x})\,d\theta\right\|_{L^2(\pi)} \approx 0.57$$

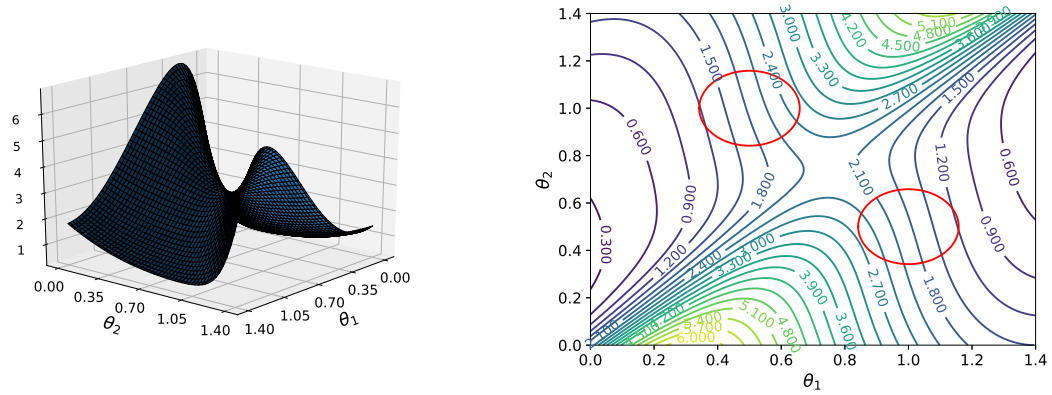
for the full matrix case. This is consistent with the numerical results for $\varepsilon(n)\Delta t \leqslant 100$, where the error on the posterior distribution in the scalar case is roughly 3 times larger than the error in the full matrix case.

2.4 Extended Adaptive Langevin Dynamics

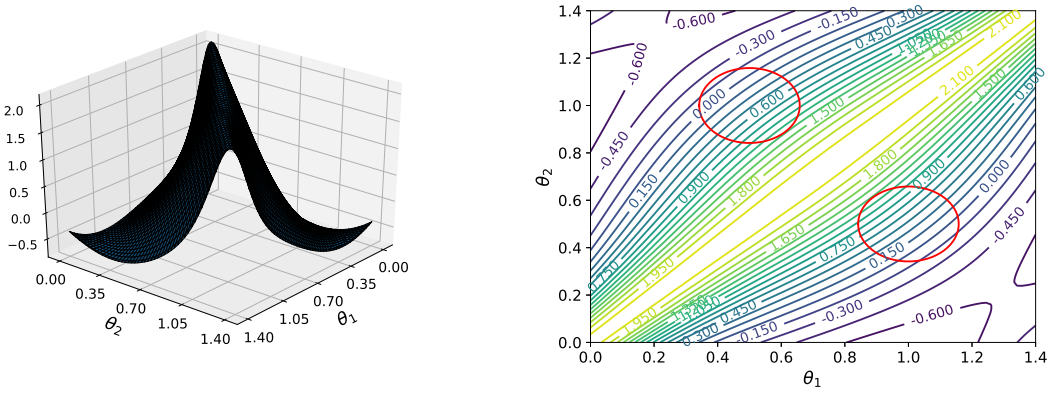
AdL corrects for the bias due to mini-batching when the covariance of the stochastic gradient is constant. This is unfortunately not always the case as we demonstrated in Section 2.3.3.



(a) $\Sigma_{1,1}$

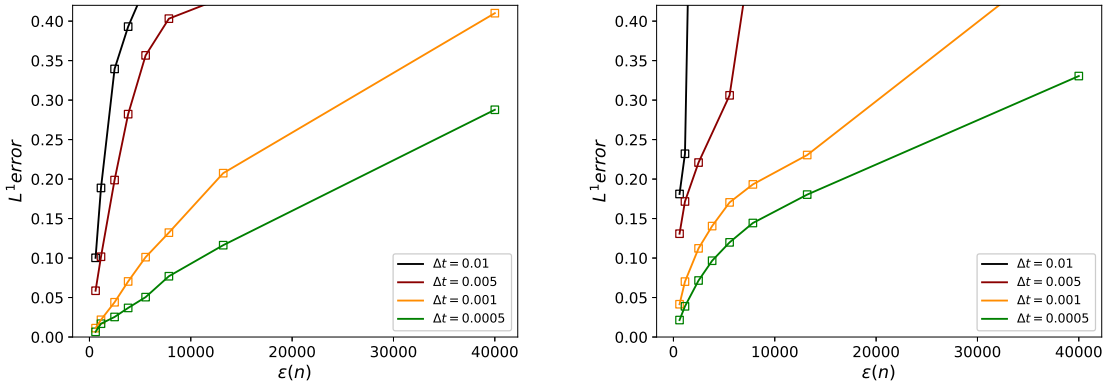


(b) $\Sigma_{2,2}$



(c) $\Sigma_{1,2}$

Figure 2.6 – The three components of the elementary covariance matrix Σ in (2.60) when the elementary likelihood are mixture of Gaussians: surface (left) and contour (right) plots. The modes of the posterior distribution are indicated by orange ellipses.



(a) ξ matrix (b) ξ scalar

Figure 2.7 – L^1 error on the posterior distribution for various values of Δt and n when sampling from the posterior distribution for the mixture of Gaussians case using AdL, with mini-batching performed without replacement.

In this section, we introduce an extension of AdL dynamics which allows to remove the bias when the covariance is not constant but can be decomposed on a finite basis of functions. This approach also allows to drastically reduce the bias even if the covariance matrix cannot be decomposed in a finite basis.

We start by presenting the modified AdL dynamics in Section 2.4.1, under the assumption that the covariance of the force estimator can be decomposed on a finite basis of functions. This key assumption guarantees that the marginal distribution in the θ variable is the target posterior (2.2), which is one important result of this work. We next propose a numerical scheme for extended AdL in Section 2.4.2, where we also quantify the bias on the invariant measure arising from the use of finite time steps, and possibly from mini-batching in situations where the covariance of the force estimator cannot be decomposed as a finite sum for the chosen basis of functions. We finally discuss in Section 2.4.3 a crucial point of the method, namely the choice of basis functions.

2.4.1 Presentation of the dynamics

Using the same notation as in Section 2.3, let us consider the case where $A_{\Delta t,n}(\theta)$ in (2.39) genuinely depends on θ in the following manner.

Assumption 1. *The matrix-valued function $A_{\Delta t,n}$ can be decomposed on a finite basis of functions $f_0, \dots, f_K$ as*

$$A_{\Delta t,n}(\theta) = \sum_{k=0}^{K} A^k_{\Delta t,n} f_k(\theta), \tag{2.61}$$

where $A^0_{\Delta t,n}, ..., A^K_{\Delta t,n} \in \mathbb{R}^{d\times d}$ are symmetric matrices. Moreover, $A_{\Delta t,n}(\theta)$ is a symmetric positive matrix for any $\theta \in \Theta$.

The choice of the basis of function is of great importance for the numerical performance of the method and is discussed more precisely in Section 2.4.3. The situation when (2.61) does not hold is considered at the end of Section 2.4.2. Note that the matrices $A^0_{\Delta t,n}, ..., A^K_{\Delta t,n}$ need not be positive as long as $A_{\Delta t,n}(\theta)$ is. In accordance with (2.61), we choose the variable $\xi_t(\theta)$ to be of the following form:

$$\xi_t(\theta) = \sum_{k=0}^{K} \xi_{k,t} f_k(\theta), \qquad \xi_{k,t} \in \mathbb{R}^{d\times d}. \tag{2.62}$$

If $K = 0$ and $f_0 = \mathbf{1}$, then (2.61) coincides with assumption (2.41), in which case AdL is sufficient to remove the bias due to mini-batching. In practice, the expression (2.61) is obtained by a truncation of the expansion of the function $A_{\Delta t,n}(\theta)$ on a complete basis. From a pragmatic viewpoint, an appropriate value of K is such that

$$\left| A_{\Delta t,n} - \sum_{k=0}^{K} A^k_{\Delta t,n} f_k \right| \ll |A_{\Delta t,n}|, \tag{2.63}$$

for some matrix norm $|\cdot|$ (*e.g.* the Frobenius norm).

We are now in position to write the following extended Adaptive Langevin dynamics (eAdL), for which we introduce $K+1$ additional (matrix) equations on the coefficients $\xi_{k,t}$ in (2.62):

$$\begin{cases} d\theta_t = p_t\, dt, \\ dp_t = \nabla_\theta(\log \pi(\theta_t|\mathbf{x}))\, dt - \xi_t(\theta_t) p_t\, dt + \sqrt{2} A_{\Delta t,n}(\theta_t)^{1/2}\, dW_t, \\ d[\xi_{k,t}]_{i,j} = \dfrac{f_k(\theta_t)}{\eta_k} \left(p_{i,t} p_{j,t} - \delta_{i,j} \right), \qquad 1 \leqslant i,j \leqslant d, \quad 0 \leqslant k \leqslant K, \end{cases} \tag{2.64}$$

where $A_{\Delta t,n}$ given by (2.39), $[\xi_{k,t}]_{i,j}$ is the (i,j) component of $\xi_{k,t} \in \mathbb{R}^{d\times d}$, and η_k are positive scalars for $0 \leqslant k \leqslant K$. The interest of eAdL is the following consistency result on the existence of an invariant probability measure with the correct marginal distribution in the θ variable.

Theorem 2.12. *Suppose that Assumption 1 holds. Then, the eAdL dynamics (2.64) admits the following invariant probability measure:*

$$\nu_K(d\theta\, dp\, d\xi_0 \dots d\xi_K) = \pi(\theta|\mathbf{x}) \tau(dp) \rho_K(d\xi)\, d\theta, \tag{2.65}$$

where $\tau(dp)$ is defined in (2.21), and

$$\rho_K(d\xi_0 \dots d\xi_K) = \prod_{k=0}^{K} \prod_{i,j=1}^{d} \sqrt{\frac{\eta_k}{2\pi}} \exp\left(-\frac{\eta_k}{2} \left([\xi_k]_{i,j} - [A^k_{\Delta t,n}]_{i,j} \right)^2 \right) d[\xi_k]_{i,j},$$

with $[A^k_{\Delta t,n}]_{i,j}$ the (i,j) component of $A^k_{\Delta t,n} \in \mathbb{R}^{d\times d}$.

As for AdL (see the discussion following Lemma 2.7), Theorem 2.12 suggests that we recover the target posterior distribution $\pi(\cdot|\mathbf{x})$ whatever the extra noise due to mini-batching, since the marginal in the variable θ of the probability measure ν_K in (2.65) is $\pi(\cdot|\mathbf{x})$. However, from a discussion similar to the one in Remark 2.8, the dynamics can not be ergodic for the extended measure ν_K, as discussed in the following remark.

Remark 2.13. *As discussed in Remarks 2.5 and 2.8, it is easier to consider all the additional variables $([\xi_k]_{i,j})_{1\leqslant i,j\leqslant d}$ for all $k \in \{0, ..., K\}$ (and not just the upper part of these matrices) to prove the invariance of the probability measure ν_K. However, for the same reasons as for AdL, the dynamics cannot be ergodic for the probability measure (2.65). The best case scenario is that the dynamics is ergodic for the restriction of ν_K onto the submanifold*

$$\mathscr{S}_K(\xi_0) = \Theta \times \mathbb{R}^d \times \prod_{k=0}^{K} \left\{ \xi_k \in \mathbb{R}^{d\times d} \,\middle|\, [\xi_k]_{i,j} - [\xi_k]_{j,i} = [\xi_{k,0}]_{i,j} - [\xi_{k,0}]_{j,i} \right\}.$$

The latter ergodicity is however unclear, since there are now $(K+1)d(d+1)$ additional degrees of freedom in the friction variables, whereas the noise acting on the system is only of dimension d. In any case, this does not affect the main result, since the marginal over θ is still $\pi(\cdot|\mathbf{x})$ as soon as the invariant measure has the tensorized structure (2.65), even if the probability measure on $\xi_0, \dots, \xi_K$ is supported on a submanifold. Let us emphasize that it is sufficient to consider only $([\xi_k]_{i,j})_{1\leqslant i\leqslant j\leqslant d}$ for the numerical experiments.

Proof. We follow the same approach as for the proof of Lemma 2.7. The generator of the dynamics reads

$$\mathcal{L}_{\mathrm{eAdL},\Sigma_{\mathbf{x}}} = \mathcal{L}_{\mathrm{ham}} + \mathcal{L}_{\mathrm{FD}} + \widetilde{\mathcal{L}}_{\mathrm{NH}},$$

where $\mathcal{L}_{\mathrm{ham}}$ and $\mathcal{L}_{\mathrm{FD}}$ are the same operators as in (2.44) and (2.45) respectively (even if $A_{\Delta t,n}$ depends on θ), while

$$\widetilde{\mathcal{L}}_{\mathrm{NH}} = \sum_{k=0}^{K} f_k \mathcal{L}_{\mathrm{NH},k},$$

where

$$\mathcal{L}_{\mathrm{NH},k} = -\sum_{i,j=1}^{d} p_i \left([\xi_k]_{i,j} - [A^k_{\Delta t,n}]_{i,j} \right) \partial^*_{p_j} + \frac{1}{\eta_k}(p_i p_j - \delta_{i,j}) \partial_{[\xi_k]_{i,j}}.$$

A computation similar to the one performed in the proof of Lemma 2.7 shows that $\mathcal{L}^*_{\mathrm{NH},k} = -\mathcal{L}_{\mathrm{NH},k}$, with adjoints taken with respect to $L^2(\nu_K)$; while $\mathcal{L}_{\mathrm{ham}}$ and $\mathcal{L}_{\mathrm{FD}}$ are respectively antisymmetric and symmetric on $L^2(\nu_K)$. This implies that $\mathcal{L}^*_{\mathrm{eAdL},\Sigma_{\mathbf{x}}} \mathbf{1} = 0$, from which the claimed invariance of ν_K follows. □

2.4.2 Numerical scheme and estimates on the bias

We present in this section a numerical integrator for the eAdL dynamics (2.64) based on a Strang splitting similar to the one considered in Section 2.3.2.2 for AdL. We consider the same elementary SDEs (2.47)-(2.50) as for the discretization of AdL, except that there are now $K+1$ elementary SDEs in (2.50), indexed by $0 \leqslant k \leqslant K$. The associated numerical scheme obtained for $\Gamma = \gamma \mathrm{I}_d$ reads

$$\begin{cases} \xi^m = \displaystyle\sum_{k=0}^{K} \xi_k^m f_k(\theta^m), \\ p^{m+\frac{1}{2}} = \mathrm{e}^{-\Delta t \xi^m/2} p^m + \left[\gamma \left(\xi^m\right)^{-1} \left(\mathrm{I}_d - \mathrm{e}^{-\Delta t \xi^m} \right) \right]^{1/2} G^m, \\ \xi_k^{m+\frac{1}{2}} = \xi_k^m + \dfrac{\Delta t}{2\eta_k} f_k(\theta^m) \left[\left(p^{m+\frac{1}{2}} \right) \left(p^{m+\frac{1}{2}} \right)^T - \mathrm{I}_d \right], \qquad k = 0, ..., K, \\ \theta^{m+\frac{1}{2}} = \theta^m + \dfrac{\Delta t}{2} p^{m+\frac{1}{2}}, \\ \widetilde{p}^{m+\frac{1}{2}} = p^{m+\frac{1}{2}} + \Delta t \widehat{F_n} \left(\theta^{m+\frac{1}{2}} \right), \\ \theta^{m+1} = \theta^{m+\frac{1}{2}} + \dfrac{\Delta t}{2} \widetilde{p}^{m+\frac{1}{2}}, \\ \xi_k^{m+1} = \xi_k^{m+\frac{1}{2}} + \dfrac{\Delta t}{2\eta_k} f_k(\theta^{m+1}) \left[\left(\widetilde{p}^{m+\frac{1}{2}} \right) \left(\widetilde{p}^{m+\frac{1}{2}} \right)^T - \mathrm{I}_d \right], \qquad k = 0, ..., K, \\ \xi^{m+1} = \displaystyle\sum_{k=0}^{K} \xi_k^{m+1} f_k(\theta^{m+1}), \\ p^{m+1} = \mathrm{e}^{-\Delta t \xi^{m+1}/2} \widetilde{p}^{m+\frac{1}{2}} + \left[\gamma \left(\xi^{m+1}\right)^{-1} \left(\mathrm{I}_d - \mathrm{e}^{-\Delta t \xi^{m+1}} \right) \right]^{1/2} G^{m+\frac{1}{2}}, \end{cases} \tag{2.66}$$

where $(G^m)_{m \geqslant 0}$ and $(G^{m+\frac{1}{2}})_{m \geqslant 0}$ are two independent families of i.i.d. standard d-dimensional Gaussian random variables.

We assume as before that the numerical scheme (2.66) admits a unique invariant probability measure, which may depend on the initial condition $(\xi_0^0, \ldots, \xi_K^0)$. Error estimates on this invariant probability measure can be obtained as in Section 2.3.3.1. The conclusion is that the bias is still of order $\mathcal{O}(\Delta t^2) + \mathcal{O}(\varepsilon(n)\Delta t)$, but with a smaller prefactor for the dominant

term $\varepsilon(n)\Delta t$. Let us emphasize that we do not require Assumption 1 to hold for this analysis. To make this precise, the first step is to write the generator as

$$\mathcal{L}_{\mathrm{eAdL},\Sigma_{\mathbf{x}}} = \mathcal{L}_{\mathrm{eAdL},\overline{\Sigma}^{\mathrm{K}}_{\mathbf{x}}} + \varepsilon(n)\Delta t \widetilde{\mathcal{L}}_{\Sigma_{\mathbf{x}}-\overline{\Sigma}^{K}_{\mathbf{x}}},$$

where $\widetilde{\mathcal{L}}_{\mathscr{M}}$ is defined in (2.58), and $\overline{\Sigma}^{K}_{\mathbf{x}}$ is a symmetric positive matrix which belongs to the vector space generated by $f_0, \ldots, f_K$. One can then state an inequality similar to (2.59), where $\overline{\nu}_{\xi^0}$ is replaced by the invariant probability measure $\overline{\nu}_{K,(\xi_0^0,\ldots,\xi_K^0)}$ of the eADL dynamics associated with the generator $\mathcal{L}_{\mathrm{eAdL},\overline{\Sigma}^{\mathrm{K}}_{\mathbf{x}}}$, and ν_{ξ^0} is replaced by invariant probability measure $\nu_{K,(\xi_0^0,\ldots,\xi_K^0)}$ of the eAdL dynamics with generator $\mathcal{L}_{\mathrm{eAdL},\Sigma_{\mathbf{x}}}$. The prefactor of the dominant error term, proportional to $\varepsilon(n)\Delta t$, depends on the function

$$f_{\mathrm{eAdL},\Sigma_{\mathbf{x}},\overline{\Sigma}^{K}_{\mathbf{x}}} = \left(-\mathcal{L}^{*}_{\mathrm{eAdL},\overline{\Sigma}^{K}_{\mathbf{x}}}\right)^{-1} \widetilde{\mathcal{L}}^{*}_{\Sigma_{\mathbf{x}}-\overline{\Sigma}^{K}_{\mathbf{x}}}\mathbf{1},$$

where adjoints are taken on $L^2(\overline{\nu}_{K,(\xi_0^0,\ldots,\xi_K^0)})$. Under appropriate conditions on the structure of $\overline{\nu}_{K,(\xi_0^0,\ldots,\xi_K^0)}$ and resolvent bounds for $\mathcal{L}_{\mathrm{eAdL},\overline{\Sigma}^{\mathrm{K}}_{\mathbf{x}}}$, similar to the ones stated in Lemma 2.10, it is possible to upper bound the norm of $f_{\mathrm{eAdL},\Sigma_{\mathbf{x}}}$ in $L^2(\overline{\nu}_{K,(\xi_0^0,\ldots,\xi_K^0)})$ by $\left\|\Sigma_{\mathbf{x}} - \overline{\Sigma}^{K}_{\mathbf{x}}\right\|_{L^2(\pi)}$. By optimizing upon the matrix valued function $\overline{\Sigma}^{K}_{\mathbf{x}}$, one can conclude that the prefactor of the error term proportional to $\varepsilon(n)\Delta t$ is bounded, up to a constant, by

$$\min_{\mathscr{M}_0,\ldots,\mathscr{M}_K \in \mathbb{R}^{d\times d}} \left\| \Sigma_{\mathbf{x}} - \sum_{k=0}^{K} \mathscr{M}_k f_k \right\|_{L^2(\pi)}, \tag{2.67}$$

which corresponds to the $L^2(\pi)$ projection of $\Sigma_{\mathbf{x}}$ onto the vector space of symmetric matrices generated by the basis. This shows that the better the approximation of the covariance $\Sigma_{\mathbf{x}}$ is for the chosen basis $(f_0, \ldots, f_K)$, the smaller the bias is.

Remark 2.14. *As for the numerical discretization of AdL (2.52) (see Remark 2.9), we can also use the numerical scheme (2.66) while considering each variable ξ_k as a scalar or a diagonal matrix. Observations similar to the ones made in Remark 2.11 apply here as well. More precisely, the prefactor of the error term proportional to $\varepsilon(n)\Delta t$ is bounded by the $L^2(\pi)$-projection of $\Sigma_{\mathbf{x}}$ onto: (i) the vector space generated by the basis when the variables ξ_k are scalars for each k, meaning that $\mathscr{M}_k = \sigma_k \mathrm{I}_d$ where $\sigma_k \in \mathbb{R}$ in (2.67); (ii) the vector space of diagonal matrices generated by the basis when the variables ξ_k are diagonal matrices for each k, meaning that $\mathscr{M}_0, \ldots, \mathscr{M}_K$ are diagonal matrices in (2.67). For the numerical simulations reported in Section 2.5, we will consider the variables ξ_k as full $d \times d$ symmetric matrices.*

2.4.3 Choice the basis functions

The choice of the basis functions $f_0, \ldots, f_K$ is a key point in our method, since there is a trade-off between a good approximation of the matrix $A_{\Delta t,n}(\theta)$ such that the condition (2.63) is satisfied, and the number of additional degrees of freedom (which is equal to the number of functions K introduced multiplied by $d(d+1)/2$ if the unknown are matrices). We typically want to introduce only a limited number of degrees of freedom. One option towards this goal is for instance to consider only isotropic matrices $\xi_k = \sigma_k \mathrm{I}_d$ with $\sigma_k \in \mathbb{R}$. Such a choice is advocated in [39] for $K = 0$ and $f_0 = \mathbf{1}$.

Spatial decomposition. Numerical experiments suggest that the covariance matrix of the estimator of the gradient may change rapidly in certain regions of the parameter space (see in particular Figure 2.6). A convenient approach to approximating this matrix is to partition the

domain Θ into $K+1$ subdomains, denoted by $\mathcal{D}_0, \dots, \mathcal{D}_K$, and to consider the functions f_k to be indicator functions of these domains, *i.e.* $f_k = \mathbf{1}_{\mathcal{D}_k}$. This corresponds to a piecewise constant approximation of the covariance matrix. If the domain is simple and the dimension d sufficiently small, one can think of simple geometric decompositions, using *e.g.* rectangles if Θ is itself rectangular, or rings around the various modes of the posterior distribution. For more complicated domains in possibly high dimension, it is possible to decompose the parameter space with a Voronoi tessellation, where the centers of the Voronoi tesselation are for instance the local maxima of the posterior distribution. Such points can be localized by performing preliminary SGLD or AdL runs. This amounts to considering a constant friction matrix in each mode of the probability distribution.

Polynomial approximation. If the domain Θ is sufficiently simple from a geometric viewpoint, a more sophisticated method is to couple a spatial decomposition with some spectral approximation, by introducing basis functions on each subdomain. For a rectangular decomposition of the domain, a polynomial basis can be defined on each subdomain

$$\mathcal{D}_k = [M_{k,1}^-, M_{k,1}^+] \times \dots \times [M_{k,d}^-, M_{k,d}^+], \tag{2.68}$$

by tensorization of monomial functions of the form

$$e_{k,j,i}(\theta_j) = \left(\frac{\theta_j - M_{k,j}^-}{M_{k,j}^+ - M_{k,j}^-} \right)^i. \tag{2.69}$$

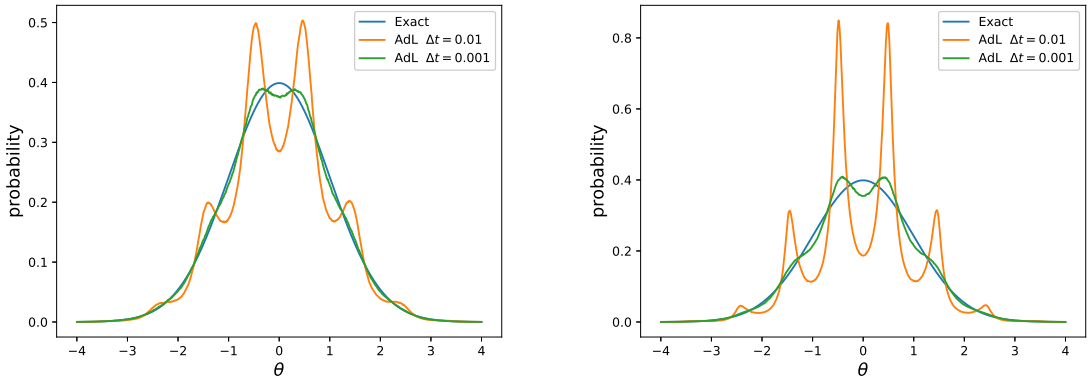
The integer k indexes the domain under consideration, j characterizes the degree of freedom under consideration, and i is the power of the monomial. The normalization we choose ensures that each of the term in the tensor product defining the basis function has values in $[0,1]$. Defining for instance the degree of the tensor product `deg` to be the maximum of the degree of the individual polynomials, the total number of degrees of freedom is given by $K(\texttt{deg}+1)^d$. When `deg` is large, the elementary polynomials (2.69) assume non negligible values only close to the boundary of the domain, which can result in numerical instabilities. We therefore rescale the basis functions so that their $L^2(\pi)$ norm is 1 (as the normalization factor being estimated using preliminary AdL runs).

2.5 Numerical illustrations

We illustrate in this section the interest of eAdL with numerical experiments showing that this dynamics allows to (almost fully) remove the mini-batching bias. We first demonstrate this in Section 2.5.1 for a one dimensional toy model where we artificially inject noise. We then consider in Section 2.5.2 the more realistic example of Section 2.2.5.2 where the elementary likelihoods are given by a mixture of Gaussians, and for which AdL fails to fully remove the bias because the covariance matrix is not constant. We demonstrate that with a reduced number function basis, the bias is significantly reduced even for the smallest values of n, including $n = 1$. We finally consider a model of logistic regression for MNIST data in Section 2.5.3, where we investigate why AdL with a scalar friction provides results of almost the same quality as AdL with a genuine friction matrix.

2.5.1 One dimensional toy model

We have seen in Section 2.3.3 that the covariance matrix $\Sigma_{\mathbf{x}}(\theta)$ is not always constant. We demonstrate here how the dependence of the covariance matrix on the parameters affects the posterior distribution and how eAdL corrects for the corresponding bias. To do so, we



(a) $\delta = 0.5$ (b) $\delta = 1.0$

Figure 2.8 – Sampling the Gaussian distribution using the scheme (2.52) for Adaptive Langevin dynamics and various values of δ. The solid blue line is the exact distribution.

consider a toy model where $\pi(\theta) = (2\pi)^{-1/2} \exp\left(-\theta^2/2\right)$, and we inject an artificial Gaussian noise, with a variance depending on θ as

$$\Sigma(\theta) = \alpha^2 \frac{1 + \delta \cos(2\pi\theta)}{2},$$

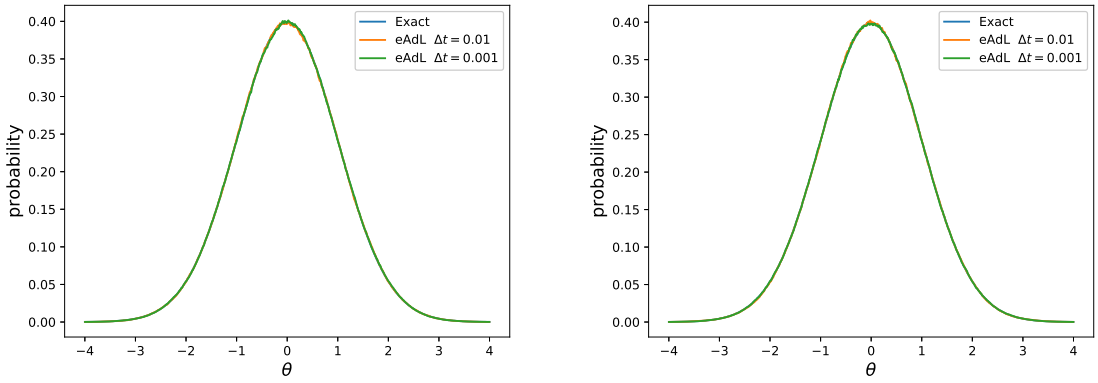
where $|\delta| \leqslant 1$ so that $\Sigma \geqslant 0$. We consider $\widehat{F_n}(\theta) = \theta + \sqrt{\Sigma(\theta)} G$ in (2.66), with G is standard Gaussian random variable. We run numerical experiments to see how the dependence of Σ on θ affects the posterior distribution. In this simple model, the parameter α^2 is the counterpart of $\varepsilon(n)$; while the parameter δ quantifies the difference between Σ and its average with respect to π (and hence should determine the bias for AdL at leading order in view of the analysis of Section 2.3.3.1).

We set $\alpha = 50$. We fist run the numerical scheme (2.52) corresponding to AdL (for which we fix $\Gamma = 1$ and $\eta = 1$) for a time $T = 10^6$ for various values of δ and Δt. We compare the resulting histograms of θ and the exact posterior distribution in Figure 2.8. It is clear that the variation of the covariance matrix affects the sampled probability distribution. The results of Figure 2.8 show that the bias is already significant for $\delta = 0.5$, and even larger for $\delta = 1$. The modes that appear in the posterior distribution are a trace of the modulations of $\Sigma(\theta)$ over the range of values of θ which are explored. As expected however, the bias decreases as Δt decreases. This is due to the fact that the magnitude of the noise depends on the stepsize, see Section 2.3.3.1, and (2.58) in particular.

We next run eAdL with the same parameters for the same values of δ. We choose $f_1(\theta) = 1$ and $f_2(\theta) = \cos(2\pi\theta)$. In this case the covariance matrix can be fully captured by the basis of functions. As expected, eAdL completely corrects for the bias introduced by the dependence of covariance matrix on the parameter θ in this simple example, even for $\delta = 1$; see Figure 2.9.

2.5.2 Mixture of Gaussians

We turn to the more realistic example introduced in Section 2.2.5.2. The covariance matrix in this case is not constant, and AdL fails to correct for the extra noise due to mini-batching; see Section 2.3.3.2 and Figure 2.7. We consider the same parameters as Section 2.3.3.2. To define the basis functions, we first consider a symmetric rectangular partition on the domain, with 4 meshes of the form (2.68), with $M^-_{0,1} = M^-_{0,2} = M^-_{1,2} = M^-_{2,1} = 0$, $M^+_{0,1} = M^+_{0,2} = M^-_{1,1} = M^+_{1,2} = M^+_{2,1} = M^-_{2,2} = M^-_{3,1} = M^-_{3,2} = 0.7$ and $M^+_{1,1} = M^+_{2,2} = M^+_{3,1} = M^+_{3,2} = 1.4$. We define polynomials on each mesh as discussed in Section 2.4.3. We run the numerical scheme (2.66)



(a) $\delta = 0.5$ (b) $\delta = 1.0$

Figure 2.9 – Sampling the Gaussian distribution using the scheme (2.66) for eAdL and various values of δ. The solid blue line is the exact distribution.

for eAdL with $\eta_k = 1$ for all values of $0 \leqslant k \leqslant K$, for a final time $T = 10^6$, various values of Δt, and various degrees of polynomials. The case $K = 0$ corresponds to standard AdL, while $K = 3$ corresponds to an estimation of the covariance matrix by a piecewise constant matrix valued function on the 4 domains under consideration, $K = 15$ to products of affine functions in each degree of freedom on the 4 domains, and $K = 35$ to products of second order polynomials in each variable on each domain. We report in Figure 2.10 the L^1 error on the marginal distribution over θ_1 of the posterior distribution sampled by the numerical scheme for various values of K. Let us first emphasize that the bias has been dramatically decreased compared to AdL even for the smallest value $K = 3$ (compare with Figure 2.7). A small bias is however remaining, due to the fact that the covariance matrix cannot be fully approximated. This residual bias can be further reduced when K is increased. Already for $K = 15$, the bias is very low and only arises from the time step error. We plot in Figure 2.11 the quantity $\|\Sigma_{\mathbf{x}} - S_K\|_{L^2(\pi)}$ where S_K is the $L^2(\pi)$ projection of $\Sigma_{\mathbf{x}}$ onto the vector space of symmetric matrices generated by the basis for each value of K. The results are consistent with the bias analysis of Section 2.4.2 since the decay of the approximation of $\Sigma_{\mathbf{x}}$ is similar to the decay of the bias with respect to K.

2.5.3 Logistic regression

In this section, we consider the example of a Bayesian logistic regression model, where we train the model on a subset of the MNIST data set containing the digits 7 and 9, and which have been pre-processed by a principal component analysis, as described in [90, Section 4.3].

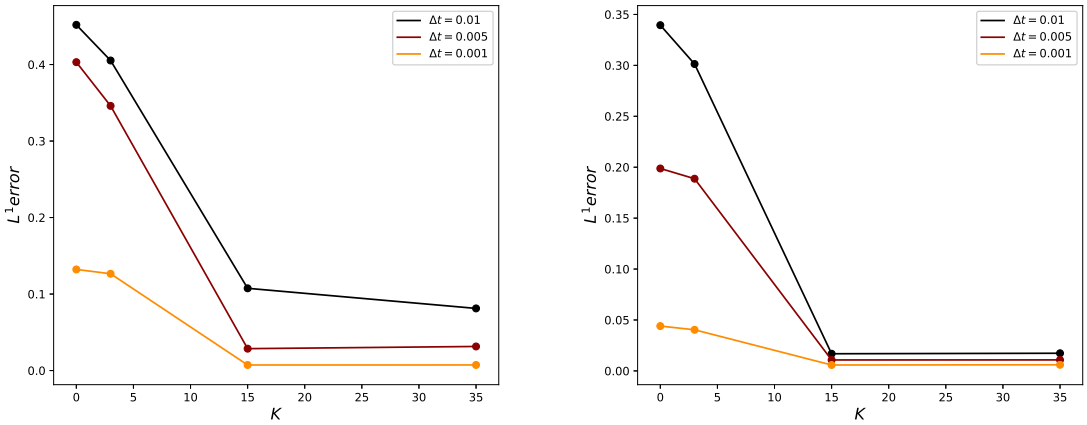
Presentation of the model. We consider $N_{\text{data}} = 12,251$ images $\mathbf{z} = (z_1, z_2, ..., z_{N_{\text{data}}}) \subset \mathbb{R}^d$ (with $d = 100$), labeled by $\mathbf{y} = (y_1, y_2, ..., y_{N_{\text{data}}}) \subset \{0, 1\}$ (the labels 0 and 1 respectively correspond to digits 7 and 9). We assume that the the elementary likelihood on the data point $x_i = (z_i, y_i)$ is

$$P_{\text{elem}}(x_i|\theta) = \sigma(\theta^T z_i)^{y_i}(1 - \sigma(\theta^T z_i))^{1-y_i},$$

where

$$\sigma(z) = \frac{\exp(z)}{1 + \exp(z)},$$

and $\theta \in \mathbb{R}^d$ is the vector of parameters we want to estimate. Assuming that the prior over the vector of parameter θ is a centered standard normal distribution, a simple computation



(a) $n = 5$ (b) $n = 15$

Figure 2.10 – L^1 error on the the marginal over θ_1 of the a posteriori distribution with eAdL for various values of K and Δt ($K = 0$ corresponds to AdL results).

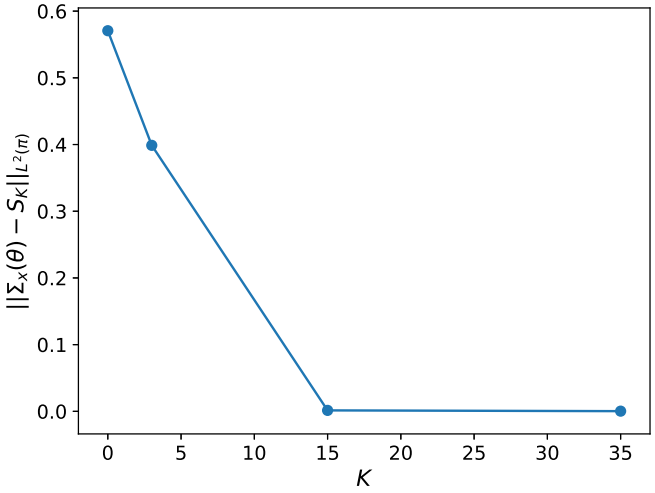
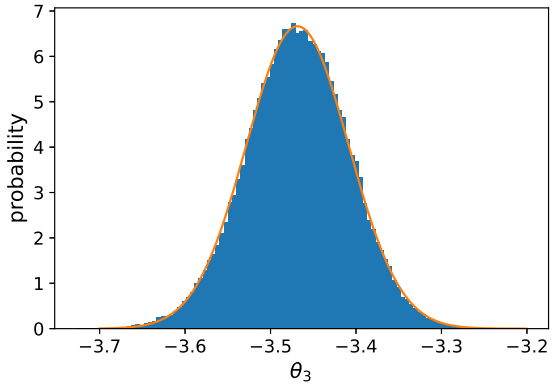
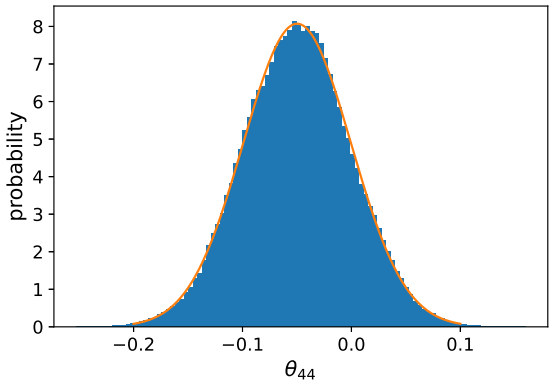


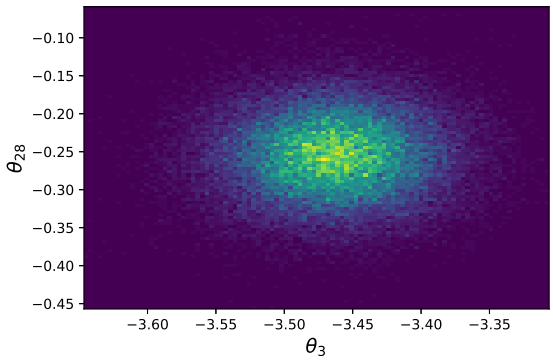
Figure 2.11 – $\|\Sigma_{\mathbf{x}} - S_K\|_{L^2(\pi)}$ where S_K is the $L^2(\pi)$ projection of $\Sigma_{\mathbf{x}}$ onto the vector space of symmetric matrices generated by the basis for each value of K.



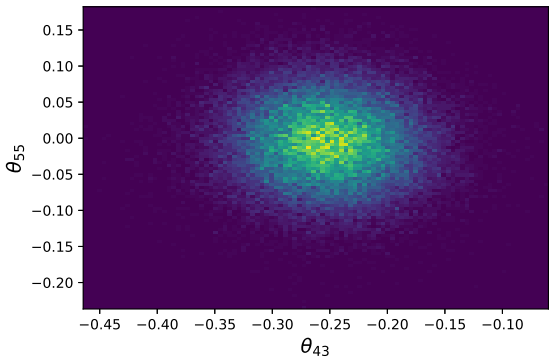
(a) Marginal distribution of θ_3



(b) Marginal distribution of θ_{44}



(c) Joint marginal distribution of (θ_3, θ_{29})



(d) Joint marginal distribution of (θ_{43}, θ_{55})

Figure 2.12 – Marginal posterior distributions of θ_3 and θ_{44} (top), and of (θ_3, θ_{28}) and (θ_{43}, θ_{55}) (bottom).

based on the identity $\sigma'(z) = \sigma(z)(1 - \sigma(z))$ shows that

$$\widehat{F}_n(\theta) = \theta + \frac{N_{\text{data}}}{n} \sum_{i \in I_n} (y_i - \sigma(\theta^T z_i)) z_i.$$

Numerical results. We use AdL to sample from the posterior distribution rather than eAdL since d is large, and the decomposition of the covariance in a basis of functions matrix would involve too many unknowns. We fix $\gamma = \eta = 1$, $T = 10^3$ and $\Delta t = 10^{-3}$. We run the numerical scheme (2.52) for the following three cases: ξ is a scalar, a diagonal matrix or a full matrix. We plot the marginal distributions of some parameters θ_i in Figure 2.12. The results suggest that the posterior distribution of the parameters is close to a Gaussian distribution. As pointed out in Section 2.3.2.3, AdL would be sufficient in this case to remove the minibatching error.

According to Lemma 2.7, the posterior distribtion of the variable ξ is centered on $A_{\Delta t,n}$ given by (2.39), when the latter matrix is constant. An estimate of the average covariance matrix can then be obtained with the estimator

$$\frac{1}{\varepsilon(n)\Delta t} \left(\frac{1}{N_{\text{iter}}} \sum_{m=1}^{N_{\text{iter}}} \xi^m - \gamma \mathrm{I}_d \right),$$

where $\gamma \mathrm{I}_d$ is replaced by γ in the scalar case. We plot in Figure 2.13 the diagonal components of the estimate of the average of the covariance matrix of the stochastic gradient for the following three cases considered here (ξ scalar, diagonal of full matrix). We can see that $\overline{\Sigma}_{\mathbf{x}}$ has a dominant part which is equal to $\sigma \mathrm{I}_d$ (although there are many non zero off diagonal

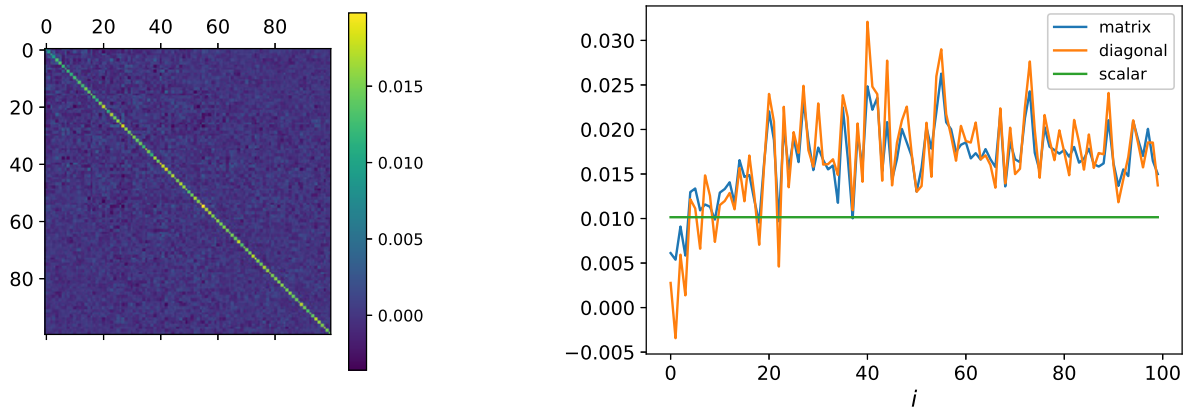


Figure 2.13 – Left: Estimated covariance matrix of θ. Right: Diagonal components of the estimate of the average covariance matrix of the stochastic gradient as a function of the index $1 \leqslant i \leqslant d$.

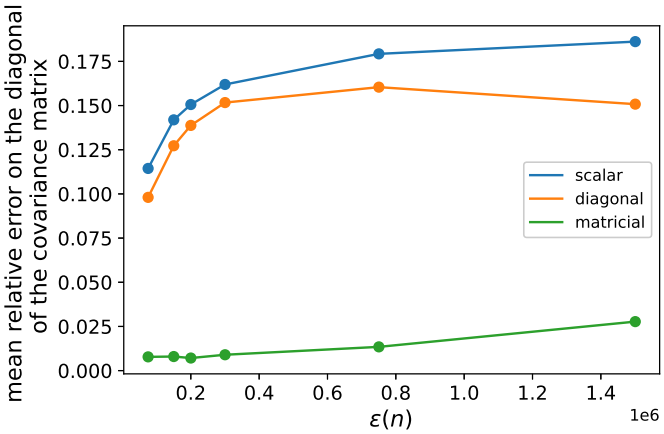


Figure 2.14 – Error on the covariance matrix of the parameters using AdL for ξ scalar, diagonal matrix or full matrix.

terms). Following Remark 2.11, we expect AdL to give comparable results whether the friction variable is a scalar, a diagonal matrix and a full matrix. In order to investigate this, we plot in Figure 2.14 the mean relative error on the diagonal entries of the covariance matrix of the vector of parameters θ with respect to $\varepsilon(n)$ given by

$$\frac{1}{d}\sum_{i=1}^{d}\left|\frac{[\mathrm{cov}(\theta_n)]_{i,i}-[\mathrm{cov}(\theta_{N_{\mathrm{data}}})]_{i,i}}{[\mathrm{cov}(\theta_{N_{\mathrm{data}}})]_{i,i}}\right|,$$

where θ_n denotes the vector of parameters obtained by a trajectory average over a realization of AdL while using the minibatching procedure, while $\theta_{N_{\mathrm{data}}}$ denote the reference vector of parameters obtained by AdL without minibatching. The error on the covariance matrix is some measure of the bias on the posterior distribution effectively sampled by the numerical scheme. The results confirm that AdL with scalar ξ already leads to acceptable results in this example and performs as well as AdL with diagonal ξ.

2.6 Discussion and perspectives

We quantified in this work the bias on the posterior distribution arising from mini-batching for (kinetic or underdamped) Langevin dynamics, both for standard and adaptive Langevin dynamics. As highlighted in (2.1), the bias is dictated by the quality of the approximation of the covariance matrix $\Sigma_{\mathbf{x}}(\theta)$ of the gradient estimator by the average (at θ fixed) of the extra friction variable ξ introduced in AdL. The latter average is independent of the parameter θ in the original formulation of AdL, and this is sufficient to remove the bias if the target distribution is Gaussian, and allows to strongly reduce the bias for posteriors close to Gaussian (as for the MNIST example considered in Section 2.5.3). The latter situation is typical when the number of data points N_{data} is large, as the posterior distribution concentrates in a Gaussian manner around one mode (up to symmetries) due to the Bernstein–von Mises theorem (see [150, Section 10.2]). In this case, it is likely that AdL already captures most of the mini-batching bias.

Standard AdL is however not sufficient in various situations, in particular when $\Sigma_{\mathbf{x}}$ genuinely depends on θ, which is the case when N_{data} is relatively small. In this case, the posterior distribution can strongly deviate from a Gaussian distribution. Extended AdL provides a framework to systematically reduce the mini-batching bias in such situations, by adding extra degrees of freedom to better tune the friction variable. Striking reductions in the bias can be observed already for not too many additional degrees of freedom, even for the extreme situation where the batch size is set to its minimal value $n = 1$. In any case, eAdL always leads to a smaller bias than AdL.

Let us also emphasize here that AdL and eAdL can be used in conjunction with other techniques to reduce the variance of the stochastic estimator of the force, such as control variate or extrapolation techniques (see for instance [110, 24, 9, 46]). AdL and eAdL should therefore not be seen as alternatives to these techniques, but as complements.

The choice of the dimensionality of the vector space for the friction variable is a key element for standard AdL and its extension. Going from scalar to matrix valued friction increases the number of degrees of freedom from 1 to d^2. When eAdL is employed, with K basis functions (for instance indicator functions of some regions in parameter space), the number of degrees of freedom is further multiplied by K. One would like the number of degrees of freedom to be as small as possible. From a theoretical perspective, this motivates further characterizing the structure of the covariance matrix $\Sigma_{\mathbf{x}}$, which have been observed to be low rank in certain situations related to neural network training [30]. This could also shed some light on the current active research efforts on understanding the so-called implicit bias of neural network training.

Appendix

2.A Proof of some technical estimates

We provide in this appendix the algebraic manipulations for various error estimates. To simplify the notation, we simply write Z in all this section for the random variable which appears in (2.13). The analysis we perform is asymptotic since it relies on the assumption that $\varepsilon(n) \geqslant 1$ and $\Delta t \leqslant 1$ are such that $\varepsilon(n)\Delta t \leqslant 1$ is sufficiently small.

2.A.a Proof of (2.18)

Let us prove that the evolution operator for SGLD satisfies (2.18) for Δt and $\varepsilon(n)\Delta t$ small. We rewrite to this end the SGLD scheme (2.17) as

$$\theta^{m+1} = \Phi_{\Delta t,n}(\theta^m, Z^m, G^m),$$

with

$$\Phi_{\Delta t,n}(\theta, Z, G) = \theta + \sqrt{\varepsilon(n)}\Delta t \Sigma_{\mathbf{x}}^{1/2}(\theta) Z + \sqrt{2\Delta t} G + \Delta t \nabla_\theta (\log \pi(\theta|\mathbf{x}).$$

For a given smooth function $\theta \mapsto \varphi(\theta)$ with compact support, it then holds

$$\begin{aligned}
\varphi(\Phi_{\Delta t,n}(\theta^m, Z^m, G^m)) &= \varphi\left(\Phi_{\Delta t,n}(\theta^m, 0, G^m) + \sqrt{\varepsilon(n)}\Delta t \Sigma_{\mathbf{x}}^{1/2}(\theta^m) Z^m\right) \\
&= \varphi(\Phi_{\Delta t,n}(\theta^m, 0, G^m)) + \sqrt{\varepsilon(n)}\Delta t \Sigma_{\mathbf{x}}^{1/2}(\theta^m) Z^m \cdot (\nabla_\theta \varphi)(\Phi_{\Delta t,n}(\theta^m, 0, G^m)) \\
&\quad + \frac{1}{2}\varepsilon(n)\Delta t^2 (\nabla_\theta^2 \varphi)(\Phi_{\Delta t,n}(\theta^m, 0, G^m)) : \Sigma_{\mathbf{x}}^{1/2}(\theta^m) Z^m \otimes \Sigma_{\mathbf{x}}^{1/2}(\theta^m) Z^m \\
&\quad + \frac{1}{6}\varepsilon(n)^{3/2}\Delta t^3 D^3 \varphi(\Phi_{\Delta t,n}(\theta^m, 0, G^m)) : \left(\Sigma_{\mathbf{x}}^{1/2}(\theta^m) Z^m\right)^{\otimes 3} \\
&\quad + \mathcal{O}(\varepsilon(n)^2 \Delta t^4),
\end{aligned}$$

where we use the notation

$$D^3\varphi(\theta) : v_1 \otimes v_2 \otimes v_3 = \sum_{i,j,k=1}^{d} [v_1]_i [v_2]_j [v_3]_k (\partial^3_{\theta_i,\theta_j,\theta_k} \varphi)(\theta),$$

and $v^{\otimes 3} = v \otimes v \otimes v$. Using classical results on the evolution operator of the Euler–Maruyama scheme to compute the expectation of $\varphi(\Phi_{\Delta t,n}(\theta^m, 0, G^m))$ with respect to G^m (see for example [154] for details), we can compute the expectation of $\varphi(\Phi_{\Delta t,n}(\theta^m, Z^m, G^m))$ with respect to the variables G^m and Z^m as

$$\begin{aligned}
\left(\widehat{P}_{\Delta t,n}\varphi\right)(\theta^m) &= \mathrm{I}_d + \Delta t \mathcal{L}_{\mathrm{ovd}}\phi(\theta^m) + \Delta t^2 \left(\frac{1}{2}\mathcal{L}_{\mathrm{ovd}}^2 + \mathcal{A}_{\mathrm{disc}}\right)\varphi(\theta^m) + \mathcal{O}(\Delta t^3) \\
&\quad + \Delta t^2 \varepsilon(n) \mathbb{E}_G\left[(\mathcal{A}_{\mathrm{mb}}\varphi)(\Phi_{\Delta t,n}(\theta^m, 0, G))\right] + \mathcal{O}\left(\varepsilon(n)^{3/2}\Delta t^3\right).
\end{aligned}$$

Since $\mathbb{E}_G\left[\mathcal{A}_{\mathrm{mb}}\varphi(\Phi_{\Delta t,n}(\theta^m, 0, G))\right] = \mathcal{A}_{\mathrm{mb}}\varphi(\theta^m) + \mathcal{O}(\Delta t)$, we obtain on the one hand that

$$\widehat{P}_{\Delta t,n} = \mathrm{I}_d + \Delta t \mathcal{L}_{\mathrm{ovd}} + \Delta t^2 \left(\frac{1}{2}\mathcal{L}_{\mathrm{ovd}}^2 + \mathcal{A}_{\mathrm{disc}} + \varepsilon(n)\mathcal{A}_{\mathrm{mb}}\right) + \mathcal{O}\left((1+\varepsilon(n)^{3/2})\Delta t^3\right). \tag{2.70}$$

On the other hand, notice that

$$\begin{aligned}
\mathrm{e}^{\Delta t(\mathcal{L}_{\mathrm{ovd}} + \varepsilon(n)\Delta t \mathcal{A}_{\mathrm{mb}} + \Delta t \mathcal{A}_{\mathrm{disc}})} &= \mathrm{I}_d + \Delta t \mathcal{L}_{\mathrm{ovd}} + \Delta t^2 \left(\frac{1}{2}\mathcal{L}_{\mathrm{ovd}}^2 + \mathcal{A}_{\mathrm{disc}} + \varepsilon(n)\mathcal{A}_{\mathrm{mb}}\right) \\
&\quad + \mathcal{O}((1+\varepsilon(n))\Delta t^3).
\end{aligned} \tag{2.71}$$

Using (2.70) alongside with (2.71), we deduce the desired result (2.18).

Remark 2.15. *Note that when $\mathbb{E}[Z^3] = 0$, the remainder term is in fact of order $\mathcal{O}\left((1+\varepsilon(n)^2\Delta t)\Delta t^3\right)$ in (2.70). This leads to a remainder of order $\mathcal{O}\left((1+\varepsilon(n))\Delta t^3\right)$ in (2.18).*

2.A.b Proof of (2.28), (2.29) and (2.30)

To prove (2.28), we consider a smooth function $(\theta, p) \mapsto \varphi(\theta, p)$ with compact support, and rewrite the operator as

$$\begin{aligned}\left(Q_{\Delta t}^{\mathcal{L}_1}\varphi\right)(\theta^m, p^m) &= \mathbb{E}\left[\varphi\left(\theta^m, p^m + \Delta t\widehat{F_n}(\theta^m)\right)\right] \\ &= \mathbb{E}\left[\varphi\left(\theta^m, p^m + \Delta t\nabla_\theta(\log\pi(\theta^m|\mathbf{x})) + \sqrt{\varepsilon(n)}\Delta t\Sigma_{\mathbf{x}}^{1/2}(\theta^m)Z^m\right)\right].\end{aligned}$$

We then have

$$\begin{aligned}\left(Q_{\Delta t}^{\mathcal{L}_1}\varphi\right)(\theta^m, p^m) = \mathbb{E}\left[\varphi\left(\theta^m, p^m + \Delta t\nabla_\theta(\log\pi(\theta^m|\mathbf{x}))\right)\right] \\ + \sqrt{\varepsilon(n)}\Delta t\mathbb{E}\left[\Sigma_{\mathbf{x}}^{1/2}(\theta^m)Z^m \cdot \nabla_p\varphi(\theta^m, p^m + \nabla_\theta(\log\pi(\theta^m|\mathbf{x}))\right] \\ + \frac{1}{2}\varepsilon(n)\Delta t^2\mathbb{E}\left[\nabla_p^2\varphi(\theta^m, p^m + \Delta t\nabla_\theta(\log\pi(\theta^m|\mathbf{x})) : \Sigma_{\mathbf{x}}^{1/2}(\theta^m)Z^m \otimes \Sigma_{\mathbf{x}}^{1/2}(\theta^m)Z^m\right] \\ + \mathcal{O}\left(\varepsilon(n)^{3/2}\Delta t^3\right),\end{aligned}$$

so that

$$\begin{aligned}\left(Q_{\Delta t}^{\mathcal{L}_1}\varphi\right)(\theta^m, p^m) &= \left(\mathrm{e}^{\Delta t\mathcal{L}_1}\varphi\right)(\theta^m, p^m) + \varepsilon(n)\Delta t^2(\mathcal{A}_{\mathrm{lan}}\varphi)(\theta^m, p^m + \Delta t\nabla_\theta(\log\pi(\theta^m|\mathbf{x}))) \\ &\quad + \mathcal{O}\left(\varepsilon(n)^{3/2}\Delta t^3\right) \\ &= \left(\mathrm{e}^{\Delta t\mathcal{L}_1}\varphi\right)(\theta^m, p^m) + \varepsilon(n)\Delta t^2\left(\mathcal{A}_{\mathrm{lan}}\varphi\right)(\theta^m, p^m) + \mathcal{O}\left(\varepsilon(n)^{3/2}\Delta t^3\right),\end{aligned} \tag{2.72}$$

from which the result directly follows.

Remark 2.16. *If $\mathbb{E}[Z^3] = 0$, the error term $\mathcal{O}\left(\varepsilon(n)^{3/2}\Delta t^3\right)$ can be replaced by $\mathcal{O}\left(\varepsilon(n)\Delta t^3\right)$.*

To prove (2.29), we use the Baker–Campbell–Hausdorff formula on the operator of the numerical scheme (2.27):

$$\begin{aligned}\widehat{P}_{\Delta t,n} &= \mathrm{e}^{\Delta t\mathcal{L}_3/2}\mathrm{e}^{\Delta t\mathcal{L}_2/2}Q_{\Delta t}^{\mathcal{L}_1}\mathrm{e}^{\Delta t\mathcal{L}_2/2}\mathrm{e}^{\Delta t\mathcal{L}_3/2} \\ &= \mathrm{e}^{\Delta t\mathcal{L}_3/2}\mathrm{e}^{\Delta t\mathcal{L}_2/2}\mathrm{e}^{\Delta t\mathcal{L}_1}\mathrm{e}^{\Delta t\mathcal{L}_2/2}\mathrm{e}^{\Delta t\mathcal{L}_3/2} + \varepsilon(n)\Delta t^2\mathrm{e}^{\Delta t\mathcal{L}_3/2}\mathrm{e}^{\Delta t\mathcal{L}_2/2}\mathcal{A}_{\mathrm{lan}}\mathrm{e}^{\Delta t\mathcal{L}_2/2}\mathrm{e}^{\Delta t\mathcal{L}_3/2} \\ &\quad + \mathcal{O}\left(\varepsilon(n)^{3/2}\Delta t^3\right) \\ &= \mathrm{e}^{\Delta t\mathcal{L}_{\mathrm{lan}}} + \mathcal{O}\left((\Delta t + \varepsilon(n))\Delta t^2\right).\end{aligned}$$

Using computations similar to the ones leading to (2.18), we can prove the following estimate on $Q_{\Delta t}^{\mathcal{L}_1}$:

$$\mathrm{e}^{\Delta t(\mathcal{L}_1 + \varepsilon(n)\Delta t\mathcal{A}_{\mathrm{lan}})} = Q_{\Delta t}^{\mathcal{L}_1} + \mathcal{O}\left((1 + \varepsilon(n)^{3/2})\Delta t^3\right). \tag{2.73}$$

The desired result (2.30) follows by using the BCH formula.

2.A.c Proof of (2.55)

The evolution operator of the numerical scheme of AdL is given by

$$\widehat{P}_{\Delta t,n} = \mathrm{e}^{\Delta t\mathcal{L}_4/2}\mathrm{e}^{\Delta t\mathcal{L}_3/2}\mathrm{e}^{\Delta t\mathcal{L}_2/2}Q_{\Delta t}^{\mathcal{L}_1}\mathrm{e}^{\Delta t\mathcal{L}_2/2}\mathrm{e}^{\Delta t\mathcal{L}_3/2}\mathrm{e}^{\Delta t\mathcal{L}_3/2},$$

where $\mathcal{L}_1 = \nabla_\theta \log(\pi(\cdot|\mathbf{x}))^T\nabla_p$ and $\mathcal{L}_2$ are respectively the generators of (2.49) when $\varepsilon(n) = 0$ and (2.47) (which coincide with the operators in (2.23)), while $\mathcal{L}_3$ and $\mathcal{L}_4$ are the generators

of (2.50) and (2.48). Replacing $Q_{\Delta t}^{\mathcal{L}_1}$ by its expression in (2.72) and using the BCH formula, we obtain, by computations similar to the ones leading to (2.73):

$$\begin{aligned}\widehat{P}_{\Delta t,n}\varphi &= \mathrm{e}^{\Delta t\mathcal{L}_4/2}\mathrm{e}^{\Delta t\mathcal{L}_3/2}\mathrm{e}^{\Delta t\mathcal{L}_2/2}\mathrm{e}^{\Delta t\mathcal{L}_1}\mathrm{e}^{\Delta t\mathcal{L}_2/2}\mathrm{e}^{\Delta t\mathcal{L}_3/2}\mathrm{e}^{\Delta t\mathcal{L}_3/2} + \varepsilon(n)\Delta t^2\mathcal{A}_{\mathrm{lan}}\varphi + \mathcal{O}\left(\varepsilon(n)^{3/2}\Delta t^3\right)\\ &= \mathrm{e}^{\Delta t(\mathcal{L}_1+\mathcal{L}_2+\mathcal{L}_3+\mathcal{L}_4)}\varphi + \varepsilon(n)\Delta t^2\mathcal{A}_{\mathrm{lan}}\varphi + \mathcal{O}\left((1+\varepsilon(n)^{3/2})\Delta t^3\right)\\ &= \mathrm{e}^{\Delta t(\mathcal{L}_1+\mathcal{L}_2+\mathcal{L}_3+\mathcal{L}_4+\varepsilon(n)\Delta t\mathcal{A}_{\mathrm{lan}})}\varphi + \mathcal{O}\left(\varepsilon(n)\Delta t^3\right) + \mathcal{O}\left((1+\varepsilon(n)^{3/2})\Delta t^3\right)\\ &= \mathrm{e}^{\Delta t\mathcal{L}_{\mathrm{AdL},\Sigma_{\mathbf{x}}}}\varphi + \mathcal{O}\left((1+\varepsilon(n)^{3/2})\Delta t^3\right).\end{aligned}$$

When $\mathbb{E}[Z^3]=0$, the error term becomes $\mathcal{O}\left((1+\varepsilon(n))\Delta t^3\right)$.

2.B Unbiasedness of the mean for Langevin dynamics with mini-batching and Gaussian posterior

We prove that, in the case of one dimensional Gaussian likelihoods, there is no bias on the mean of the posterior distribution when using the discretization of Langevin dynamics (2.27) (see Section 2.2.5.1). We start by rewriting the numerical scheme as

$$\begin{pmatrix}\theta^{m+1}\\ p^{m+1}\end{pmatrix} = M_1\begin{pmatrix}\theta^m\\ p^m\end{pmatrix} + (1-\alpha_{\Delta t})^{1/2}M_2\begin{pmatrix}G^m\\ G^{m+1/2}\end{pmatrix} + V_3^m,$$

where

$$M_1 = \begin{pmatrix}1-a\dfrac{\Delta t^2}{2} & \alpha_{\Delta t/2}\left(\Delta t - a\dfrac{\Delta t^3}{4}\right)\\ -a\Delta t\alpha_{\Delta t/2} & \alpha_{\Delta t}\left(1-a\dfrac{\Delta t^2}{2}\right)\end{pmatrix},\quad M_2 = \begin{pmatrix}\Delta t\left(1-a\dfrac{\Delta t^2}{4}\right) & 0\\ \alpha_{\Delta t/2}\left(1-a\dfrac{\Delta t^2}{2}\right) & 1\end{pmatrix},\quad V_3^m = \begin{pmatrix}\dfrac{\Delta t^2}{2}b^m\\ \alpha_{\Delta t/2}\Delta t b^m\end{pmatrix},$$

and

$$a = \frac{1}{\sigma_\theta^2} + \frac{N_{\mathrm{data}}}{\sigma_x^2},\qquad b^m = \frac{N_{\mathrm{data}}}{n}\sum_{i\in I_n^m}\frac{x_i}{\sigma_x^2}.$$

Since

$$\mathbb{E}[b^m] = \frac{1}{\sigma_x^2}\sum_{i=1}^{N_{\mathrm{data}}} x_i := b,$$

we obtain, by first taking expectations with respect to G^m and $G^{m+1/2}$, and then with respect to realizations of I_n^m in b^m:

$$\mathbb{E}\left[\begin{pmatrix}\theta^m\\ p^m\end{pmatrix}\right] = M_1^m\mathbb{E}\left[\begin{pmatrix}\theta^0\\ p^0\end{pmatrix}\right] + \sum_{j=0}^{m-1}M_1^jV_3, \tag{2.74}$$

with $V_2 = (\Delta t^2 b/2, \alpha_{\Delta t/2}\Delta t b)^T$. We note that (recalling that, for the one dimensional case considered here, the friction $\Gamma>0$ is a scalar)

$$M_1 = \mathrm{I}_d - \Delta t\begin{pmatrix}0 & -1\\ a & \Gamma\end{pmatrix} + \mathcal{O}(\Delta t^2),$$

which shows that the eigenvalues of M_1 have real parts which are strictly smaller than 1 provided Δt is sufficiently small. Therefore, $M_1^m\to 0$ as $m\to+\infty$, and the matrix $\mathrm{I}_d - M_1$ is invertible. Moreover, a simple computation shows that

$$(\mathrm{I}_d - M_1)\begin{pmatrix}b/a\\ 0\end{pmatrix} = V_3,$$

so that

$$(\mathrm{I}_d - M_1)^{-1} V_3 = \begin{pmatrix} b/a \\ 0 \end{pmatrix}.$$

By letting m go to infinity in (2.74), and using the above equality, we finally conclude that

$$\lim_{m \to +\infty} \mathbb{E}\left[\begin{pmatrix} \theta^m \\ p^m \end{pmatrix} \right] = \begin{pmatrix} b/a \\ 0 \end{pmatrix} = \begin{pmatrix} \mu_{\mathrm{post}} \\ 0 \end{pmatrix},$$

which shows that the mean posterior distribution is unbiased.

CHAPTER 3

MINIBATCHING ERROR FOR BAYESIAN NEURAL NETWORKS

Contents

This work started during a research visit to Edinburgh, where I worked with Ben Leimkuhler and Tiffany Vlaar. We present in this chapter the preliminary results of this work on which we are currently still working.

Bayesian neural networks acquired more attention with the development of scalable sampling methods. MCMC methods give some guarantees when sampling from the posterior distribution. However, using minibatching introduces some extra bias on the posterior distribution, determined by the covariance matrix of the stochastic estimator of the gradient. Numerical analysis of this covariance matrix for BNNs in some numerical toy models suggests that it is of low rank. This opens the possibility to develop new scalable methods based on the AdL framework that reduce the minibatching error.

3.1 Introduction

Using neural networks [111, 51] for supervised learning tasks has attracted a lot of attention due to the performance of neural networks for a wide range of applications, for example learning to play Go or chess [140], medical diagnosis [21] or generating new data based on the Bayesian formalism using e.g. variational autoencoders [71, 124]. The mathematical and theoretical understanding of the performance of neural networks is still elusive. The loss function associated with these models is high dimensional and non convex. Variants of stochastic gradient descent are generally used to optimize the loss function, see for example [32] for an analysis of the convergence properties of such methods. These algorithms have shown great performance to optimize the loss function but an analysis of the generalization properties of neural networks trained by this approach is lacking. Some works have focused on the properties of the loss function, see for example [68]. More recent works have focused on the properties of the minima reached by stochastic descent gradient algorithm, see for example [120] where the authors use the concept of implicit bias to characterize the minimum reached by SGD in the case of diagonal linear networks. We also refer to references in [120] for works on the generalization properties of SGD.

Bayesian Neural Networks (BNNs) [111, 51, 67, 59] have emerged as one way to quantify the uncertainty associated with the predictions and avoid overfitting. Some stochasticity in artificial neural networks is incorporated either in the parameters (considered as random variables) or in the use of stochastic activation functions [158]. We focus in this work on stochastic parameters. For example, a ReLU neural network trained to classify dogs and cats can happen to (erroneously) classify a human as a dog with a high probability. This phenomenon corresponds to what is known as overconfident networks. In [60], the authors show that ReLU neural networks are necessarily overconfident for out of the distribution points *i.e.* data points that are not close to elements in the training set. They propose a new optimization technique which enforces low confidence predictions far away from the training data. In [77], they argue that a Bayesian approach, even for the last layer only, is sufficient to prevent overconfident ReLU neural networks. Bayesian Neural Networks are also naturally adapted for online learning (which corresponds to the situation when the data is added sequentially to the training set, so the predictor needs to be updated) or if the size of the data set is moderately large. The use of a prior distribution can seem as a disadvantage of BNNs, forcing some "random" prior knowledge on the parameters. However one can argue that it provides a way to understand many regularization techniques already used in optimization and which have been proved to increase the performance.

Sampling from the exact posterior distribution of the neural networks' parameters is particularly challenging due to the high dimensionality and non convexity of the distribution [64]. Scalable methods have been developed. One can use scalable MCMC methods based on SGLD-like algorithms [157, 98, 76, 95]. Variational inference methods [17] have also been adapted in the context of sampling the parameters of neural networks to approximate the posterior distribution by optimizing the *evidence lower bound* (ELBO), see [18]. One can also cite deep ensembles [79] and the Laplace approximation [72, 125] as methods relying on the Bayesian framework for neural networks. Scalable methods generally induce a bias on the posterior distribution. In [64], the authors used Hamiltonian Monte Carlo without minibatching to sample from the exact posterior, even if this method is not practical in real life examples. They numerically proved that BNNs allow to achieve better performance than classical training algorithms when the posterior is sampled exactly. This motivates the development of scalable Bayesian methods with low bias for sampling parameters of neural networks.

The analysis of Chapter 2 shows that the covariance matrix of the stochastic estimator of the force plays a central role in Langevin-like algorithms. In this chapter, we address the following points:

- Understanding the structure of the covariance matrix of the stochastic estimator of the gradient of the posterior distribution of parameters for neural networks (rank, sparsity, etc.). Sparsity, suggested already by [30], can lead to efficient approximations of the covariance matrix and then development of efficient Bayesian methods allowing to reduce the bias on the posterior distribution.
- Using AdL instead of computing full gradient as in [64] to reduce the bias on the posterior distribution while keeping a reasonable computational time.

3.2 Presentation of the models

In this section, we first recall in Section 3.2.1 the mathematical framework behind BNNs. We then present in Section 3.2.2 the numerical toy models used for the numerical illustrations. Finally, we introduce in Section 3.2.3 the architecture of the neural network used for the computations.

3.2.1 Mathematical framework

We first recall the notation of Section 1.1.2 (see the latter section for more details). Let $\mathbf{x} = (x_1, ..., x_{N_{\text{data}}}) \in \mathcal{X}^{N_{\text{data}}}$ be the set of input data and $\mathbf{y} = (y_1, ..., y_{N_{\text{data}}}) \in \mathcal{Y}^{N_{\text{data}}} = \{0,1\}^{N_{\text{data}}}$ their corresponding labels (we only consider binary classification problems in this chapter for simplicity of exposition, bur the approach can be adapted to other situations). In machine learning, the goal of supervised learning is, given the set of pairs (x_i, y_i) where $i \in \{1, ..., N_{\text{data}}\}$, to predict an output y from a new given input x. A neural network defines a map $N_\theta : \mathcal{X} \to \mathcal{Y}$, where $\theta \in \Theta = \mathbb{R}^d$ represent the parameters of the NN (weights and biases). The classical setting consists in finding the best θ^* (the one that minimize the loss function, which corresponds to a maximum likelihood parameter) using optimization algorithms. The predictions are then computed as

$$y = \begin{cases} 1 & \text{if} \quad N_{\theta^*}(x) \geqslant 1/2, \\ 0 & \text{if} \quad N_{\theta^*}(x) < 1/2. \end{cases} \tag{3.1}$$

Recall that the loss function used for binary classification problems for a given pair (x, y) is (see Equation (1.3) in Section 1.1.2)

$$L(x, y, \theta) = y \log(N_\theta(x)) + (1 - y) \log(1 - N_\theta(x)).$$

In the Bayesian perspective, we infer the posterior distribution over the parameters $\pi(\theta|\mathbf{x})$ which is given, considering Bayes' rule, by

$$\pi(\theta|\mathbf{x}) \propto P_{\text{prior}}(\theta) P_{\text{likelihood}}(\mathbf{x}|\theta), \tag{3.2}$$

where $P_{\text{prior}}(\theta)$ is the prior distribution on the vector of parameters and $P_{\text{likelihood}}$ the likelihood distribution of the data. The likelihood of the data can be expressed in terms of the loss function as

$$P_{\text{likelihood}}(\mathbf{x}, \mathbf{y}|\theta) = \prod_{i=1}^{N} P_{\text{elem}}(y_i, x_i|\theta) = \exp\left(- \sum_{i=1}^{N_{\text{data}}} L(x_i, y_i, \theta) \right).$$

In this case, one way to compute predictions is

$$p(y|x, \mathbf{x}) = \int_{\mathbb{R}^d} N_\theta(x) \pi(\theta|\mathbf{x}) \, d\theta \approx \frac{1}{N_{\text{iter}}} \sum_{k=1}^{N_{\text{iter}}} p(y|x, \theta_k),$$

where $(\theta_k)_{0\leqslant k\leqslant N_{\text{iter}}}$ is a Markov Chain with invariant probability measure $\pi(\cdot|\mathbf{x})$. In this case

$$y = \begin{cases} 1 & \text{if} \quad p(y|x,\mathbf{x}) \geqslant 1/2, \\ 0 & \text{if} \quad p(y|x,\mathbf{x}) < 1/2. \end{cases} \tag{3.3}$$

We focus on MCMC methods based on the discretization of SDEs to sample from π in this chapter, in particular the Adaptive Langevin dynamics (AdL) (see Section 1.2.2.4 for more details).

Remark 3.1 (Optimization vs sampling)**.** *The MCMC methods presented in Section 1.2 were initially used to sample from* $\exp(-\beta V)$ *in the context of molecular dynamics. When* β *is not equal to* 1*, setting* $\log \pi = -V$*, one is sampling from* π^β*. This shows a well known link between posterior sampling and MAP estimation [88]. When* $\beta = 1$*, we are in a Bayesian context, however if we set* $\beta \to 0$*, one can think of the MCMC method as a maximum a posteriori algorithm, since the distribution concentrates on global maxima of* π*.*

Sampling from the posterior probability measure. Various MCMC algorithms require the computation of $\nabla_\theta \log \pi(\cdot|\mathbf{x})$ given by

$$\nabla_\theta \log \pi(\theta|\mathbf{x}) = \nabla_\theta \log P_{\text{prior}}(\theta) - \sum_{i=1}^{N_{\text{data}}} \nabla_\theta L(x_i, y_i, \theta) \tag{3.4}$$

The gradient of the loss function L with respect to the neural networks' parameters can easily be computed by backpropagation, which is already implemented in many packages (including PyTorch which we use for the numerical results presented in Sections 3.3 and 3.4). To reduce the computational time, one uses minibatching to approximate the exact gradient, see the presentation in Section 1.3.1. An unbiased estimator of (3.4)(see (1.50))

$$\widehat{F}_n(\theta) = \nabla_\theta \log P_{\text{prior}}(\theta) - \frac{N_{\text{data}}}{n} \sum_{i\in I_n} \nabla_\theta L(x_i, y_i, \theta)$$

where I_n a subset of size n generated by sampling uniformly from $\{1, ..., N_{\text{data}}\}$. The error induced by minibatching on the posterior distribution for Langevin-like MCMC algorithms is at dominant order proportional to (see Chapter 2)

$$\varepsilon(n)\Delta t \min_{S\in\mathscr{S}} \|\Sigma_{\mathbf{x}} - S\|_{L^2(\pi)} = \varepsilon(n)\Delta t \|\Sigma_{\mathbf{x}} - S^*\|_{L^2(\pi)}, \tag{3.5}$$

where $\Sigma_{\mathbf{x}}(\theta)$ the empirical covariance of the gradient estimator for $n = 1$ (*i.e.* with expectations computed with respect to the random variable $\mathcal{I}$ uniformly distributed in $\{1, ..., N_{\text{data}}\}$):

$$\Sigma_{\mathbf{x}}(\theta) = \text{cov}_{\mathcal{I}} \left[\nabla_\theta (L(x_{\mathcal{I}}, y_{\mathcal{I}}, \theta))\right], \tag{3.6}$$

and the expression of $\varepsilon(n)$ depends on the type of minibatching, see Section 1.3.1 for more details (more precisely, Equations (1.55) and (1.56)). We also recall that the matrix S^* depends on the MCMC algorithm used, namely:

- $S^*_{\text{scalar}} = \frac{1}{d}\text{Tr}(\overline{\Sigma}_{\mathbf{x}})\text{I}_d$ for scalar AdL;
- S^*_{diagonal} a diagonal matrix with entries $\int_\Theta [\Sigma_{\boldsymbol{x}}(\theta)]_{i,i}\, \pi(\theta|\mathbf{x})\, d\theta$ for $1 \leqslant i \leqslant d$ for diagonal AdL;
- $S^*_{\text{matricial}} = \overline{\Sigma}_{\mathbf{x}} = \int_\Theta \Sigma_{\mathbf{x}}(\theta)\pi(\theta|\mathbf{x})\, d\theta$ for matricial AdL.

Remark 3.2. *In fact, for the sake of simplicity in numerical simulations (for which we use the PyTorch package), we use reshuffling methods to compute the estimator of the gradient. However, we expect that the error estimates given by (3.5) remains unchanged, although there is no simple expression for the noise magnitude* $\varepsilon(n)$*.*

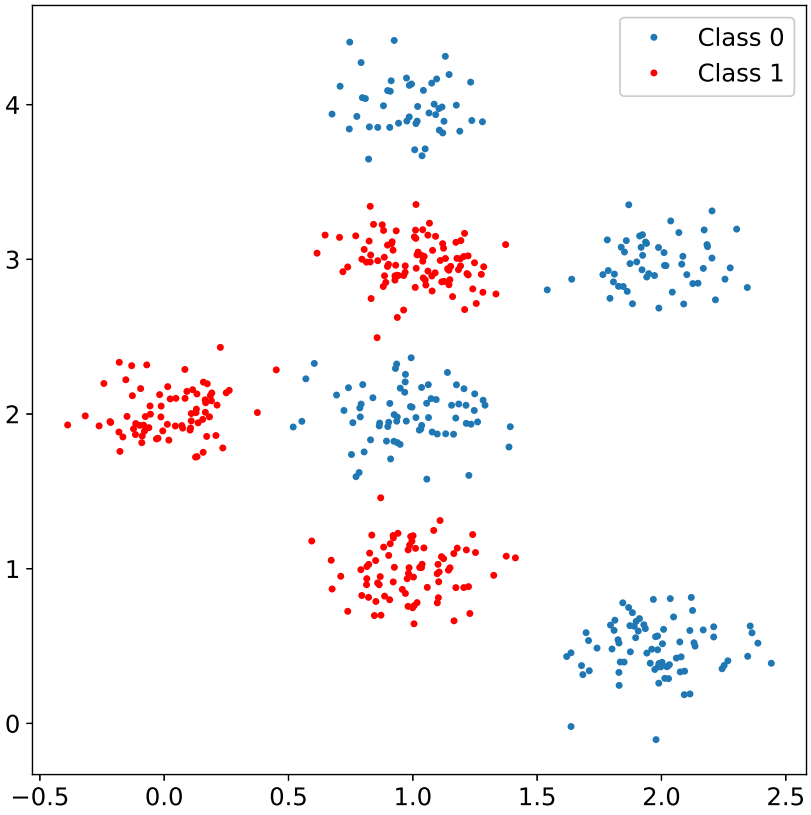


Figure 3.1 – Toy classification data.

3.2.2 Numerical toy models

We present two toy examples which will be the running numerical examples of this chapter.

Toy classification model. The first toy model we consider is the classification problem for the data presented in Figure 3.1. The data set is created by generating points using a mixture of Gaussians. The class 0 (resp. 1) is generated by considering realizations of X_0 (resp. 1) such that

$$X_0 = x_{0,\mathcal{I}} + cG,$$

$$X_1 = x_{1,\mathcal{J}} + cG,$$

where $\mathcal{I} \sim \mathcal{U}\{1,2,3\}$, $\mathcal{J} \sim \mathcal{U}\{1,2,3,4\}$, $G \sim \mathcal{N}(0,1)$ and

$$\begin{cases} x_{0,1} = (1,1), \\ x_{0,2} = (0,2), \\ x_{0,3} = (1,3). \end{cases} \qquad \begin{cases} x_{1,1} = (2,0.5), \\ x_{1,2} = (1,2), \\ x_{1,3} = (2,3), \\ x_{1,4} = (1,4). \end{cases}$$

with $c^2 = 0.03$. We use $N_{\text{data}} = 500$ to train the algorithms and generate an additional $N_{\text{test}} = 500$ for test.

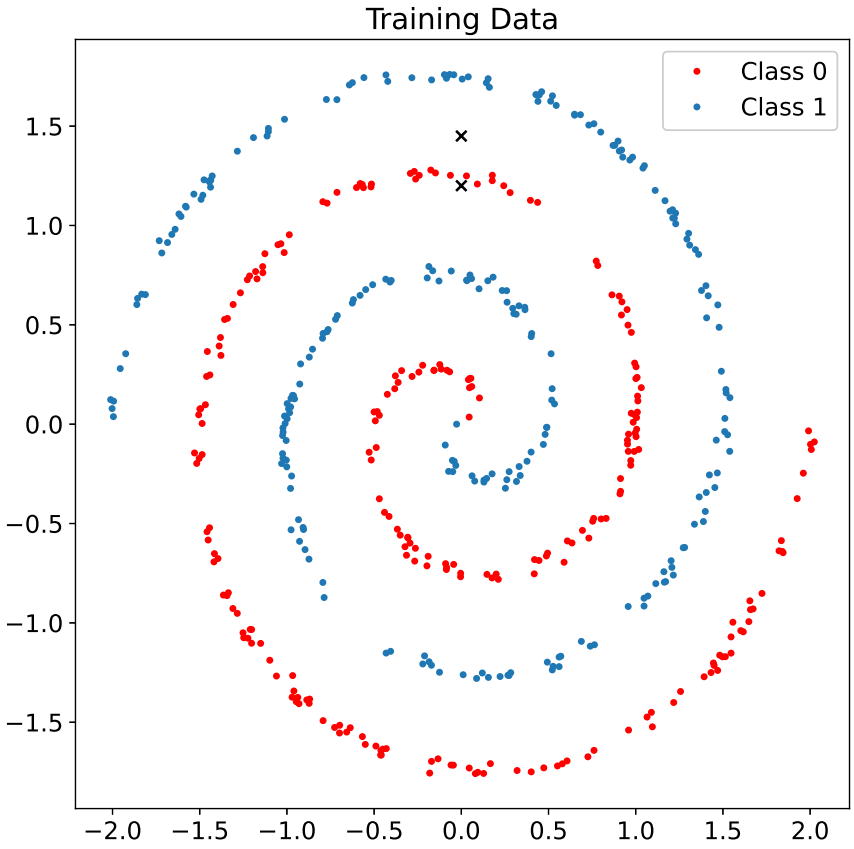


Figure 3.2 – Spiral data.

Spiral data. We next consider a slightly more complicated toy data set used in [88] and generated as follows:

$$X_1 = aU^p \cos(2bU^p\pi) + cG, \tag{3.7}$$
$$X_2 = aU^p \sin(2bU^p\pi) + cG, \tag{3.8}$$

where $G \sim \mathcal{N}(0,1)$ and $U \sim \mathcal{U}[0,1]$ where $\mathcal{U}$ stands for the uniform distribution. To create the data set, we generate realizations of (X_1, X_2) to be classified as 0 and realizations of $(-X_1, -X_2)$ to be classified as 1. We plot in Figure 3.2 the data set used with $p = 0.5$, $a = 2$, $c = 0.02$ and $b = 2$. The hyperparameter b can be seen as a way to control the complexity of the classification problem since it controls the number of turns of the data. We use $N_{\text{data}} = 500$ to train the algorithms and generate an additional $N_{\text{test}} = 500$ for test. We fix $N_{\text{data}}/2$ (respectively $N_{\text{test}}/2$) data points in classes 0 and 1.

3.2.3 Neural network architectures

For the numerical computations, we use the following architecture:

$$N_\theta(x) = \sigma(W^2 r(W^1 x + b)),$$

where $W^1 \in \mathbb{R}^{d_1 \times 2}$, $b \in \mathbb{R}^{d_1}$ and $W^2 \in \mathbb{R}^{d_1 \times 1}$. For the activation functions, r stands for the ReLU function defined in (1.4) and σ for the sigmoid function defined in (1.6) since

we are working on binary classification problems. In this case, the vector of parameters is $\theta = (W^1, b, W^2) \in \mathbb{R}^d$ with $d = 4d_1$. For the numerical simulations, we set $d_1 = 64$. For the loss function, we use the binary cross entropy loss defined in (1.3).

3.3 Analysis of the covariance matrix

In this section, we first give in Section 3.3.1 more details about how to adapt the numerical discretization of AdL to neural networks, then we provide in Section 3.3.2 numerical results on the covariance matrix of the stochastic estimator of the gradient.

3.3.1 Adaptive Langevin for neural networks

We use the numerical scheme of AdL presented in Section 2.3 (see in particular (2.52)) to sample from the posterior distribution of the neural network's parameters. In fact the neural netwrok's parameters are composed of weights and biases of different layers, namely $\theta = (W^1, b, W^2)$, where W^1 (resp. W^2) is reshaped into a vector of dimension $2d_1$ (resp. d_1). For the sake of simplicity of the numerical simulations, we separate the various "sets" of parameters (each set representing weights or biases of a particular layer). More precisely, for the scalar case, we consider the variable ξ to be of the form

$$\xi = \left(\begin{array}{c|c|c} \xi_1 \mathrm{I}_{2d_1} & 0 & 0 \\ \hline 0 & \xi_2 \mathrm{I}_{d_1} & 0 \\ \hline 0 & 0 & \xi_3 \mathrm{I}_{d_1} \end{array}\right)$$

with $\xi_1, \xi_2, \xi_3 \in \mathbb{R}$. For the diagonal case, the variable ξ is diagonal in this representation. For the matricial case, one uses a variable ξ of the form

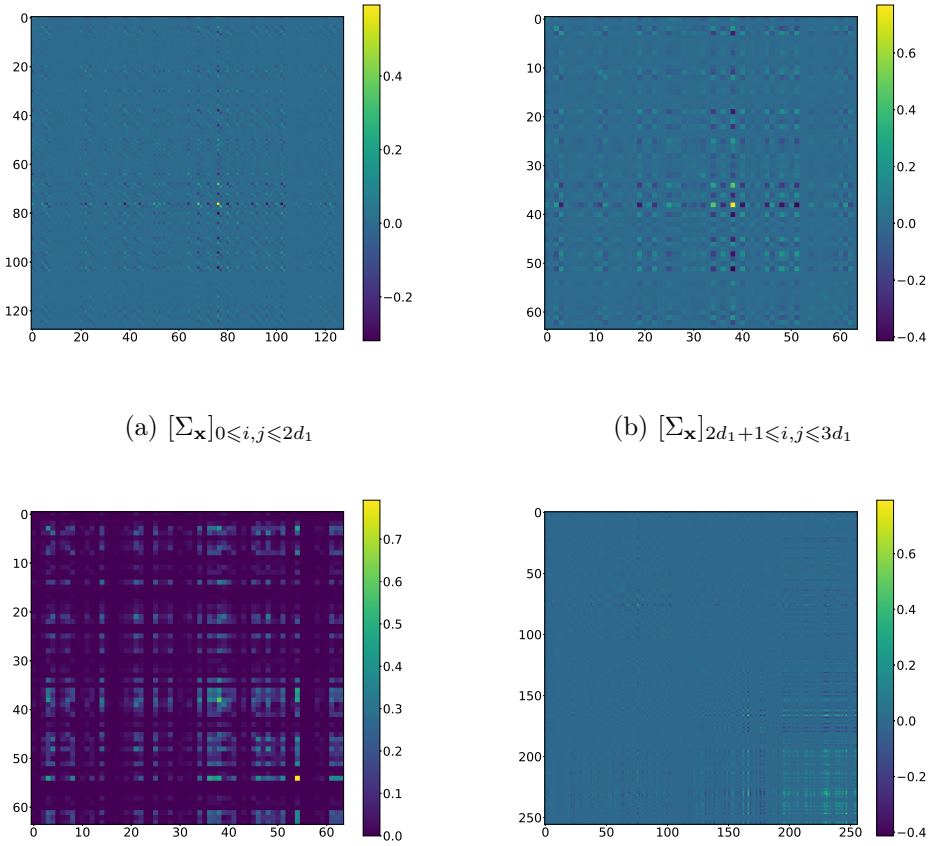
$$\xi = \left(\begin{array}{c|c|c} \xi_1 & 0 & 0 \\ \hline 0 & \xi_2 & 0 \\ \hline 0 & 0 & \xi_3 \end{array}\right)$$

with $\xi_1 \in \mathbb{R}^{2d_1 \times 2d_1}$, $\xi_2 \in \mathbb{R}^{d_1 \times d_1}$ and $\xi_3 \in \mathbb{R}^{d_1 \times d_1}$. This is in fact equivalent to using a matricial version of of AdL for each set of parameters, since the various sets are related only through the loss function. One should note however that the matricial case of AdL is computationally expensive, and cannot be considered as a practical sampling algorithm in the BNN framework. We use it only for the purpose of a better understanding of the covariance matrix of the gradient estimator.

Prior distribution. For the following numerical simulations, we set the prior to be a standard Gaussian distribution. In [64], the authors argue that BNNs are robust to the choice of prior distribution. However, they advocate to use Gaussian priors with large variances to increase the performance of the model. We set the variance to 1, which is already large for regularization terms.

Choice of hyperparameters The number of hyperparameters may appear to be a disadvantage of AdL in the context of neural networks. We try to give some insights about the choice of these hyperparameters

- set Δt small because of the coefficient N_{data} in the gradient estimator, more precisely, one should aim for $\Delta t N_{\text{data}}^2 / n = \mathcal{O}(1)$;
- choosing η such that $\eta / \Delta t$ is of order 1. This allows a better sampling of the variable ξ;
- the algorithm seems to be robust to the choice of γ.



(a) $[\Sigma_{\mathbf{x}}]_{0\leqslant i,j\leqslant 2d_1}$

(b) $[\Sigma_{\mathbf{x}}]_{2d_1+1\leqslant i,j\leqslant 3d_1}$

(c) $[\Sigma_{\mathbf{x}}]_{3d_1+1\leqslant i,j\leqslant 4d_1}$

(d) $\Sigma_{\mathbf{x}}$

Figure 3.3 – Covariance matrix of the stochastic estimator $\Sigma_{\mathbf{x}}$ at iteration 5×10^5 for the spiral data case (top left: weights of fisrt layer, top right: bias of first layer, bottom left: weights of the second layer, bottom right: full covariance matrix).

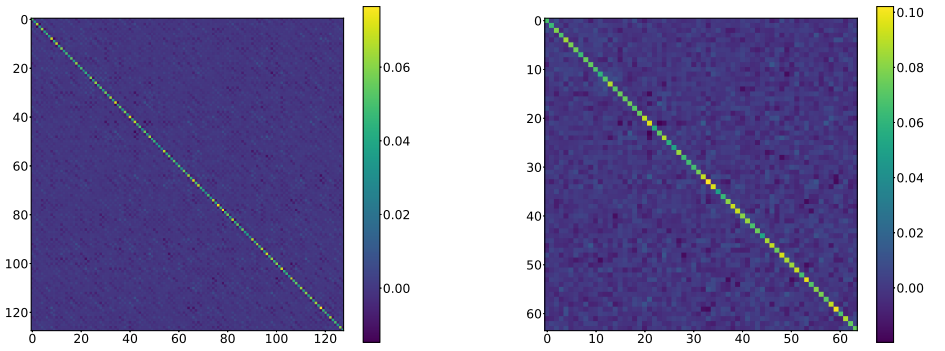
3.3.2 Numerical results

We run the numerical scheme for the scalar version of AdL for 10^7 iterations, for a minibatch size of 50 (this therefore corresponds to 10^6 epochs), $\Delta t = 5 \times 10^{-5}$, $\gamma = 1$ and $\eta = 200$. To compute the covariance matrix of the estimator of the gradient $\Sigma_{\mathbf{x}}(\theta)$, we use the following exact formula (see (2.11))

$$\Sigma_{\mathbf{x}}(\theta) = \frac{1}{N_{\text{data}}} \sum_{i=1}^{N_{\text{data}}} \nabla_\theta L(x_i, y_i, \theta) \nabla_\theta L(x_i, y_i, \theta)^T - \frac{1}{N_{\text{data}}^2} \left(\sum_{i=1}^{N_{\text{data}}} \nabla_\theta L(x_i, y_i, \theta) \right) \left(\sum_{i=1}^{N_{\text{data}}} \nabla_\theta L(x_i, y_i, \theta) \right)^T.$$

We plot in Figure 3.3 the covariance matrix during the training (at iteration 5×10^5) for the spiral case. The value of the elements of the covariance matrix of the weights of the last layer seems larger than other parts. The same remark applies to the toy classification problem. We plot in Figure 3.4 (resp Figure 3.5) the mean of the covariance matrices in the spiral case (resp. toy classification case) approximated over a trajectory of AdL as

$$\int_{\mathbb{R}^d} \Sigma_{\mathbf{x}}(\theta) \pi(\theta|\mathbf{x}) \, d\theta \approx \frac{1}{N_{\text{iter}}} \sum_{k=1}^{N_{\text{iter}}} \Sigma_{\mathbf{x}}(\theta_k). \tag{3.9}$$



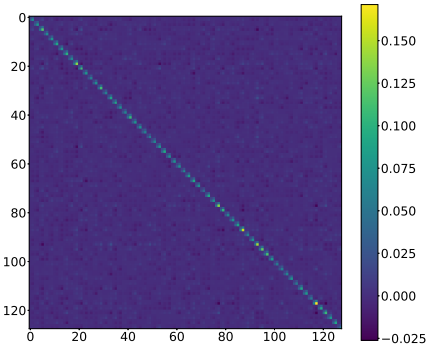
(a) $\int_{\mathbb{R}^d} [\Sigma_{\mathbf{x}}]_{0 \leqslant i,j \leqslant 2d_1}(\theta)\, d\theta$

(b) $\int_{\mathbb{R}^d} [\Sigma_{\mathbf{x}}]_{2d_1+1 \leqslant i,j \leqslant 3d_1}(\theta)\, d\theta$

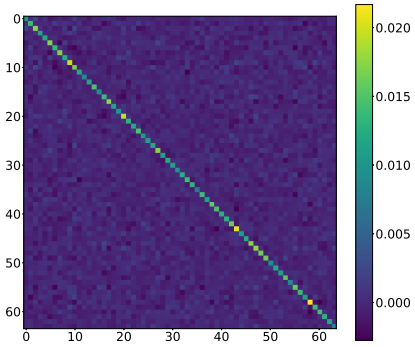
(c) $\int_{\mathbb{R}^d} [\Sigma_{\mathbf{x}}]_{3d_1+1 \leqslant i,j \leqslant 4d_1}(\theta)\, d\theta$

(d) $\int_{\mathbb{R}^d} \Sigma_{\mathbf{x}}(\theta)\, d\theta$

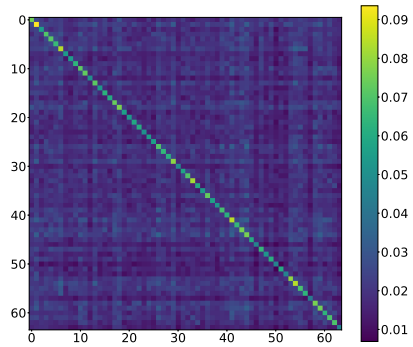
Figure 3.4 – Mean of the covariance matrix of the stochastic estimator $\Sigma_{\mathbf{x}}$ for the spiral data case (top left: weights of the first layer, top right: bias of the first layer, bottom left: weights of the last layer, bottom left: full covariance matrix).



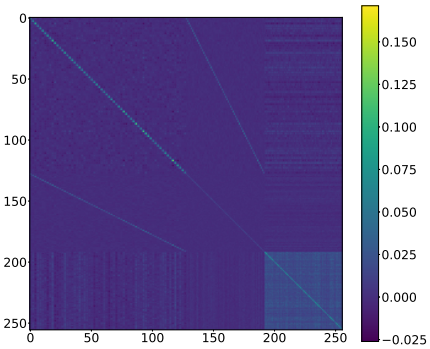
(a) $\int_{\mathbb{R}^d} [\Sigma_{\mathbf{x}}]_{0 \leqslant i,j \leqslant 2d_1}(\theta)\, d\theta$



(b) $\int_{\mathbb{R}^d} [\Sigma_{\mathbf{x}}]_{2d_1+1 \leqslant i,j \leqslant 3d_1}(\theta)\, d\theta$



(c) $\int_{\mathbb{R}^d} [\Sigma_{\mathbf{x}}]_{3d_1+1 \leqslant i,j \leqslant d_1}(\theta)\, d\theta$



(d) $\int_{\mathbb{R}^d} \Sigma_{\mathbf{x}}(\theta)\, d\theta$

Figure 3.5 – Mean of the covariance matrix of the stochastic estimator $\Sigma_{\mathbf{x}}$ for the toy classification model (top left: weights of the first layer, top right: bias of the first layer, bottom left: weights of the last layer, bottom left: full covariance matrix).

The first remark is that the mean is also dominated by the elements corresponding to the weights of the last layer. Secondly, one can say that the mean of the covariance matrix is roughly $\alpha \mathrm{I}_d$, either for each set separately or for the whole covariance. However, we should point that the covariance matrix associated with the weights of the last layer shows non zero non diagonal elements. This suggests that using the scalar, diagonal or matricial version of AdL would yield similar biases on the posterior distribution, at least for the weights and bias of the first layer. To confirm this, we plot in Figure 3.6 (resp. Figure 3.7) the Frobenius norm of the difference between the covariance matrix $\Sigma_{\mathbf{x}}$ at each iteration and its trajectory average for the spiral data case (resp. the toy classification case); more precisely we plot

$$\frac{\|\Sigma_{\mathbf{x}} - S^*_{\text{scalar}}\|}{\|S^*_{\text{scalar}}\|}, \tag{3.10}$$

$$\frac{\|S^*_{\text{matricial}} - S^*_{\text{scalar}}\|}{\|S^*_{\text{scalar}}\|}, \tag{3.11}$$

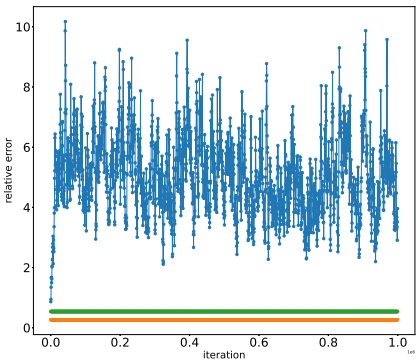
$$\frac{\|S^*_{\text{diagonal}} - S^*_{\text{scalar}}\|}{\|S^*_{\text{scalar}}\|}, \tag{3.12}$$

where S^*_{scalar}, S^*_{diagonal}, and $S^*_{\text{matricial}}$ are defined at the end of Section 3.2.1. The mean of the covariance matrix is approximated by (3.9). The plot is first produced for each set separately (for W^1 we use indices $0 \leqslant i, j, \leqslant 2d_1$, for b we use indices $2d_1 + 1 \leqslant i, j \leqslant 3d_1$ and for W^2 we use indices $3d_1 + 1 \leqslant i, j, \leqslant 4d_1$ of $\Sigma_{\mathbf{x}}$ and S^*) and then for the full covariance matrix. The covariance matrix varies a lot around its mean for all sets (W^1, b, W^2). The quantities $\|S^*_{\text{matricial}} - S^*_{\text{scalar}}\|$ and $\|S^*_{\text{diagonal}} - S^*_{\text{scalar}}\|$ are small (at least for the first layer), suggesting that the scalar version of AdL is sufficient for the weights and bias of the first layer, whereas for the last layer one needs a matricial friction to capture the covariance matrix.

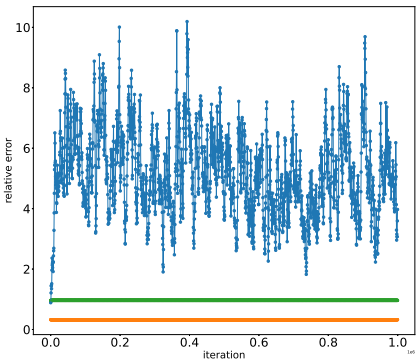
We plot in Figure 3.8 (resp. Figure 3.9) the logarithm of the eigenvalues of some covariance matrices for all sets for the spiral data (resp. for the toy classification model). It is clear from the figures that the covariance matrices are of low effective rank since there are only handful of eigenvalues above 10^{-2} and the eigenvalues decrease rapidly with respect to the index (more or less exponentially). To confirm and quantify this, we plot in Figure 3.10 (resp. Figure 3.11) the histogram of the number of eigenvalues required to reach a given value of explained variance. More precisely, denoting by $(\lambda_i)_{1 \leqslant i \leqslant d}$ the ordered eigenvalues of a covariance matrix, and by η a given fraction of explained variance, the index K_η is defined as

$$K_\eta = \min_{k \in \{1,\dots,d\}} \left\{ \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{d} \lambda_i} \geqslant \eta \right\}. \tag{3.13}$$

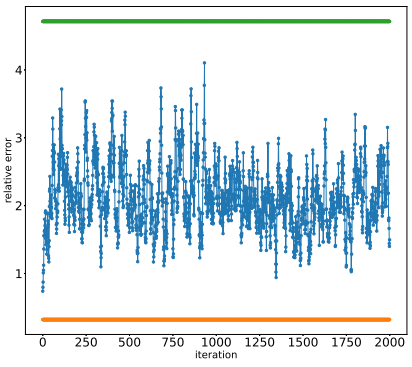
The results show that about 5% of the eigenvalues are needed to account for 0.9 of the covariance. The fact that the matrix is of low rank and that it can be explained by few eigenvalues suggests an adaptation of scalar AdL where ξ would be decomposed using the first eigenvectors only. This however requires understanding the statistics of the first eigenvectors and eigenvalues to propose relevant approximation spaces. This calls for a careful numerical investigation of the statistics of eigenvectors associated with the first eigenvalues. If the directions of these eigenvectors were fixed and did not depend on the parameters of the neural network θ (but eigenvalues could genuinely depend on θ), one could come up with a version of eADL (presented in Section 2.4) where the friction variable ξ is decomposed in a fixed basis of projection matrices.



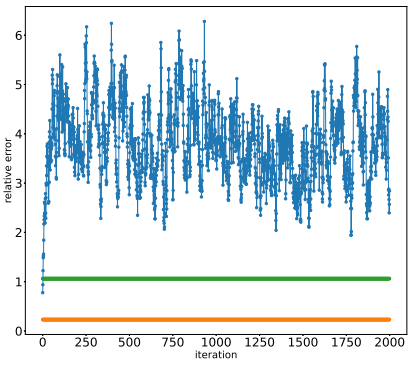
(a) weights of first layer W^1



(b) bias of first layer b

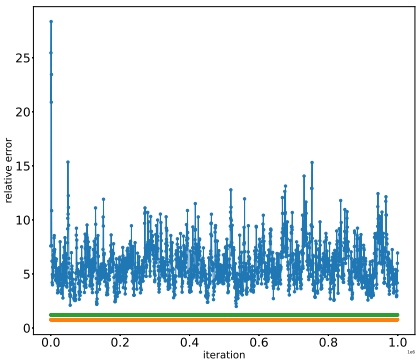


(c) weights of second layer W^2

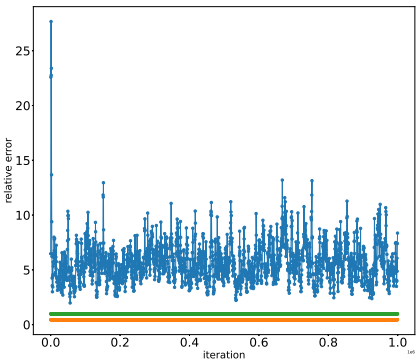


(d) $\theta = (W^1, b, W^2)$

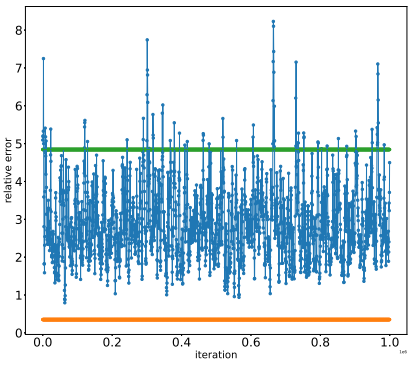
Figure 3.6 – Difference between the covariance matrix and its mean under the posterior measure $\pi(\theta|\mathbf{x})$ for the spiral data case. The blue line represents the expression (3.10), the orange line represents the expression (3.11) and the green line represents the equation (3.12).



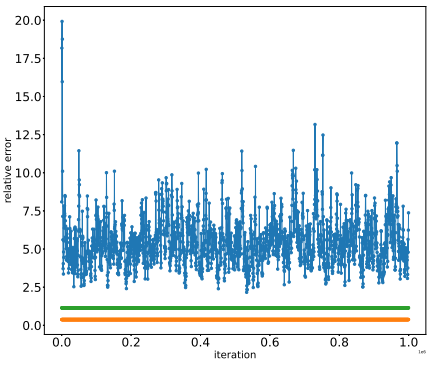
(a) weights of first layer W^1



(b) bias of first layer b

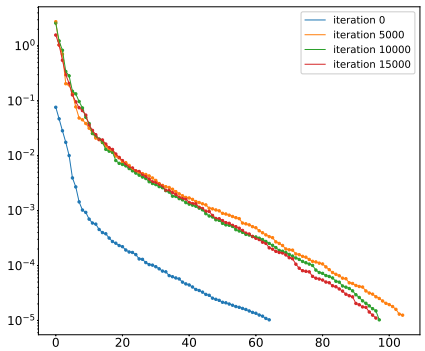


(c) weights of second layer W^2

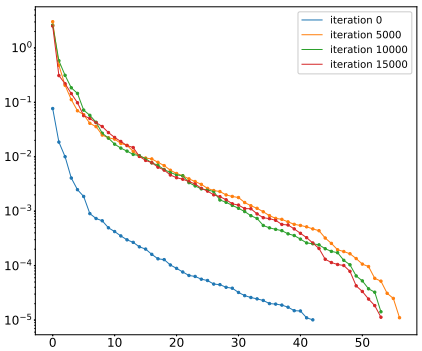


(d) $\theta = (W^1, b, W^2)$

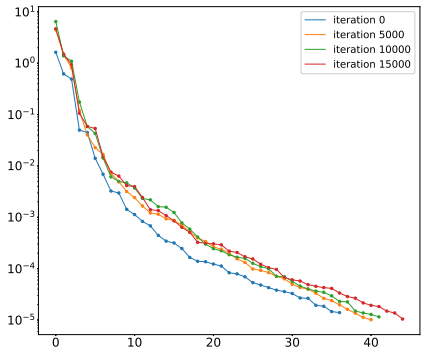
Figure 3.7 – Difference between the covariance matrix and its mean under the posterior measure $\pi(\theta|\mathbf{x})$ for the toy classification model. The blue line represents the expression (3.10), the orange line represents the expression (3.11) and the green line represents the equation (3.12).



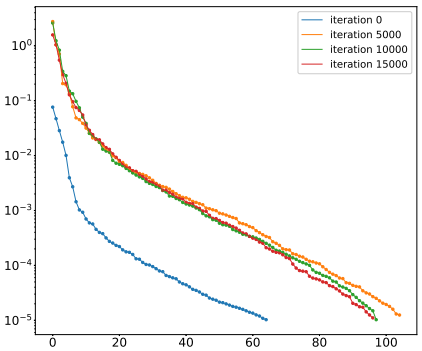
(a) weights of first layer W^1



(b) bias of first layer b

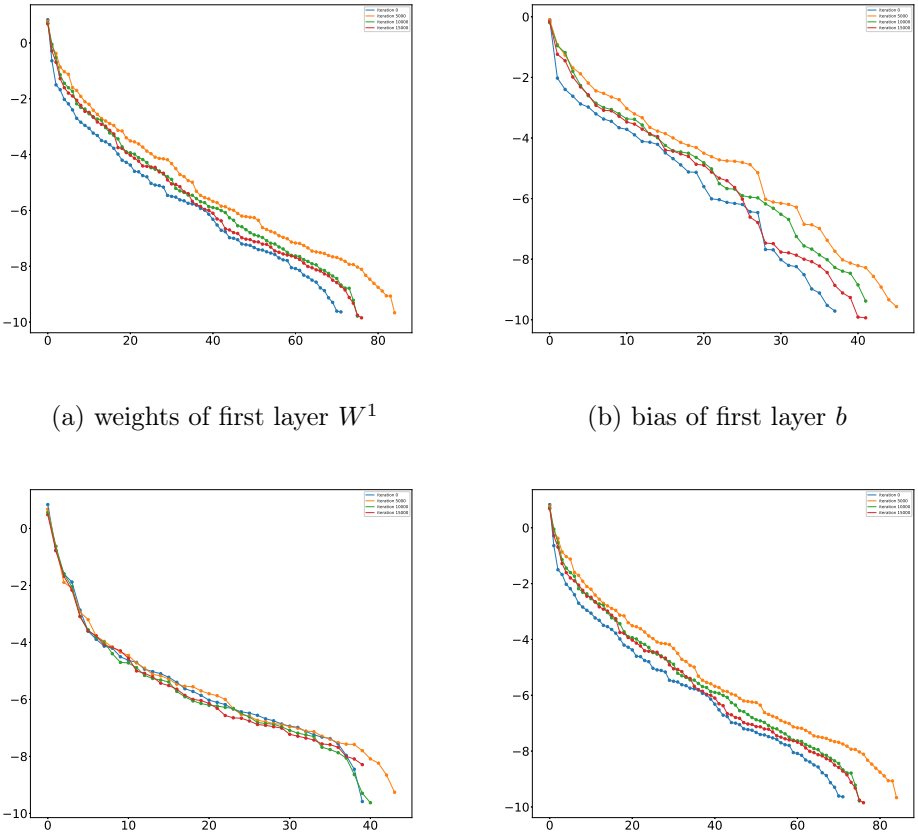


(c) weights of second layer W^2



(d) $\theta = (W^1, b, W^2)$

Figure 3.8 – Logarithm of eigenvalues of covariance matrices for the spiral data case.



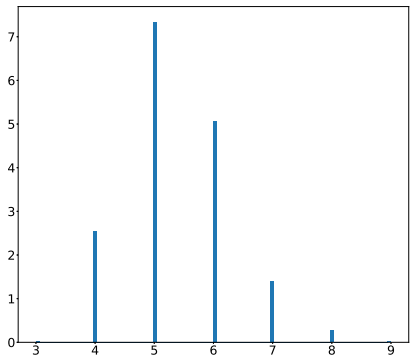
(a) weights of first layer W^1

(b) bias of first layer b

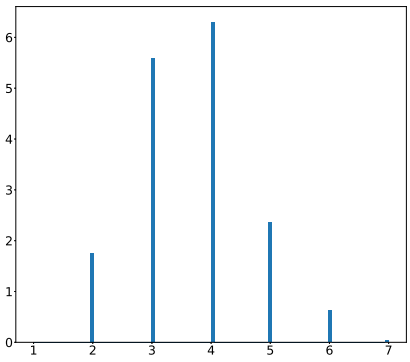
(c) weights of second layer W^2

(d) $\theta = (W^1, b, W^2)$

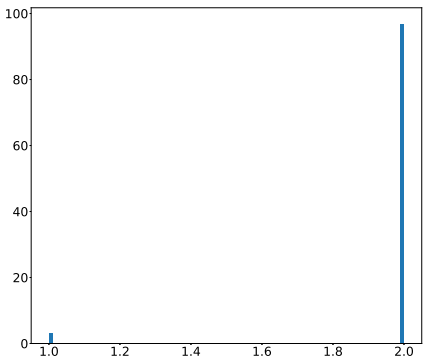
Figure 3.9 – Logarithm of eigenvalues of covariance matrices for the toy classification model.



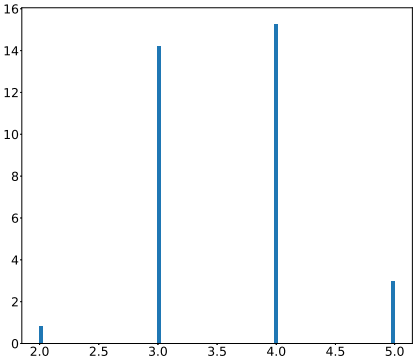
(a) weights of first layer W^1



(b) bias of first layer b

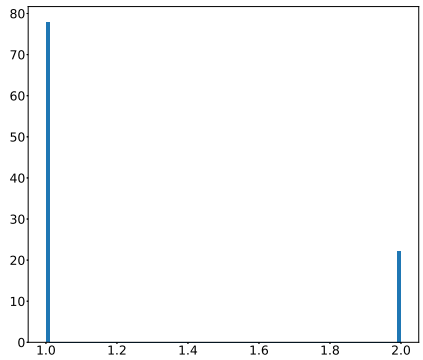


(c) weights of second layer W^2

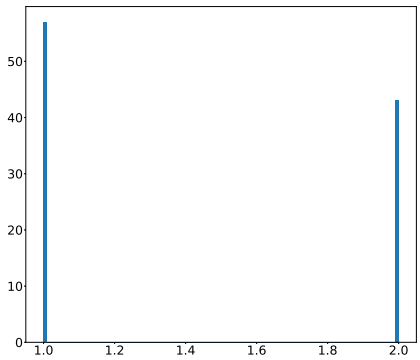


(d) $\theta = (W^1, b, W^2)$

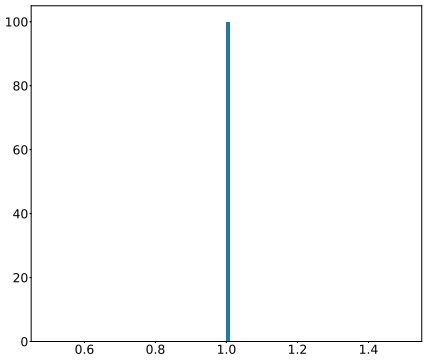
Figure 3.10 – Histogram of the number of eigenvalues given by (3.13) required to reach 0.9 of explained variance (left: weights of the first layer, middle: bias of the first layer, right: weights of the last layer) for spiral data.



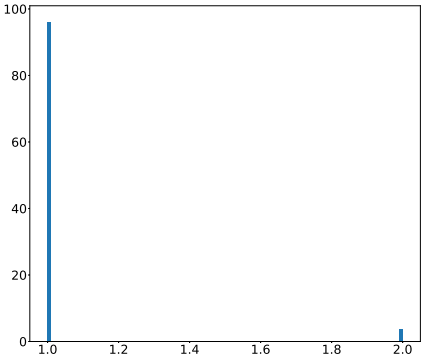
(a) weights of first layer W^1



(b) bias of first layer b

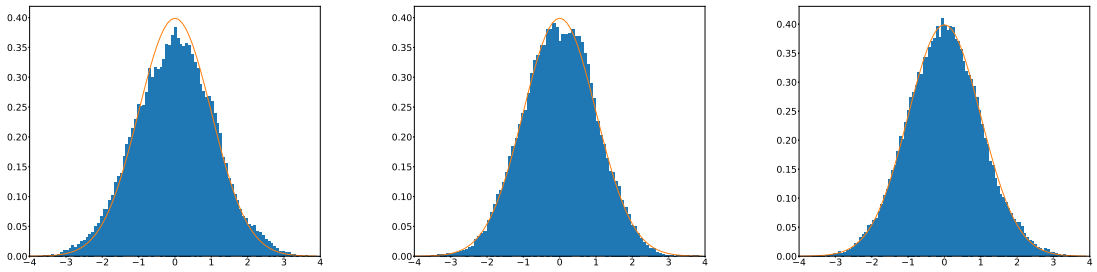


(c) weights of second layer W^2



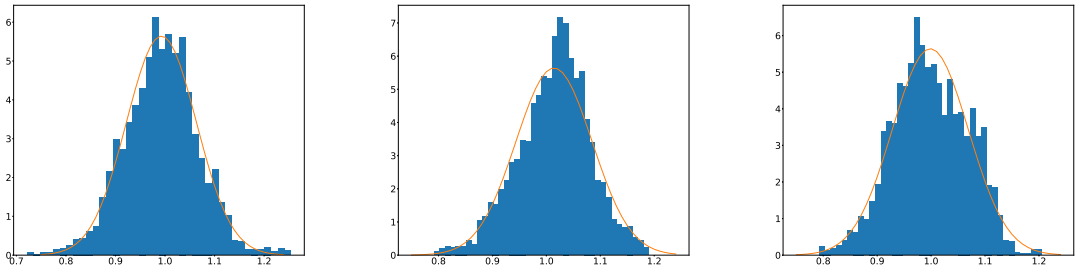
(d) $\theta = (W^1, b, W^2)$

Figure 3.11 – Histogram of the number of eigenvalues given by (3.13) required to reach 0.9 of explained variance (left: weights of the first layer, middle: bias of the first layer, right: weights of the last layer) for toy classification model.



(a) p_{11} set 1 (b) p_{19} set 2 (c) p_{13} set 3

Figure 3.12 – Histogram of randomly selected elements of p for each set for spiral data.



(a) ξ_1 (b) ξ_2 (c) ξ_3

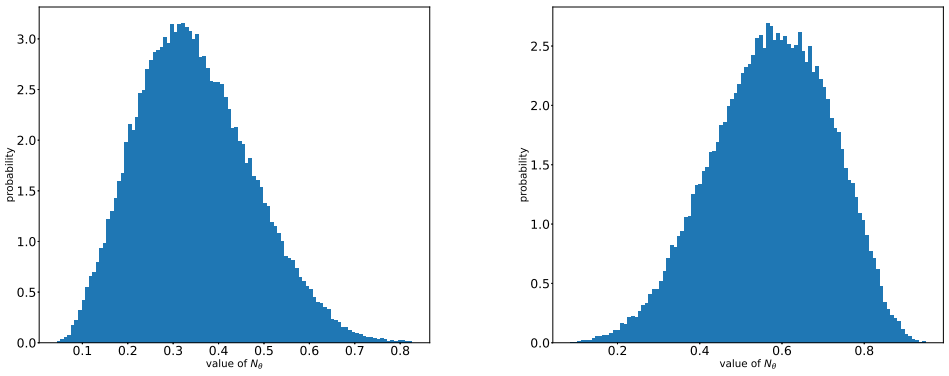
Figure 3.13 – Histogram of ξ for each set.

3.4 Sampling of the posterior distribution

Sampling from the exact posterior allows to significantly increase the performance of the neural network in addition to avoiding overfitting and overconfident networks. In this section, we look on the posterior distribution when using the scalar version of AdL. We are currently still working on quantifying the bias on this distribution. Here, we only consider the spiral data case. The hope is that results should be similar for the toy classification model but this has yet to be confirmed. We run the scalar version of AdL for $\Delta t = 5 \times 10^{-5}$, $n = 50$ for 10^6 iterations. We fix $\gamma = 1$ and $\eta = 200$. To assess that the algorithm reached the stationary distribution, we check the marginal distribution of ξ and some components of p see Figures 3.12 and 3.13. The variables ξ and p are indeed normally distributed, as suggested by the invariant probability measure of AdL, see Section 2.3. We consider the distribution of N_θ instead of the the distribution of θ to avoid taking into account the various symmetries of the network related to exchanges of parameters and also because this is the quantity of interest. It is also suggested in [64] that the mixing of N_θ is better compared to the mixing in the parameters space. We plot in Figure 3.14 the resulting histogram of $N_\theta(x)$ for $x = (0, 1.08)$ and $x = (0, 1.35)$ (the points are represented in Figure 3.2). The second point is relatively out of the distribution. The posterior distribution corresponding to this point shows a higher variance.

3.5 Perspectives

The analysis of the covariance matrix, together with the results of Chapter 2 suggests that numerical methods can be developed to reduce the minibatching error on the posterior distribution of BNNs without drastically increasing the computational time. Here are some



(a) $x = (0, 1.2)$ (b) $x = (0, 1.45)$

Figure 3.14 – Histogram of $N_\theta(x)$ for various values of x.

suggestions to explore to this end:

- A scalar version of the extended version of AdL, where the first eigenvalue of the stochastic estimator of the gradient is decomposed in a finite basis of functions. More precisely, the idea would be to approximate the covariance matrix as $\Sigma_{\mathbf{x}} = \sum_{i=1}^{I} \lambda_i(\theta) e_i(\theta) e_i^T(\theta)$, for I small, where $e_i(\theta)$ represents the eigenvectors associated with the eigenvalue $\lambda_i(\theta)$. One can use for example a separated neural network to this end;
- Use extrapolation with two different sizes of minibatches in conjunction with coupling to reduce the minibatching error and reduce the variance. Note that extrapolation using Δt for Langevin dynamics and SGD has already been studied in [153].

Moreover, work is in progress to confirm our numerical observations on more realistic data sets such as MNIST.

CHAPTER 4

GENERATIVE METHODS FOR TRANSITION PATHS IN MOLECULAR DYNAMICS

Contents

The material for this chapter has been preprinted in [92] (submitted to ESAIM Proceedings *in the special issue gathering contributions from the research projects initiated during the 6 week long summer research school CEMRACS 2021)..*

Abstract.

Molecular systems often remain trapped for long times around some local minimum of the potential energy function, before switching to another one – a behavior known as metastability. Simulating transition paths linking one metastable state to another one is difficult by direct numerical methods. In view of the promises of machine learning techniques, we explore in this work two approaches to more efficiently generate transition paths: sampling methods based on generative models such as variational autoencoders, and importance sampling methods based on reinforcement learning.

4.1 Introduction

Molecular dynamics aims at simulating the physical movement of atoms in order to sample the Boltzmann–Gibbs probability measure and the associated trajectories, and to compute macroscopic properties using Monte Carlo estimates [48, 5]. One of the main difficulties when performing these numerical simulations is metastability: the system tends to stay trapped in some regions of the phase space, typically in the vicinity of local maxima of the target probability measure. In this context, transitions from one metastable state to another one are of particular interest in complex systems, as they characterize for example crystallisation or enzymatic reactions. These reactions happen on a long time scale compared to the molecular timescale, so that the simulation of realistic rare events is computationally difficult.

On the one hand, many efforts have been devoted to the development of rare events sampling methods in molecular dynamics. The goal of these methods is to characterize transition paths and to compute associated transition rates and mean transition times; see for instance [56] for a review of rare event sampling methods in molecular dynamics. The most notable methods can be classified in two groups:

(i) importance sampling techniques, where the dynamics is biased (by modifying the potential for instance) to reduce the variance of Monte Carlo estimators when computing expectations, see for instance [47, 25] for more details, and also [94, Section 6.2]. It is possible to use adaptive importance sampling strategies to choose the importance function, see [93, Chapter 5]. Another viewpoint is offered by the framework of stochastic control, as in [56] where the modification in the drift of the dynamics is determined by the solution of an optimal control problem.

(ii) splitting methods, where the idea is to decompose the rare event to sample as a succession of moderately rare events. In the context of trajectories relating two local maxima of the target probability measure, this can be done in an adaptive manner using the so-called Adaptive Multilevel Splitting algorithm, where an ensemble of trajectories are concurrently evolved, removing the ones that lag behind in terms of progress towards the target state, and replicating the ones exploring more successfully the path towards the target state; see [34, 8, 28, 29].

On the other hand, generative models aim at generating samples whose distribution approximates some unknown target distribution. They have attracted a lot of attention lately due to their wide range of applications, such as text translation, out-of-distribution detection, generation of new human poses, etc. The currently most popular generative models are Generative Adversarial Networks (GANs) [53], Variational AutoEncoders (VAEs) [71, 124], as well as Energy-Based Models and their extensions; see [20] for a review of the most important models. Generative models have also been used in the context of rare event sampling. For instance, GANs can be used to generate data from extreme tails of (heavy tailed) distributions, as discussed in [6] and references therein. Generative models also offer the perspective to detect anomalies which can be considered as rare events [7, 40], or maybe even generate anomalous states by sampling from outlier regions in the embedding space [80]. In the context of molecular dynamics, machine learning techniques have been used to study transition pathways [151, 133, 161]. In [133], the authors suggest to use a neural network to approximate the committor function giving the probability of reaching a metastable state before another one, importance sampling techniques being used to reduce the statistical error in these computations. VAEs have been used in [151] to find collective variables by using mixtures of Gaussian priors in the latent space to encode the trajectories.

The goal of this work is to explore some machine learning techniques to efficiently generate transition paths in molecular dynamics. We first tried a data-driven generative method: from a given data set of transition paths, we learn to generate new ones using variational

autoencoders. Using VAEs naively, the temporal aspect of the trajectories is not encoded in the latent variables, which produces unconvincing results when generating new trajectories. We tested two techniques to learn the temporal aspect on the latent space, namely vector quantized variational autoencoders [149] and variational recurrent neural network [33] but these approaches were not successful. We therefore turned to a data free approach, relying on reinforcement learning algorithms to construct trajectories following the dynamics introduced in (4.2), while guiding it to transition from one well to another one. Reinforcement learning is more convenient than generative approaches learning from a dataset when the construction of the data set is computationally challenging.

This work is organized as follows. We introduce the main settings of the molecular dynamics problem we tackle in Section 4.2. In Section 4.3, we briefly present variational autoencoders, and the methods used to learn the temporal aspect on the latent space. Section 4.4 is dedicated to results obtained with reinforcement learning.

4.2 Sampling transition paths of metastable processes

We present in this section the main settings of the problem we tackle.

Sampling from the Boltzmann–Gibbs distribution. Let us consider a diffusion process $(q_t)_{t\geqslant 0}$ with values in $\mathcal{D} = \mathbb{R}^d$, whose drift derives from a potential $V : \mathcal{D} \to \mathbb{R}$. We typically consider the case when the potential V has many local minima. We want to sample from the Boltzmann-Gibbs distribution given by $\mu(dq) = Z^{-1}\mathrm{e}^{-\beta V(q)}\,dq$. In this case, one of the main issues when sampling trajectories is metastability: the system remains trapped for a long time around some local minimum of V before jumping to another local minimum. Our goal is to simulate *transition paths*, that we define in this work as trajectories which, from a fixed initial condition q_0 located in an initial potential well A, reach a pre-specified set $B \subset \mathbb{R}^d$ before time $T \geqslant 0$. Typically, B corresponds to another well in the energy landscape.

Overdamped Langevin dynamics. The evolution of molecular systems can be modelled by Langevin dynamics, which are stochastic perturbations of the Hamiltonian dynamics. For simplicity in this work, we consider that the system evolves according to the overdamped Langevin diffusion

$$dQ_t = -\nabla V(Q_t)\,dt + \sqrt{\frac{2}{\beta}}\,dW_t, \tag{4.1}$$

where $(W_t)_{t\geqslant 0}$ is a standard d-dimensional Wiener process. The dynamics (4.1) admits the Boltzmann–Gibbs distribution as a unique invariant probability measure (see for instance [73]). In practice, we use a Euler–Maruyama discretization with a time step $\Delta t > 0$ to approximate the exact solution of the stochastic differential equation (4.1). We obtain the following discrete-time process:

$$q_{k+1} = q_k - \nabla V(q_k)\Delta t + \sqrt{\frac{2\Delta t}{\beta}}G_k, \tag{4.2}$$

where $G_k \sim \mathcal{N}(0, \mathrm{I}_d)$ for all $k \geqslant 0$ are independent Gaussian random variables. We assume that the drift of the dynamics is globally Lipschitz or that Lyapunov conditions are satisfied, so that the Markov chain corresponding to the time discretization (4.2) admits a unique invariant probability measure, denoted by $\mu_{\Delta t}$; see [101]. It is well known that the Euler–Maruyama discretization (4.2) is consistent (weakly and strongly) of order 1, and that $\mu_{\Delta t}$ agrees with μ up to errors of order Δt (see for instance [144] and [101, Theorem 7.3] for the latter point).

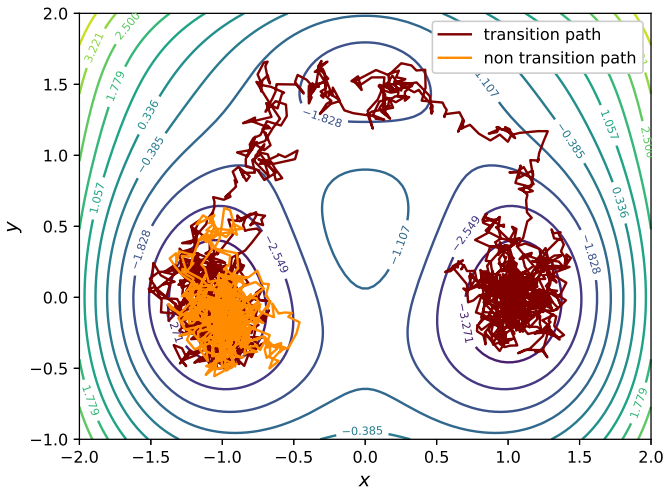


Figure 4.1 – Transition path and non transition path in the 2-dimensional potential given by (4.3).

Two–dimensional numerical example. To illustrate the metastability issue, we present a simple two dimensional example, which will be the running numerical example of this paper. We assume that $\mathcal{D} = \mathbb{R}^2$, and consider the following potential for $q = (x, y)$ (already used in [117, 105]):

$$V(q) = 3\exp\left(-x^2 - \left(y - \frac{1}{3}\right)^2\right) - 3\exp\left(-x^2 - \left(y - \frac{5}{3}\right)^2\right) - 5\exp\left(-(x-1)^2 - y^2\right) - 5\exp\left(-(x+1)^2 - y^2\right) + 0.2x^4 + 0.2\left(y - \frac{1}{3}\right)^4. \tag{4.3}$$

We plot in Figure 4.1 two trajectories generated using the discretization (4.2), with $\Delta t = 5 \times 10^{-3}$, $\beta = 3.5$ and a final time $T = 10$. The first trajectory displayed in orange remains trapped in the first well A (located around $(-1, 0)$), whereas the second trajectory displayed in red jumps to the second well B by going through the local minimum of V on the top. An alternative path for the particle to go from the first to the second well passes through the bottom. We performed direct numerical simulations of trajectories initialized from $(-1.05, -0.04)$ to confirm that sampling paths transitioning from A to B is rare, with a probability of the order of 0.01. Directly integrating (4.2) to explore the configurational space is therefore inefficient in this case. The issue becomes even more acute in higher dimensions, especially when the potential V has many local minima.

4.3 Generating transition paths with Variational AutoEncoders

The purpose of this section is to use a data set of transition paths to generate new ones using variational autoencoders (VAEs). We first present VAEs in Section 4.3.1. We briefly recall in Section 4.3.2 the convolutional layers which are the building blocks of the various architectures used in this section. The construction of the data set used to train the various models is discussed in Section 4.3.3. We then present in Section 4.3.4 the 2-dimensional VAE we consider, alongside with the numerical results for this model. Finally, in Section 4.3.5 we describe some methods to incorporate the temporal aspect of the data in the latent space, first with a “naive" VAE, and then using vector quantized VAEs (VQ-VAEs) and variational

recurrent neural networks.

4.3.1 Presentation of Variational AutoEncoders

Variational AutoEncoders, as introduced by [71, 124], are a class of generative models based on latent variables. Assume that the data consists of n observed variables, which we denote by $\mathbf{q} = (q^1, \dots, q^n) \in \mathcal{D}^n$, distributed according to some probability measure $p(\cdot)$. In our context, each element q^i is a trajectory, *i.e.* a time ordered sequence of configurations of the system (see Section 4.3.3 for a more precise description).

The aim is to approximate the distribution of the data by a parametric distribution $p_\theta \approx p$. Instead of considering simple but limited parametric distributions, generative models assume that there exist latent, unobserved variables, which we denote by $\mathbf{z} = (z^1, \dots, z^n) \in (\mathbb{R}^\ell)^n$, where ℓ, the dimension of the latent variables (called the intrinsic dimension), is generally smaller than the dimension of the data. In this context, the likelihood is given by

$$p_\theta(q) = \int_{\mathbb{R}^\ell} p_\theta(q, z)\, dz = \int_{\mathbb{R}^\ell} p_\theta(q|z) p_\theta(z)\, dz. \tag{4.4}$$

The aim is to maximize the likelihood of $(q^1, q^2, ..., q^n)$ with respect to θ. Note that, with some abuse of notation, the joint distributions of z and q, the marginal distributions in q and z and the prior distribution on z are all denoted by p_θ. The joint distribution of $(\mathbf{q}, \mathbf{z})$ is defined through a parametric model with unknown parameter θ as

$$\begin{aligned} z^i &\sim p_\theta(z), \\ q^i | z^i &\sim p_\theta(q|z^i). \end{aligned} \tag{4.5}$$

Variational inference. In the setting considered here, computing the likelihood (4.4), as well as the conditional probability $p_\theta(z|q)$, is intractable. In view of Bayes' relation, the likelihood $p_\theta(q)$ and the conditional likelihood $p_\theta(z|q)$ are related as

$$p_\theta(q) = \frac{p_\theta(q, z)}{p_\theta(z|q)}.$$

From a Bayesian perspective, one of the aims of VAEs is the inference of the posterior distribution of the latent variables, $p_\theta(z|q)$. To do so, VAEs rely on *variational inference* (see, e.g., [17]), which can be seen as an alternative to Markov Chain Monte Carlo sampling in complex Bayesian models where $p_\theta(z|q)$ is intractable. The idea of variational inference is to posit a family of probability distributions Π, and to approximate the posterior $p_\theta(z|q)$ by a distribution in the family Π of distributions in z indexed by q which minimizes the Kullback–Leibler divergence

$$\pi^\star = \operatorname*{argmin}_{\pi \in \Pi} \mathsf{KL}\left(\pi(z|q) || p_\theta(z|q)\right), \qquad \mathsf{KL}(\pi(z|q)||p_\theta(z|q)) = \int_{\mathbb{R}^\ell} \pi(z|q) \log\left(\frac{\pi(z|q)}{p_\theta(z|q)}\right) dz. \tag{4.6}$$

However, the minimization problem (4.6) cannot be solved directly, as it involves the computation of the intractable marginal log-likelihood $\log p_\theta(q)$:

$$\begin{aligned} \mathsf{KL}(\pi(z|q)||p_\theta(z|q)) &= \mathbb{E}_{\pi(\cdot|q)}[\log \pi(z|q)] - \mathbb{E}_{\pi(\cdot|q)}[\log p_\theta(z|q)] \\ &= \mathbb{E}_{\pi(\cdot|q)}[\log \pi(z|q)] - \mathbb{E}_{\pi(\cdot|q)}[\log p_\theta(z, q)] + \log p_\theta(q). \end{aligned} \tag{4.7}$$

Reordering the terms in (4.7), we obtain:

$$\begin{aligned} \log p_\theta(q) - \mathsf{KL}(\pi(z|q)||p_\theta(z|q)) &= -\mathbb{E}_{\pi(\cdot|q)}[\log \pi(z|q)] + \mathbb{E}_{\pi(\cdot|q)}[\log p_\theta(z, q)] \\ &= \underbrace{\mathbb{E}_{\pi(\cdot|q)}[\log p_\theta(q|z)] - \mathsf{KL}(\pi(z|q)||p_\theta(z))}_{\mathsf{ELBO}}. \end{aligned} \tag{4.8}$$

The term on the right hand side of the previous equality is called the *Evidence Lower Bound* (ELBO). On the one hand, for $p_\theta(q)$ fixed, solving the minimization problem (4.6) is equivalent to maximizing the ELBO with respect to the variational distribution $\pi(z|q)$. On the other hand, since $\mathsf{KL}(\pi(z|q)||p_\theta(z|q)) \geqslant 0$, the ELBO is a lower bound on the marginal log-likelihood $\log p_\theta(q)$. Thus, maximizing the ELBO with respect to θ provides a proxy for the maximum likelihood estimate of parameter θ. Note that this is an important goal, since approximating θ yields an approximation to the distribution of the observed data $\mathbf{q}$, and thus allows to generate new samples.

Variational autoencoders. The idea of variational inference is to choose a class of distributions Π which is consistent with our intuition of the problem, and yields a tractable optimization problem. Variational Autoencoders rely on a different class of distributions Π, defined as follows:

$$\pi(z|q) = \Psi(z|g_\phi(q)). \tag{4.9}$$

In (4.9), $\{\Psi(.|\gamma), \gamma \in \mathscr{G}\}$ is a parametric family of densities, and $g_\phi : \mathcal{D} \to \mathscr{G}$ is a differentiable function. For instance, $\Psi(z|\mu, \Sigma)$ can be chosen as the multivariate Gaussian distribution with mean μ and covariance matrix Σ. In this case, $g_\phi(q) = (\mu_\phi(q), \Sigma_\phi(q))$, where μ_ϕ and Σ_ϕ are parametrized by a neural network with weights ϕ. The function g_ϕ is referred to as the *encoder*: it allows to construct the distribution of the latent variables, given the observations.

Autoencoding variational bound algorithm. Assembling the concepts of the previous paragraphs, VAEs aim at solving the following optimization problem:

$$\underset{\theta,\phi}{\operatorname{argmin}} \sum_{i=1}^{n} \mathcal{L}(\theta, \phi; q^i), \tag{4.10}$$

where

$$\mathcal{L}(\theta, \phi; q^i) := \mathbb{E}_{z\sim\Psi(.|g_\phi(q^i))}\left[\log p_\theta\left(q^i|z\right)\right] - \mathbb{E}_{z\sim\Psi(.|g_\phi(q^i))}\left[\log \Psi\left(z|g_\phi(q^i)\right) - \log(p_\theta(z))\right],$$

where $(q^1, ..., q^n)$ is the given data set of observed quantities. Problem (4.10) is solved using a stochastic gradient descent (SGD) algorithm. While computing the gradient with respect to θ is straightforward, using a Monte Carlo estimator to compute the gradient of $\mathcal{L}(\theta, \phi; q^i)$ with respect to ϕ however leads to a large variance [115, 71]. It is suggested in [71] to use the so-called reparametrization trick to obtain expressions of gradients both with respect to ϕ and θ. The method goes as follows. Considering a diagonal covariance matrix $\Sigma_\phi = \mathrm{diag}(\sigma_\phi^2)$ with $\sigma_\phi \in \mathbb{R}^\ell$, and since $\{\Psi(.|\gamma), \gamma \in \mathscr{G}\}$ corresponds to a family of multivariate Gaussian distributions $\Psi(q|\mu_\phi, \Sigma_\phi)$ with mean μ_ϕ, the random variable $z \sim \Psi(z|g_\phi(q^i))$ can be reparametrized as

$$z = \mu_\phi(q^i) + \mathrm{diag}(\sigma_\phi(q^i))\varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, \mathrm{I}_\ell). \tag{4.11}$$

Note that similar reparametrizations can be considered whenever Ψ is a "location-scale" family of distribution (Laplace, Student, etc.). Alternatively, if Ψ has a tractable CDF, one can reparametrize z with a uniform random variable $\varepsilon \sim \mathcal{U}([0, 1])$. With the choice (4.11),

$$\begin{aligned}\mathcal{L}(\theta, \phi; q^i) = {} & \mathbb{E}_{\varepsilon\sim\mathcal{N}(0,1)}[\log p_\theta(q|\mu_\phi(q^i) + \sigma_\phi(q^i)\varepsilon)] \\ & - \mathbb{E}_{\varepsilon\sim\mathcal{N}(0,1)}[\log \Psi(\mu_\phi(q^i) + \sigma_\phi(q^i)\varepsilon|g_\phi(q^i)) - \log p_\theta(\mu_\phi(q^i) + \sigma_\phi(q^i)\varepsilon)].\end{aligned} \tag{4.12}$$

To make the loss function (4.12) even more explicit, we consider that the prior $p_\theta(z)$ is a centered reduced Gaussian distribution. In this case, the term on the second line, which corresponds to the Kullback–Leibler divergence of two Gaussian distributions, can be analytically

computed. The term on the first line can be approximated by a Monte Carlo discretization using $(\varepsilon^1, \dots, \varepsilon^L)$ independent ℓ-dimensional standard normal Gaussian vectors. This amounts to considering the following estimator $\widehat{\mathcal{L}}(\theta, \phi; q^i)$ of $\mathcal{L}(\theta, \phi; q^i)$:

$$\widehat{\mathcal{L}}(\theta, \phi; q^i) = \frac{1}{L} \sum_{j=1}^{L} \log p_\theta(q^i | \mu_\phi(q^i) + \sigma_\phi(q^i)\varepsilon^j) + \sum_{k=1}^{\ell} \left[1 + \log(\sigma_\phi^2(q^i)_k) - \mu_\phi^2(q^i)_k - \sigma_\phi^2(q^i)_k\right], \tag{4.13}$$

where we denote by $\mu_\phi(q)_k$ and $\sigma_\phi(q)_k$ the components of the vectors $\mu_\phi(q)$ and $\sigma_\phi(q)$. In practice, we consider $L = 1$ as suggested in [71]. We also assume that $p_\theta(q|z) = \Phi(q|f_\theta(z))$, where $\{\Phi(\cdot|\eta), \eta \in \mathscr{H}\}$ is a family of densities parametrized by $\eta \in \mathscr{H}$. For instance, one can consider for $\Phi(q|\mu, \Sigma)$ the multivariate Gaussian distribution with mean μ and covariance matrix Σ, and $f_\theta(z) = (\mu_\theta(z), \Sigma_\theta(z))$ where $\Sigma_\theta(z) = \sigma_\theta^2(z)\mathrm{I}_d$. This particular case amounts to considering p_θ as an infinite mixture of Gaussians. The function $f_\theta : \mathbb{R}^\ell \to H$ is called the *decoder*: it allows to reconstruct the distribution of the data given the latent variables. In general, one assumes f_θ to be parametrized by a neural network with weights θ. Note that in this specific Gaussian setting,

$$\log p_\theta(q|z) = -\frac{d}{2}\log(2\pi) - d\log(\sigma_\theta(z)) - \frac{\|q - \mu_\theta(z)\|^2}{2\sigma_\theta^2(z)}, \tag{4.14}$$

where z is given by (4.11). In fact, for the VAEs we used, we considered σ_θ as a constant and fixed it to $\sigma_\theta = 1.2 \times 10^{-2}$. As a result, the terms in (4.13) and (4.14) are differentiable with respect to ϕ and θ, and the gradients can be computed using backpropagation.

To summarize, we use a neural network as the encoder, which takes as input a trajectory q^i, and gives as output $\mu_\phi(q^i), \sigma_\phi(q^i)$. We use the reparametrization trick to compute the latent variable z^i, which is given as input to the decoder (which is itself a neural network). The output of the decoder is the trajectory $\mu_\theta(z^i)$. The loss function in (4.10) (using in particular (4.14)) can then be minimized with respect to θ, ϕ with gradients computed using backpropagation. It is approximated in practice by minibatching. The building block of the encoder and decoder we use are convolutional neural networks presented in the next section.

4.3.2 Convolutional neural networks

Convolutional neural networks [81, 82] are generally used for data invariant under translation and scaling. They are composed of hidden stacked layers: convolutional layers, pooling layers, normalization layers, to which one generally adds a fully connected layer at the end. We focus on convolutional layers, since they are the only ones used in the various architectures we consider alongside the fully connected layers and batch normalization layers. The input, which corresponds to the first layer, is a trajectory q.

A convolutional layer transforms an input time series $(x_t)_{1 \leqslant t \leqslant T_{\text{in}}} \in \mathbb{R}^{T_{\text{in}} \times m_{\text{in}}}$ into a time series $y \in \mathbb{R}^{T_{\text{out}} \times m_{\text{out}}}$. The parameters of a convolutional layer are the elements of the m_{out} convolution kernels of size $k \times m_{\text{in}}$, denoted by $W \in \mathbb{R}^{k \times m_{\text{in}} \times m_{\text{out}}}$, with $k < T_{\text{in}}$. More precisely, each convolution kernel $W^j \in \mathbb{R}^{k \times m_{\text{in}}}$ for $j \in \{1, ..., m_{\text{out}}\}$ performs Frobenius products (denoted by :) with parts of the input time series. The kernel W^j convolves through the input to form a new time series, which is then passed to an activation function. The stride s controls the shift of the kernel through the input time series. More precisely, y_t^j is defined for $1 \leqslant t \leqslant T_{\text{out}}$ and $1 \leqslant j \leqslant m_{\text{out}}$ as

$$y_t^j = f\left(W^j : x_{[(t-1)s+1;(t-1)s+k]}\right).$$

There are several important things to notice. The first one is that the output time series $(y_t)_{1 \leqslant t \leqslant T_{\text{out}}}$ does not have the same dimension as the input time series. Its length T_{out} is

defined by the following formula:

$$T_{\text{out}} = \frac{T_{\text{in}} - k}{s} + 1. \tag{4.15}$$

If we want it to have the same size as the input, we should set $s = 1$ and use some padding to compensate for the missing entries. For all the models considered in this work, we use ReLU, given by $f(x) = \max(0, x)$, as an activation function. The learnable parameter is W, and the other parameters are defined by the user (including k and m_{out}). The integer m_{in} is the number of channels of the input and m_{out} the number of channels of the ouput. In our model, we use a convolutional neural network obtained by stacking together convolutional layers. In this context, the receptive field is defined as the number of elements of the input time series contributing to the value of one element of the output. It is traditionally considered that a good network is one that has a receptive field of the size of the largest characteristic scale (the characteristic time of the transition from A to B here), while also reducing the temporal complexity. A trade off from an optimization perspective is to have $T_{\text{out}} \ll T$ and $m_{\text{out}} \gg d$.

4.3.3 Data set for training

To train VAEs, one needs a data base of transition paths. We use the system described in Section 4.3.1 for which we generate $12,968$ trajectories. We start from a fixed initial condition $q_0 = (-1.05, -0.04)$, and integrate the dynamics using (4.2) with the potential (4.3) to construct a discrete trajectory $(q_0, q_1, \ldots, q_T)$, where $T = 1984$. We finally classify the trajectories as transitions through the bottom, transitions through the top and absence of transition. A trajectory is considered a transition if there exists $k \in \{1, ..., T\}$ such that $x_k > 0$. If y_k is greater than 0.7, the trajectory is classified as transitioning through the top, otherwise it is classified as transitioning through the bottom. The data set used to train all the models in this section is composed of 1743 transition paths (510 with a transition through the top and $1,233$ through the bottom) and $11,225$ non transition paths. We use 80% of the data to train the models, whereas 20% of the data is used for testing. We also tried to train the models without the non transition paths from the data set, but the results were very similar. Let us emphasize here that constructing a data set containing transition paths can be computationally hard and possibly infeasible (i) if β is large; (ii) if the energy barriers are too high; (iii) for problems in high dimension. However, exploring data based approaches remains interesting, at least for academic reasons.

4.3.4 "Naive" Variational AutoEncoders to generate transition paths

We first use VAEs with a bottleneck of dimension 2. The encoder is composed of a convolution block denoted by CNN-A (which is a combination of some 1–dimensional convolutional layers and batch normalization layers; see Appendix 4.A for the exact architecture). The CNN-A is stacked with an additional linear layer that produces as output $\mu(q), \sigma(q) \in \mathbb{R}^2$ (which is the output of the encoder). The decoder structure is the “transpose" structure of CNN-A (meaning that the convolutional blocks are defined with the transposed convolutional layer and the architecture parameters are the same as CNN-A but taken in the inverse order). We used the AdamW PyTorch [97] optimizer with learning rate 10^{-4} and trained the VAE for 1400 epochs with a batch size of 64.

The results presented in Figure 4.2 provide a representation of the data in the 2-dimensional latent space corresponding to the bottleneck of the VAE. We can clearly distinguish between the three types of trajectories in this 2-dimensional space: transition paths are on the outer part of the space, while non transition paths concentrate around the origin. Once the VAE is trained, one can sample new points in the 2–dimensional latent space, in order to generate new trajectories.

We first plot in Figure 4.3 some reconstructed trajectories and the original ones (from the test set) using the trained VAE. The trajectories reconstructed by the VAE have the correct shape but the magnitude of oscillations of the configurations are small compared to the original ones. The sampling of the well A is also better in the original trajectories.

We plot in Figure 4.4 some generated trajectories using points from the latent space represented in Figure 4.2 by crosses, in an attempt to obtain new trajectories by an extrapolation procedure in latent space. Although their overall shape is rather correct (except for some trajectories far from the data points in latent space), most of the generated trajectories are not really convincing. Using this VAE architecture, we do not take into account the temporal aspect of the trajectory, since everything is encoded in $\mathbb{R}^2$ (which is the dimension of the bottleneck). This bi-dimensional space is too small to account for the fluctuations of the whole trajectory. One way to deal with this issue would be to post-process the generated trajectories using transition path sampling techniques [19] to locally relax the generated paths, for instance by relying on the so-called Brownian tube proposal suggested in [142]. We take an alternative route in Section 4.3.5 and try instead to incorporate the temporal aspect in the latent space by increasing the dimension of the bottleneck.

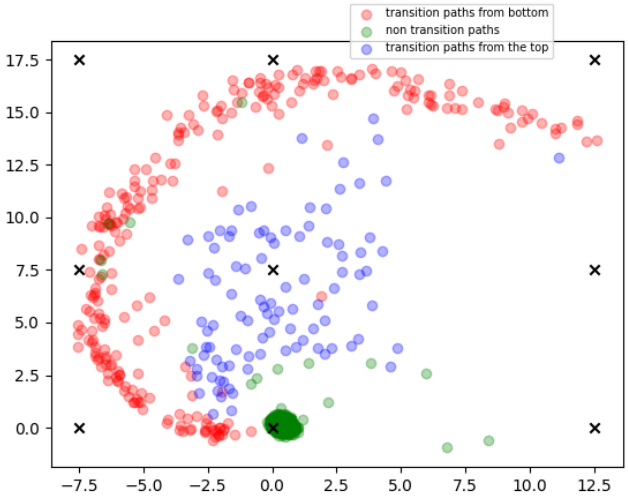


Figure 4.2 – Illustration of the mean embeddings for the test set for the "naive" VAE. The crosses represent the points used to generate new trajectories, see Figure 4.4.

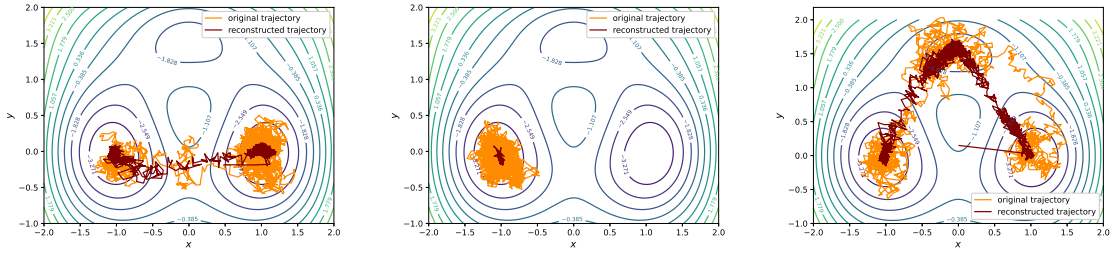
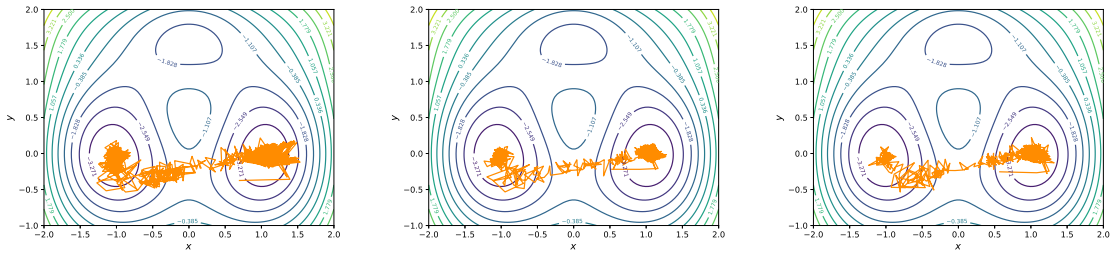
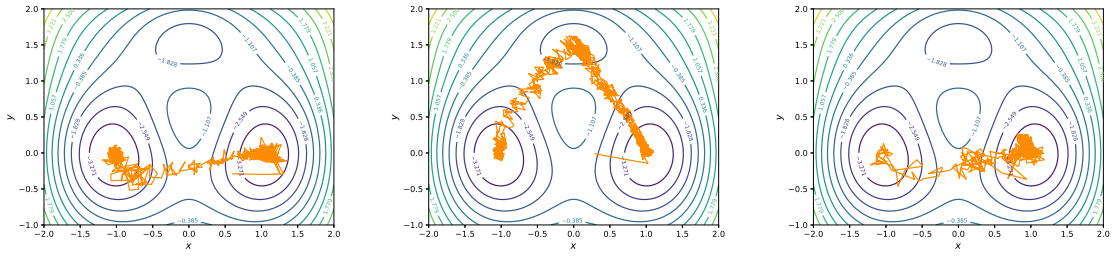


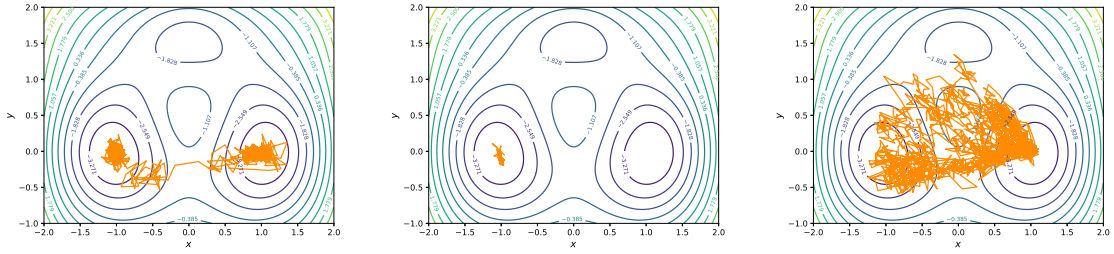
Figure 4.3 – Comparison between original and reconstructed trajectories using the trained "naive" VAE. The orange lines represent the original trajectories (from the test set) and the brown lines the reconstructed ones.



(a) $z = (-7.5, 17.5)$ (b) $z = (0, 17.5)$ (c) $z = (12.5, 17.5)$



(d) $z = (-7.5, 7.5)$ (e) $z = (0, 7.5)$ (f) $z = (12.5, 7.5)$



(g) $z = (-7.5, 0)$ (h) $z = (0, 0)$ (i) $z = (12.5, 0)$

Figure 4.4 – Generated trajectories using the trained "naive" VAE, for various values of the latent variables (see crosses in Figure 4.2).

4.3.5 VAEs with larger embedding space

To take into account the temporal aspect of the data, we use a VAE composed of the same convolutional blocks denoted by CNN-A in Appendix 4.A, and an additional linear layer as encoder to produce an output $\mu(q), \sigma(q) \in \mathbb{R}^{T_z \times 2}$, where $T_z = T/2^6 = 31$ (because we chose to work with a neural network composed of 6 convolutional layers for which the length of the input is divided by 2). This means that each trajectory is encoded by a latent variable of dimension $T_z \times 2$. The inputs of the decoder are $z = (z_1, ..., z_{T_z})$, with $z_i = \mu(q)_i + \sigma(q)_i \varepsilon_i$, where $(\varepsilon_i)_{1 \leqslant i \leqslant T_z}$ are independent and identically distributed standard 2-dimensional Gaussian random vectors. Again, the decoder is simply the "transpose" of the encoder. To train the model, we use again the AdamW PyTorch optimizer with learning rate 10^{-4} for 1400 epochs with batch size 64.

We first plot in Figure 4.5 the mean embeddings for the test set, corresponding to the scatter plot of the elements of the vectors $\mu(q)$ for each trajectory q in the test set. We can see the zone indicating the windows that are in A or in B, as well as the reactive parts of the trajectory which corresponds to the part of the transition path between the last time it leaves A and the first time it enters B. We plot in Figure 4.6 the trajectories reconstructions when passed through our VAE architecture for trajectories in the test set. The results are much better than the ones obtained in Section 4.3.4. The fluctuations in the generated trajectories are more representative of the ones of the original trajectories. The problem with

such a model is that we cannot generate new trajectories since we need a sequence $(z_1, ...z_{T_z})$ in the bottleneck space. This requires a more refined modeling of the distribution on the latent space, taking into account correlations between subsequent values of the variables z_i.

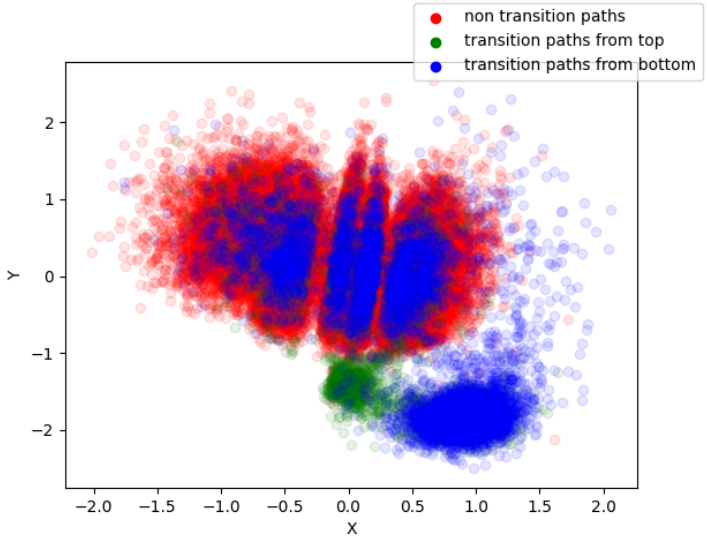


Figure 4.5 – Illustration of the mean embeddings for the test set (using the VAE with embedding space of dimension 31×2). Each trajectory is represented by 31 points.

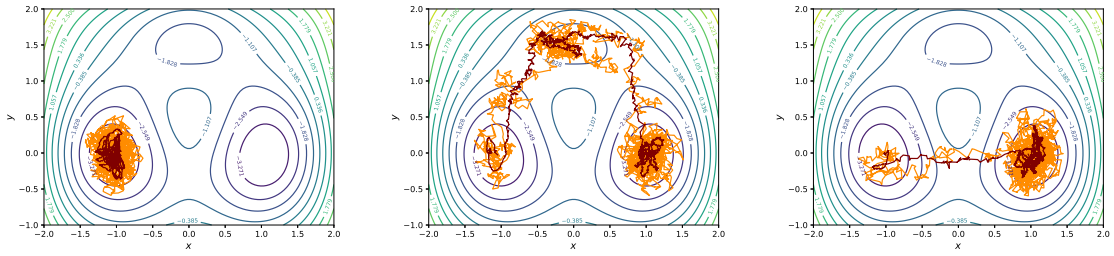


Figure 4.6 – Comparison between original and reconstructed trajectories using the trained VAE with an embedding space of dimension 31. The orange lines represent the original trajectories (from the test set) and the brown lines the reconstructed ones.

In order to do so, we tried a very recent development for learning tasks on data with a recursive structure (be it one dimensional time series as audio or two dimensional data like images): the vector quantized variational autoencoder network (VQ-VAE) [149]. The main idea of this method is to first learn a quantization of our feature space z so that each input trajectory $q_1, ..., q_T$ is represented by $z_1, ..., z_{T_z}$ where z_i is an element of a finite set of vectors $Z = \{\tilde{z}^1, ..., \tilde{z}^K\} \in \mathbb{R}^\ell$, which is also learned by the autoencoder. Once a good quantized autoencoder is available, it can be used to learn an autoregressive model, defined on the finite set of possible elements of Z. More precisely, denoting by I_i the index of the embedding for the part i of the sequence represented in Z, namely $z_i = \tilde{z}^{I_i}$, the autoregressive model learns a transformation predicting I_k from $I_1, ..., I_{k-1}$. The trajectories generated with VQ-VAE were however not convincing, which is why we do not report their results here. We also considered introducing a variational recurrent neural network [33] between the encoder and the decoder to learn the distribution on the latent space, but this also led to poor performances when generating trajectories. This motivates moving to a data free approach, as we do in Section 4.4.

4.4 Generating transition paths with reinforcement learning

In this section, we first present the main setting of reinforcement learning alongside with the Q-learning framework in Section 4.4.1. We next present in Section 4.4.2 how we apply it to our problem, and make precise in particular the reward function used to train the models. Finally, we present in Section 4.4.3 the numerical results obtained with this approach.

4.4.1 Overview of reinforcement learning

The conceptual framework behind reinforcement learning is the theory of Markov decision processes. A Markov decision process is defined by a 4-tuple $(\mathcal{Q}, \mathcal{A}, \mathcal{T}, r)$ with

- a set of states $\mathcal{Q}$;
- a set of actions $\mathcal{A}$;
- a family of transition kernels $\mathcal{T}(q_{k+1}|q_k, a_k)$ where $q_k, q_{k+1} \in \mathcal{Q}$ and $a_k \in \mathcal{A}$;
- a reward function $r_{a_k}(q_{k+1}, q_k)$ where $q_k, q_{k+1} \in \mathcal{Q}$ and $a_k \in \mathcal{A}$.

This framework allows to simulate a "game" consisting of a set of states $\mathcal{Q}$, where at each step an action a_k is selected depending on the current state q_k of the system. This action a_k leads to a transition to the state q_{k+1} with probability $\mathcal{T}(q_{k+1}|q_k, a_k)$, with an associated reward $r_{a_k}(q_{k+1}, q_k)$. In this work, actions are selected according to a deterministic policy function $P : \mathcal{Q} \to \mathcal{A}$. For a given policy P, an important function is the value function V_P, which gives the (discounted) cumulated expected reward obtained when starting from the position q_k at time k when the agent follows the policy P:

$$V_P(q_k) = \mathbb{E}_{\mathcal{T}_P}\left[\sum_{i=k}^{\infty} \gamma^{i-k} r_{P(q_i)}(q_{i+1}, q_i)\right], \tag{4.16}$$

In the latter expression, expectations are taken with respect to realizations of the Markov chain with transition kernel $\mathcal{T}_P(q_{k+1}|q_k) := \mathcal{T}(q_{k+1}|q_k, P(q_k))$, and $\gamma \in (0, 1)$ is a discount factor, which balances the importance between having a gain at time step k and having a gain in the future. We also recall the action value function (also called the Q-function), which returns the average expected reward of taking an action a_k from state q_k and then following the policy P. The action value function reads

$$Q_P(q_k, a_k) = \mathbb{E}_{\mathcal{T}(\cdot|q_k,a_k)\otimes\mathcal{T}_P}\left[r_{a_k}(q_{k+1}, q_k) + \sum_{i=k+1}^{\infty} \gamma^{i-t} r_{P(q_i)}(q_{i+1}, q_i)\right], \tag{4.17}$$

where the expectation is taken over realizations of q_{k+1} distributed according to $\mathcal{T}(\cdot|q_k, a_k)$, and $q_{i+1} \sim \mathcal{T}_P(\cdot|q_i)$ for $i \geqslant k+1$. In particular, $V_P(q_k) = Q_P(q_k, P(q_k))$. The optimal value function is defined for each state q_k by finding a policy which maximizes the value function (in particular, this policy depends a priori on the state q_k):

$$V^*(q_k) = \max_{P:\mathcal{Q}\to\mathcal{A}} V_P(q_k), \tag{4.18}$$

Similarly, the optimal action value function is defined for each state-action pair (q_k, a_k) as

$$Q^*(q_k, a_k) = \max_{P:\mathcal{Q}\to\mathcal{A}} Q_P(q_k, a_k). \tag{4.19}$$

An important issue at this stage is to find a policy which is optimal in some sense. A natural way to define a partial order on the policies is the following: $P_1 \geqslant P_2$ if $V_{P_1}(q) \geqslant V_{P_2}(q)$

for any $q \in \mathcal{Q}$. An optimal policy P^* is therefore such that $V_{P^*}(q) \geqslant V_P(q)$ for any $q \in \mathcal{Q}$. One can prove that P^* given by

$$P^*(q) = \arg\max_{a \in \mathcal{A}} Q^*(q, a), \tag{4.20}$$

is an optimal policy [121]. Furthermore, $V^* = V_{P^*}$ and $Q^* = Q_{P^*}$, see again [121] for a proof. One way to determine the optimal policy is then to solve the problem (4.20), using the expression (4.19) for $Q^*(q_k, a_k)$.

Q-Learning. When the action space is finite, a common approach for solving the optimization goal stated in (4.19) is to rely on the Hamilton–Jacobi–Bellman formulation of the Q-function, directly obtained from (4.17):

$$Q_P(q_k, a_k) = \mathbb{E}_{\mathcal{T}(\cdot|q_k, a_k)} \left[r_{a_k}(q_{k+1}, q_k) + \gamma Q_P(q_{k+1}, P(q_{k+1})) \right], \tag{4.21}$$

where the expectation is over all realizations of q_{k+1} distributed according to $\mathcal{T}(\cdot|q_k, a_k)$. For a discrete action space $\mathcal{A}$, the algorithm for Q learning is based on the fact that Q_P can be seen as a fixed point of the mapping appearing in (4.21). We briefly describe the method following [156]. An initial Q-table is considered, for instance by providing random values for the Q-function for each pair (q_k, a_k). At each time step k, the agent takes an action a_k (either randomly or by choosing the best one among the current estimates for the Q function, see [156] for more details) and moves from the state q_k to q_{k+1} with a reward $r_{a_k}(q_{k+1}, q_k)$. In view of (4.21), the Q-table is then updated as follows

$$Q(q_k, a_k) \leftarrow (1 - \alpha) Q(q_k, a_k) + \alpha \left(r_{a_k}(q_{k+1}, q_k) + \gamma \max_{a \in \mathcal{A}} Q(q_{k+1}, a) \right),$$

where α is the learning rate. The convergence of Q-learning algorithms can be shown under mild hypotheses, see [156, 143] for more details. Once the algorithm has converged and a final Q-function is obtained, the optimal policy at state q_k is defined as $\arg\max_{a \in \mathcal{A}} Q(q_k, a_k)$.

In large state spaces, either discrete or continuous, a better way to maximize (4.17) is to consider a family of functions $f(q_k, a, \theta)$ parametrized by θ to approximate the Q-function. This is more efficient in terms of computational time if the state space is discrete but large. Typically, θ are the parameters of a NN. The policy in this case would be $P_\theta(q_k) = \arg\max_{a \in \mathcal{A}} f(q_k, a, \theta)$. In practice, one iterates between updating a data set of realizations (here, snapshots of the trajectory instead of full trajectories, in fact; see Section 4.4.2 below), and the approximation of the Q function via $f(q_k, a, \theta)$. At each iteration, the algorithm consists in adding a sample of realization as

$$D_N = D_{N-1} \cup \{(q_N, a_N, q_{N+1})\},, \qquad D_0 = \emptyset,$$

and updating the parameters of the NN as

$$\theta^* = \arg\min_{\theta'} \sum_{i=1}^{N} \left[f(q_i, a_i, \theta') - \left(r_{a_i}(q_{i+1}, q_i) + \gamma \max_{a \in \mathcal{A}} f(q_{i+1}, a, \theta') \right) \right]^2. \tag{4.22}$$

The precise procedure, as well as various strategies to make the numerical method more stable, can be read in [108].

Infinite action space. When the action space is continuous, taking the maximum over the action space is impossible. One way to solve the Q-learning task, is to introduce an additional neural network: the policy network $\mathcal{P}(q_k, \omega)$, which aims at learning the optimal policy $P^*(q_k)$. The policy network is used in conjunction with the critic network $\mathcal{Q}(q_k, a_k, \theta)$ in an ensemble

called the actor-critic setting. The function $\mathcal{Q}$ is the counterpart for continuous action spaces of the function f introduced above for discrete action spaces. In this framework, one aims at solving the following optimization problem:

$$\omega^* = \arg\max_{\omega} \sum_{i=1}^{N} \mathcal{Q}(q_i, \mathcal{P}(q_i, \omega), \theta^*), \tag{4.23}$$

$$\theta^* = \arg\min_{\theta} \sum_{i=1}^{N} \left[\mathcal{Q}(q_i, \mathcal{P}(q_i, \omega^*), \theta) - \left(r_{a_i}(q_{i+1}, q_i) + \gamma \max_{a \in \mathcal{A}} \mathcal{Q}(q_{i+1}, a, \theta) \right) \right]^2. \tag{4.24}$$

Note that the policy network implicitly appears in (4.24) through the generation of the dataset.

The maximizations in (4.23)-(4.24) are more challenging than the corresponding maximization problem (4.22) for finite action spaces. Naive strategies are known to be ill behaved, leading notably to overconfident predictions on the $\mathcal{Q}$ networks (*i.e.* the resulting approximation of the Q-function gives values higher than the actual ones). Several ways of dealing with this issue have been proposed over the years. We follow the so called Twin Delayed Deep Deterministic policy gradient (TD3) algorithm defined in [50].

This algorithm introduces several tools to stabilize the learning of $\mathcal{Q}$ and $\mathcal{P}$. One of the main ideas is to replicate the $\mathcal{Q}$ network into two $\mathcal{Q}$ networks with parameters θ_1, θ_2. Taking the minimal value of $\mathcal{Q}_{\theta_1}(q, a)$ and $\mathcal{Q}_{\theta_2}(q, a)$ allows to mitigate the issue of overconfident $\mathcal{Q}$ networks mentioned above. Another important idea is to use delayed networks that are copies of $\mathcal{Q}$ and $\mathcal{P}$ but in which the weights are updated with some memory function (as an exponentially weighted linear combination of current and past weights). We refer to the original paper [50] for a more in-depth discussion of these points.

4.4.2 Application to sampling transition paths

We now describe how to adapt the reinforcement learning framework previously described to the problem of sampling transition paths described in Section 4.2. We successively define all the elements of the Markov decision process introduced in Section 4.4.1.

State space and action space. The state space is simply defined as the space $\mathcal{D}$ of the diffusion process, $\mathcal{D} \subset \mathbb{R}^d$. The action space is the same space as the image of the gradient of the potential, *i.e.* $\mathbb{R}^d$. In this context, we consider the policy to be a vector field, introduced as a bias into the governing equation (4.1) to alter the trajectory. This is similar to the choice made in [136, 57] where controlled stochastic differentials are considered. More explicitly, the controlled version of (4.1) we consider reads

$$dq_t = (-\nabla V(q_t) + P(q_t))\, dt + \sqrt{\frac{2}{\beta}}\, dW_t, \tag{4.25}$$

discretized in the actor-critic framework as

$$q_{k+1} = q_k + (-\nabla V(q_k) + \mathcal{P}(q_k, \omega))\Delta t + \sqrt{\frac{2\Delta t}{\beta}} G_k, \tag{4.26}$$

where ω denotes the state of the policy network. We chose here to work with a generic policy, although we could have looked for it in gradient form according to results of stochastic optimal control (as reviewed in [94, Section 6.2]). We also denote by $\mathcal{T}_{\mathcal{P}}$ the corresponding transition kernel in the sequel (not explicitly writing out the state ω of the neural network).

Probability kernel. Given a policy $\mathcal{P}(\cdot, \omega)$, the probability kernel to go from a configuration q_k to a new one q_{k+1} can be deduced from (4.26) to be

$$\mathcal{T}_P(q_{k+1}|q_k) = \left(\frac{\beta}{4\pi\Delta t} \right)^{d/2} \exp\left(-\beta \frac{\|q_{k+1} - q_k - \Delta t(-\nabla V(q_k) + \mathcal{P}(q_k, \omega))\|^2}{4\Delta t} \right). \tag{4.27}$$

Reward function. We want to maximize the likelihood of trajectories leaving the well A. The reward function we consider to this end reads (omitting the dependence on the action in the notation)

$$r(q_{k+1}, q_k) = \log\left(\frac{\mathcal{T}_0(q_{k+1}|q_k)}{\mathcal{T}_{\mathcal{P}}(q_{k+1}|q_k)}\right) + \alpha h(q_k, q_0), \tag{4.28}$$

where $\alpha \geqslant 0$ and $h(\cdot, q_0)$ is a function which measures the distance of the current configuration to the center q_0 of the well A; for instance, $h(q_k, q_0) = \|q_k - q_0\|^2$. The first term of the reward function compensates for the bias introduced in the dynamics by computing the relative likelihood with respect to an unbiased evolution. The second term of the reward function forces the particle to leave the well A. A simple computation gives

$$\begin{aligned}\log\left(\frac{\mathcal{T}_0(q_{k+1}|q_k)}{\mathcal{T}_{\mathcal{P}}(q_{k+1}|q_k)}\right) &= \frac{\beta}{4\Delta t}\left[\|q_{k+1} - q_k - \Delta t(-\nabla V(q_k) + \mathcal{P}(q_k,\omega))\|^2 - \|q_{k+1} - q_k + \Delta t \nabla V(q_k)\|^2\right] \\ &= \frac{\beta}{4\Delta t}\left[-2\Delta t(q_{k+1} - q_k)\cdot \mathcal{P}(q_k,\omega) + \Delta t^2(\|\mathcal{P}(q_k,\omega)\|^2 - 2\nabla V(q_k)\cdot \mathcal{P}(q_k,\omega))\right].\end{aligned} \tag{4.29}$$

Using equation (4.26),

$$\log\left(\frac{\mathcal{T}_0(q_{k+1}|q_k)}{\mathcal{T}_P(q_{k+1}|q_k)}\right) = \frac{\beta}{2}\left[-\sqrt{2\Delta t\beta^{-1}}G_k \cdot \mathcal{P}(q_k,\omega) - \Delta t^2\|P(q_k,\omega)\|^2\right]. \tag{4.30}$$

Note that, when summing up over k the contributions from the transitions between q_k and q_{k+1}, one ends up with a discretization of the logarithm of the Girsanov weight allowing to compare the path probabilites between two dynamics differing in their drifts.

Remark 4.1. *Other choices for the function h can be considered. In particular, we tried $h(q_k, q_0) = \log(\|q_k - q_0\|^2)$ but the results were less convincing than with the choice $h(q_k, q_0) = \|q_k - q_0\|^2$. Other works consider reward functions favoring that the final state is in B, rather favoring exits out of A as we do, but this requires knowing in advance the target metastable region [36].*

4.4.3 Numerical results

We trained neural networks using the TD3 Algorithm [50] to generate parameters $(\theta_{\mathrm{f}}, \omega_{\mathrm{f}})$ which are approximations of the optimal parameters (θ^*, ω^*) in (4.23)-(4.24). We use the open source code provided by the authors of [50], available at `https://github.com/sfujim/TD3`. We fix $\Delta t = 5 \times 10^{-3}$ and $\alpha = 0.071$. Precise information on hyper parameters and network architectures can be found in Appendices 4.B and 4.C.

The final actor network obtained is depicted in Figure 4.7 where we plot the final policy $\mathcal{P}(q, \omega_{\mathrm{f}})$ alongside with the final value function. We can see that the drift corresponding to the policy network biases the trajectories towards the saddle point of the potential in the vicinity of $(0, 1)$. It does not bias the trajectories towards the saddle point around $(0, -0.25)$, and even discourages them from performing a transition from A to B. This therefore biases transitions towards transitions through the upper channel. Note also that the closer the current state is to the well B, the smaller the drift is, and therefore the closer the evolution is to the true dynamics. We were not able to produce transitions through the bottom saddle point.

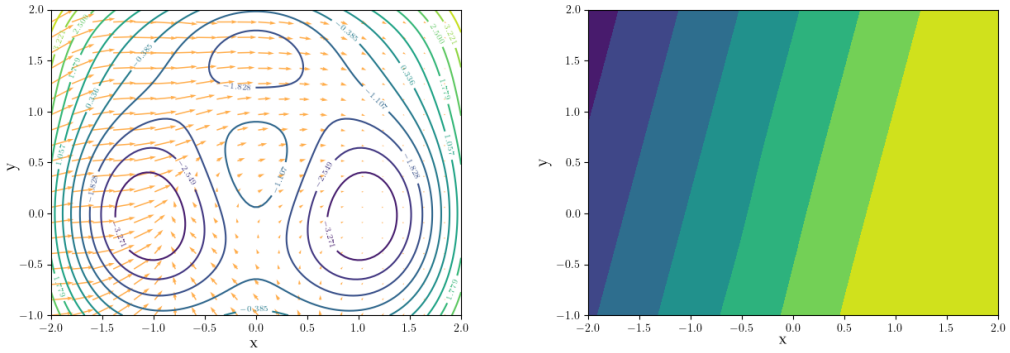


Figure 4.7 – Left: Visualization of the policy $\mathcal{P}(q, \omega_{\mathrm{f}})$. The action field "pushes" the simulation from A through the upper saddle point and diminishes in magnitude in the neighborhood of the second metastable region. Right: Visualization of the critic value at a given position for the optimal action, namely $\mathcal{Q}(q, \mathcal{P}(q, \omega_{\mathrm{f}}), \theta_{\mathrm{f}})$.

In order to generate new transitions, we initialize the system at $(-1, 0)$ and then run the discrete dynamics (4.26) using the final policy $\mathcal{P}(\cdot, \omega_{\mathrm{f}})$ for 600 time steps with $\Delta t = 5 \times 10^{-3}$. We generated 1000 trajectories, and observed transitions from one metastable state to the other for more than 99% of the trajectories. Some of these trajectories are plotted in Figure 4.8. They are visually more realistic than those produced in Figures 4.3 and 4.6. Another advantage of the method is that, once the network is trained, there is no need to sample a latent variable to generate a new trajectory, as for example in Figure 4.6.

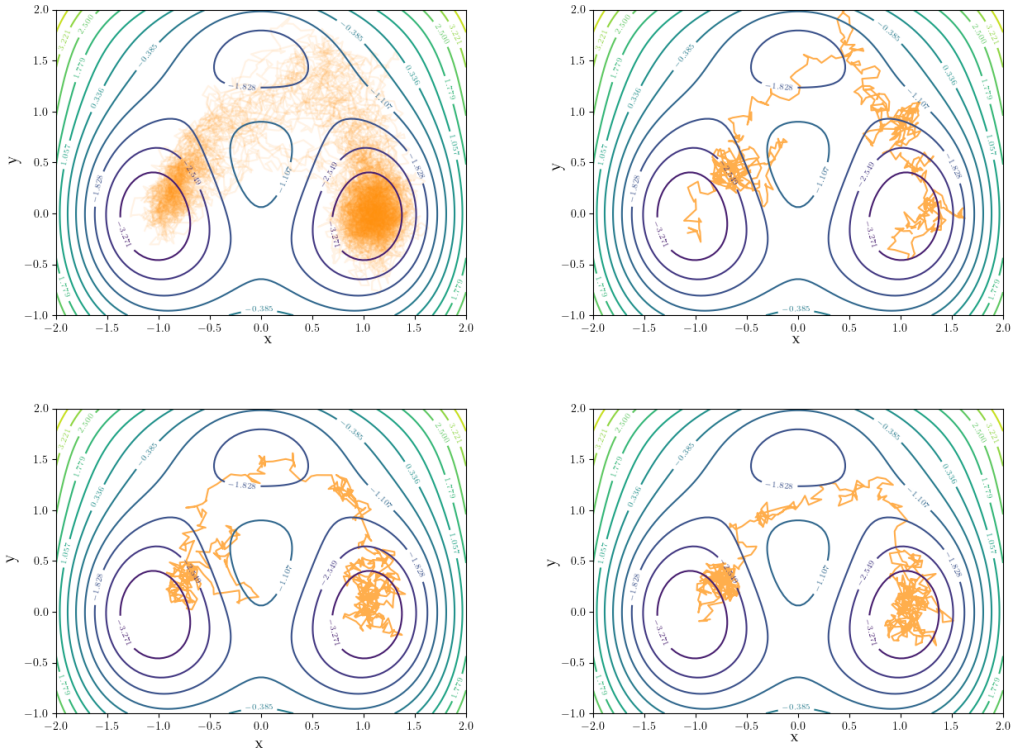


Figure 4.8 – Sample trajectories generated using (4.26) (one per picture, except the top left in which 25 trajectories are represented).

4.5 Discussion and perspectives

The results presented in this work suggest that reinforcement learning provides a way to sample transition paths by finding some biasing force field guiding the system in its excursion out of A and into B. Approaches based on generative methods such as variational autoencoders are intrinsically more limited, especially given that they need a database of transition paths to start with. Even when such a database is available, the generation of new paths may be cumbersome as this requires a large latent space with some structure to reproduce the temporal organization of the components of the latent variables.

While writing up this work, we became aware of recent works in the computational statistical physics community making use of reinforcement learning and neural networks to construct effective biases and favor otherwise unlikely transitions (in particular, one work where reinforcement learning is used to obtain transition paths but not using neural networks [36], and one work making use of neural networks to compute rare events but not in the reinforcement learning context [160]). We believe that further efforts are required to better understand various choices in the reinforcement learning procedure, in particular the reward function. From a theoretical perspective, this calls for a better understanding of the links between reinforcement learning and optimal importance sampling in path space (see for instance [94, Section 6.2] and references therein).

Appendix

4.A Architecture of CNN-A used in Section 4.3

The following sequence of block parameters was used for the encoder network CNN-A, for an input of size $T \times d$ with $d = 2$:

- layer 1: $m_{\text{in}}^1 = 2$, $m_{\text{out}}^1 = 30$, kernel size $k = 4$, stride $s = 2$, $T_{\text{out}}^1 = T/2$.
- batch normalization layer.
- layer 2: $m_{\text{in}}^2 = 30$, $m_{\text{out}}^2 = 20$, kernel size $k = 4$, stride $s = 2$, $T_{\text{out}}^2 = T_{\text{out}}^1/2$.
- batch normalization layer.
- layer 3: $m_{\text{in}}^3 = 20$, $m_{\text{out}}^3 = 15$, kernel size $k = 4$, stride $s = 2$, $T_{\text{out}}^3 = T_{\text{out}}^2/2$.
- batch normalization layer.
- layer 4: $m_{\text{in}}^4 = 15$, $m_{\text{out}}^4 = 10$, kernel size $k = 4$, stride $s = 2$, $T_{\text{out}}^4 = T_{\text{out}}^3/2$.
- batch normalization layer.
- layer 5: $m_{\text{in}}^5 = 10$, $m_{\text{out}}^5 = 20$, kernel size $k = 4$, stride $s = 2$, $T_{\text{out}}^5 = T_{\text{out}}^4/2$.
- batch normalization layer.
- layer 6: $m_{\text{in}}^6 = 20$, $m_{\text{out}}^6 = 20$, kernel size $k = 2$, stride $s = 2$, $T_{\text{out}}^6 = T_{\text{out}}^5/2$.

The batch normalization layer standardizes the inputs to a layer for each mini-batch to stabilize the learning. The properties of the CNN:

- receptive field: 125
- jump between two consecutive starts: 64
- overlap: 61 at each extremity

The length of the output time series is $T/32$, with $T = 1984$.

4.B Architecture of the neural networks used for TD3 algorithm

The actor network $\mathcal{P}(\cdot, \omega)$ had the following architecture:

- layer 1: Linear(2, 128) + ReLU
- layer 2: Linear(128, 256) + ReLU
- layer 3: Linear(256, 2) + Tanh

The output of the network was then multiplied by a user specified maximum value $c_{\max}$. In our experiments we set $c_{\max} = 10$. The critic network $\mathcal{Q}(\cdot, \cdot, \theta)$ had the following architecture:

- Layer 1: Linear(4, 256) + ReLU
- Layer 2: Linear(256, 256) + ReLU
- Layer 3: Linear(256, 1)

4.C Parameters for the TD3 algorithm

The TD3 algorithm used in this work is the one described in [50], available at the following link: https://github.com/sfujim/TD3/blob/master/TD3.py. We have used the following parameters for the code:

- $\tau = 5 \times 10^{-2}$
- discount: 0.99
- policy_noise: 0.2
- policy_frequency: 60
- max_action: 10
- noise_clip: 0.5
- action_dim: 2
- state_dim: 2
- learning rate: 3×10^{-4}

The parameters for the training (game related) are the following:

- dataset maximum size: 30000
- number of games: 50000
- batch size: 512
- number of rounds per game $T = 600$
- train periodicity (in rounds per game): $d = 100$
- exploration noise: 0.1
- $\Delta t = 5 \times 10^{-3}$
- Probability of random step decay coefficient: 0.99
- Random step decay period (in rounds per game): 2000

The optimization routine used was the Adam optimization algorithm [70] from the PyTorch library.

CHAPTER 5

RÉSUMÉ DE LA THÈSE EN FRANÇAIS

Le but de cette section est de fournir un résumé en français de la thèse. Pour chacun des problèmes considérés, on commence par introduire les motivations puis on donne un bref résumé des contributions.

5.1 Réduction systématique de l'erreur de minibatching dans l'inférence Bayésienne à l'aide de la dynamique de Langevin adaptative

5.1.1 Motivation pour l'inférence Bayésienne

Commençons par présenter d'abord le contexte de l'inférence Bayésienne [127, 84]. Considérons un jeu de données contenants N_{data} points indépendants et identiquement distribués (i.i.d.) qu'on note $\mathbf{x} = (x_1, ..., x_{N_{\text{data}}}) \in (\mathbb{R}^{d_{\text{data}}})^{N_{\text{data}}}$. Nous supposons que les éléments du jeu de données sont distribués selon une mesure de probabilité paramétrée par un vecteur $\theta \in \mathbb{R}^d$ et qu'on note $P_{\text{elem}}(\cdot|\theta)$. La vraisemblance de l'ensemble de données est alors donnée par

$$P_{\text{likelihood}}(\mathbf{x}|\theta) = \prod_{i=1}^{N} P_{\text{elem}}(x_i|\theta).$$

L'idée principale de l'approche Bayésienne est de considérer que le vecteur de paramètres θ est lui-même une variable aléatoire, et qu'en capturant son incertitude/dispersion, on peut éviter un ajustement excessif à l'ensemble des données. On commence par fixer une distribution a priori sur le vecteur des paramètres, notée $P_{\text{prior}}(\theta)$, qui exprime l'information initiale sur les paramètres. En utilisant la formule de Bayes, la distribution a postériori $\pi(\theta|\mathbf{x})$ du vecteur de paramètres θ est donnée par

$$\pi(\theta|\mathbf{x}) = \frac{P_{\text{prior}}(\theta) P_{\text{likelihood}}(\mathbf{x}|\theta)}{Z}, \qquad Z = \int_{\mathbb{R}^d} P_{\text{prior}}(\theta) P_{\text{likelihood}}(\mathbf{x}|\theta)\, d\theta, \tag{5.1}$$

où Z est la constante de normalisation. Dans la plupart des cas, cette constante est difficile à calculer. Par exemple, lorsque d est grand, les méthodes standards de quadrature ne peuvent pas être utilisées pour calculer cette quantité. L'objectif principal de l'approche Bayésienne est d'échantillonner la mesure de probabilité π. Les quantités d'intérêt sont les espérances par rapport à la mesure cible π. Dans la section 1.2, nous rappelons quelques méthodes d'echantillonnage, qui constituent une base pour la version étendue de la dynamique de Langevin adaptative [137] introduite dans le Chapitre 2. Utiliser des algorithmes de

Markov chain Monte Carlo (MCMC) pour échantillonner π nécessite généralement le calcul de $\nabla_\theta \log \pi(\cdot|\mathbf{x})$ à chaque itération, ce qui a un coût de calcul élevé. L'approximation de ce terme par minibatching permet de limiter le coût de calcul comme discuté dans la Section 1.3.

5.1.2 Contributions

Ce travail, présenté dans le Chapitre 2, est pré-publié dans [137] et soumis au *Journal of Machine Learning Research*.

Le minibatching est un moyen simple de réduire efficacement le temps de calcul des algorithmes MCMC basés sur la discrétisation des équations différentielles stochastiques (voir Section 1.3.1 pour plus de détails). L'idée est de remplacer le terme $\nabla_\theta \log \pi(\cdot|\mathbf{x})$ par un estimateur stochastique donné par

$$\begin{aligned}\widehat{F_n}(\theta) &= \nabla_\theta(\log P_{\text{prior}}(\theta)) + \frac{N_{\text{data}}}{n}\sum_{i\in I_n}\nabla_\theta(\log P_{\text{elem}}(x_i|\theta)),\\ &= \nabla_\theta(\log \pi(\theta|\mathbf{x})) + \sqrt{\varepsilon(n)}\Sigma_{\mathbf{x}}^{\frac{1}{2}}(\theta)Z_{\mathbf{x},N_{\text{data}},n}.\end{aligned} \tag{5.2}$$

Les termes $\varepsilon(n)$, $\Sigma_{\mathbf{x}}$ et $Z_{\mathbf{x},N_{\text{data}},n}$ sont détaillés dans le Chapitre 1. Le bruit introduit par l'estimateur du gradient est encodé dans le terme

$$\sqrt{\varepsilon(n)}\Sigma_{\mathbf{x}}^{\frac{1}{2}}(\theta)Z_{\mathbf{x},N_{\text{data}},n}.$$

Pour ce travail, nous nous plaçons dans le cas où $Z_{\mathbf{x},N_{\text{data}},n}$ n'est pas nécessairement une variable aléatoire Gaussienne standard. Le Chapitre 2 est d'abord consacré à l'analyse numérique de l'erreur de minibatching dans le cadre de l'inférence Bayésienne pour les algorithmes de type Langevin. La première contribution est le raffinement du premier ordre de l'erreur faible sur la mesure invariante. En particulier, nous montrons que le biais sur la mesure invariante est d'ordre $\mathcal{O}((1+\varepsilon(n))\Delta t)$ pour l'algorithme du "Stochastic gradient Langevin dynamics" et d'ordre $\mathcal{O}((\Delta t+\varepsilon(n))\Delta t)$ pour les discrétisations de second ordre de la dynamique de Langevin.

La deuxième partie du Chapitre 2 est consacrée à la dynamique du Langevin adaptative (AdL) introduite dans la Section 1.2.2.4. Nous contestons d'abord numériquement l'hypothèse selon laquelle $\Sigma_{\mathbf{x}}$ est constant, auquel cas la procédure de minibatching n'induit aucun biais sur la distribution a posteriori π lors de l'utilisation de la discrétisation d'AdL. Nous fournissons une analyse de l'erreur de minibatching pour la discrétisation d'AdL dans le cas où $\Sigma_{\mathbf{x}}$ dépend de θ, montrant que là encore l'erreur est linéaire en $\varepsilon(n)$ à condition que $\varepsilon(n)\Delta t$ soit suffisamment petit. L'expression de l'erreur peut être résumée comme suit

$$\varepsilon(n)\Delta t\|\Sigma_{\mathbf{x}} - S^*\|_{L^2(\pi)},$$

où

- $S^* = \frac{1}{d}\overline{\Sigma}_{\mathbf{x}} = \int_\Theta \Sigma_{\mathbf{x}}(\theta)\pi(\theta|\mathbf{x})\,d\theta$ pour AdL matriciel ;
- S^* est une matrice diagonale avec des entrées $\int_\Theta [\Sigma_{\boldsymbol{x}}(\theta)]_{i,i}\,\pi(\theta|\boldsymbol{x})\,d\theta$, où $1 \leqslant i \leqslant d$ pour AdL diagonal.
- $S^* = \frac{1}{d}\text{Tr}(\overline{\Sigma}_{\mathbf{x}})\text{I}_d$ pour l'AdL scalaire.

5.2 Réseaux de neurones Bayésiens

5.2.1 Motivations pour les réseaux de neurones Bayésien

Le deuxième problème que nous considérons est la simulation de réseaux de neurones Bayésiens (BNNs). Notons $\mathbf{y} = (y_1, ..., y_{N_{\text{data}}}) \in \mathcal{Y}^{N_{\text{data}}}$ les labels associés à un ensemble de données $\mathbf{x} = (x_1, ..., x_{N_{\text{data}}}) \in \mathcal{X}^{N_{\text{data}}} = (\mathbb{R}^{d_{\text{data}}})^{N_{\text{data}}}$. En général, on peut distinguer deux types de problèmes. Premièrement, les problèmes de classification, où chaque entrée x_i est classée dans l'une de deux ou plusieurs classes. La classification binaire correspond au cas où $\mathcal{Y}^{N_{\text{data}}} = \{0, 1\}^{N_{\text{data}}}$ (0 correspondant à la première classe et 1 à la seconde). La classification multi-label correspond au cas où $\mathcal{Y}^{N_{\text{data}}} = \{0, ..., d_{\text{label}}\}^{N_{\text{data}}}$, où d_{label} est le nombre de classes. Le deuxième type de problèmes sont les régressions où les labels $\mathbf{y}$ sont des éléments continus. Dans ce cas, $\mathcal{Y}^{N_{\text{data}}} = (\mathbb{R}^{d_{\text{label}}})^{N_{\text{data}}}$, où d_{label} représente la dimension des labels.

Nous supposons que les éléments de l'ensemble de données sont des échantillons indépendants d'une certaine distribution inconnue qu'on note ϕ, à savoir $(x_i, y_i) \sim \phi$ pour tout $1 \leqslant i \leqslant N_{\text{data}}$. En apprentissage automatique, l'objectif de l'apprentissage supervisé est, étant donné un ensemble de données $(\mathbf{x}, \mathbf{y})$, construire un prédicteur qui, à partir d'une nouvelle entrée x, prédit une sortie y. L'idée est d'apprendre une fonction $f : \mathcal{X} \to \mathcal{Y}$, qui lie les entrées et les sorties, de sorte que l'espérance par rapport à ϕ d'une certaine fonction de perte, désignée par L_f, soit minimisée. La fonction de perte quantifie la qualité de la prédiction fournie par f. Étant donné que la distribution ϕ est inconnue, et que nous n'avons accès qu'à un échantillon de ϕ, une approximation raisonnable de l'espérance de la fonction de perte par rapport à la distribution ϕ est donnée par

$$\mathbb{E}_\phi[L_f(X, Y)] \approx \frac{1}{N_{\text{data}}} \sum_{i=1}^{N_{\text{data}}} L_f(x_i, y_i).$$

Cette approximation est justifiée par la loi des grands nombres. Par exemple, dans un problème de régression, on peut considérer la fonction de perte des moindres carrés, donnée par

$$L_f(x, y) = \frac{1}{2}(f(x) - y)^2, \tag{5.3}$$

pour $x \in \mathcal{X}$ et $y \in \mathcal{Y}$. Pour un problème de classification binaire, une fonction de perte adaptée est l'entropie relative binaire, donnée par

$$L_f(x, y) = y \log(f(x)) + (1 - y) \log(1 - f(x)), \tag{5.4}$$

pour $x \in \mathcal{X}$ et $y \in \mathcal{Y}$ (voir Section 1.1.2 pour une explication de cette fonction). Pour la classification multi-label, on peut utiliser la généralisation donnée par l'entropie relative catégorique. Dans ce cas, les résultats du prédicteur sont les probabilités d'appartenir à chacune des classes.

Les réseaux de neurones [81, 82, 16] sont une classe de fonctions (prédicteurs) qui ont montré une grande capacité à résoudre les problèmes d'apprentissage automatique. Ils sont définis comme des fonctions $N_\theta : \mathcal{X} \to \mathcal{Y}$ paramétrées par $\theta \in \mathbb{R}^d$. Dans ce travail, nous nous concentrons sur les réseaux de neurones feed-forward entièrement connectés. Ceux-ci peuvent être définis comme des compositions successives de transformations linéaires et de fonctions d'activation non-linéaires. Un réseau de neurones est typiquement composé de K couches cachées. Chaque couche $\ell \in \{1, ..., K\}$ prend en entrée $x_{\text{in},\ell} \in \mathbb{R}^{d_\ell}$ et produit une sortie $x_{\text{out},\ell} \in \mathbb{R}^{d_{\ell+1}}$ telle que

$$x_{\text{out},\ell} = \sigma(b_\ell + W_\ell x_{\text{in},\ell}),$$

ou $b_\ell \in \mathbb{R}^{d_{\ell+1}}$ correspond au biais, $W_\ell \in \mathbb{R}^{d_{\ell+1} \times d_\ell}$ correspond au poids et $\sigma : \mathbb{R}^{d_{\ell+1}} \to \mathbb{R}^{d_{\ell+1}}$ est une fonction non–linéaire appelée fonction d'activation. Par exemple, on peut utiliser

la fonction linéaire redressée ReLU, qui, pour un vecteur $z \in \mathbb{R}^{d_{\ell+1}}$, agit composante par composante comme suit

$$(\sigma(z))_i = \max(0, z_i), \tag{5.5}$$

pour $i \in \{1, ..., d_{\ell+1}\}$. Il est clair que pour $\ell = 1$, la dimension de l'entrée est $d_1 = d_{\text{data}}$ de sorte que la matrice W^1 devrait avoir d_{data} colonnes. La dernière couche dépend du problème à résoudre : (i) pour un problème de régression, $d_{K+1} = d_{\text{label}}$; (ii) pour un problème de classification binaire $d_{K+1} = 1$; (iii) pour un problème de classification multi-labels $d_{K+1} = d_{\text{label}}$. Le réseau N_θ peut être défini récursivement comme suit

$$\begin{cases} z_1 = x, \\ z_k = \sigma(b_{k-1} + W^{k-1} z_{k-1}), \quad 2 \leqslant k \leqslant K \\ N_\theta(x) = \sigma_K(b_K + W^K z_K). \end{cases} \tag{5.6}$$

Les paramètres du réseau de neurones sont composés de toutes les matrices et les biais, *i.e.* $\theta = (W^\ell, b^\ell)_{1 \leqslant \ell \leqslant K}$. Pour les réseaux de neurones, nous notons la fonction de perte par $L(x, y, \theta)$ au lieu de $L_{N_\theta}(x, y)$. Dans ce contexte, une façon de déterminer un bon prédicteur consiste à résoudre le problème d'optimisation suivant

$$\theta^* = \arg\min_{\theta \in \mathbb{R}^d} \mathbb{E}_\phi[L(X, Y, \theta)].$$

Optimisation. Une approximation raisonnable de θ^* consiste à minimiser la perte empirique, le problème devient donc

$$\theta^* \approx \widehat{\theta}^*_{N_{\text{data}}} = \arg\min_{\theta \in \mathbb{R}^d} \frac{1}{N_{\text{data}}} \sum_{i=1}^{N_{\text{data}}} L(x_i, y_i, \theta). \tag{5.7}$$

Une fois les paramètres du réseau de neurones définis, pour une entrée x donnée, une prédiction du label est donnée par $y = N_{\widehat{\theta}^*_{N_{\text{data}}}}(x)$, où, par abus de notation, nous désignons par $\widehat{\theta}^*_{N_{\text{data}}}$ une solution approximative du problème (5.7). L'un des principaux problèmes rencontrés lors de l'étude des prédicteurs basés sur (5.7) est le surajustement. Les performances du réseau peuvent être très faibles en dehors de l'ensemble d'apprentissage $\mathbf{x}$, même lorsque le problème (5.7) est parfaitement résolu.

Echantillonnage. Le paradigme Bayésien a été introduit dans le cadre des réseaux de neurones comme une alternative à l'optimisation pour éviter le surajustement aux données [111]. L'idée est de considérer les paramètres du réseau de neurones comme des variables aléatoires et d'inférer leur distribution a posteriori $\pi(\theta|\mathbf{x}, \mathbf{y})$ qui est donnée, en considérant la formule de Bayes, par

$$\pi(\theta|\mathbf{x}, \mathbf{y}) \propto P_{\text{prior}}(\theta) \prod_{i=1}^{N} P_{\text{elem}}(y_i, x_i|\theta), \tag{5.8}$$

où $P_{\text{prior}}(\theta)$ est la distribution a priori du vecteur des paramètres et $P_{\text{likelihood}}$ la vraisemblance des données, donnée en terme de fonction de perte par

$$P_{\text{elem}}(x, y|\theta) \propto \exp(-L(x, y, \theta)).$$

Dans ce contexte, pour une entrée donnée x, le label est prédit pour un problème de classification binaire basé sur la quantité

$$p(y|x, \mathbf{x}) = \int_\Theta N_\theta(x) \pi(\theta|\mathbf{x}) \, d\theta,$$

comme

$$y = \begin{cases} 1 & \text{si} \quad p(y|x, \mathbf{x}) \geqslant 1/2, \\ 0 & \text{si} \quad p(y|x, \mathbf{x}) < 1/2. \end{cases} \tag{5.9}$$

Comme dans la Section 5.1.1, le but ici est d'échantillonner la densité de probabilité a posteriori et de calculer des moyennes par rapport à celle-ci. Nous renvoyons à la Section 1.2 pour un aperçu de certaines méthodes d'échantillonnage, et au Chapitre 3 pour une analyse de l'erreur de minibatching sur la distribution a posteriori à travers l'analyse numérique de la matrice de covariance de l'estimateur stochastique du gradient et l'utilisation de l'AdL pour échantillonner la mesure π dans le cadre des réseau de neurones bayésien.

5.2.2 Contributions

Ce travail a débuté à l'université d'Edimbourg lors d'un programme de mobilité doctorale de deux mois, à travers des interactions avec Ben Leimkuhler et Tiffany Vlaar.

Dans le Chapitre 3, nous abordons l'échantillonnage des distributions a posteriori ans le cadre de réseaux de neurones Bayésiens. Suite aux résultats du Chapitre 2, nous commençons par analyser numériquement la matrice de covariance de l'estimateur stochastic du gradient. Pour cela, nous considérons deux modèles numériques : (i) un modèle de classification jouet ; (ii) un modèle de données spirales jouet [88]. Les principales conclusions obtenues par les résultats numériques peuvent être résumées comme suit :

- La moyenne de la matrice de covariance $\Sigma_{\mathbf{x}}$ semble être grossièrement isotrope, ce qui suggère que les versions scalaire, diagonale et matricielle de l'AdL produiront le même biais sur la mesure invariante ;
- La matrice de covariance est de rang faible, ce qui suggère qu'elle peut être efficacement approximée de manière peu coûteuse. Cela ouvre la voie à une alternative à l'AdL matriciel, où au lieu d'apprendre la moyenne de $\Sigma_{\mathbf{x}}$, on apprendrait les premières valeurs propres de celle-ci, réduisant ainsi le biais sans recourir à une friction matricielle (ce qui est infaisable dans le cadre des réseaux de neurones).
- La forme particulière de la matrice de covariance associée à la dernière couche suggère d'utiliser AdL uniquement pour cette dernière. Notez que l'échantillonnage d'uniquement la dernière couche a déjà été suggéré dans [77].

5.3 Méthodes génératives pour l'échantillonnage de chemins de transition en dynamique moléculaire

5.3.1 Motivations

Le dernier problème que nous considérons dans ce manuscrit est l'échantillonnage de certaines trajectoires de dynamiques stochastiques. Considérons une équation différentielle stochastique donnée par

$$\{ dy_t = b(t, y_t)\, dt + \sigma(t, y_t) dW_t, \tag{5.10}$$

avec y_0 fixé, où $y_t \in \mathbb{R}^d$, $b : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^d$, $\sigma : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^{d \times d'}$ et $W_t \in \mathbb{R}^{d'}$ est un processus de Wiener standard de dimension d'. Sous certaines conditions sur b et σ, on peut prouver l'existence d'une solution unique de (1.11) (voir par exemple [69, 123]).

Une trajectoire est définie comme étant une réalisation du processus stochastique $(y_t)_{0 \leqslant t \leqslant T}$. Pour certains choix de b et de σ, qui seront explicités d'abord dans la section 1.2.2.2 et avec plus de détails dans le Chapitre 4, une trajectoire telle que $y_0 \in A \subset \mathbb{R}^d$ et $y_T \in B \subset \mathbb{R}^d$ peut être un événement rare, et peut donc être difficile à simuler en pratique. Nous considérons

le cas où σ est constant et $b = -\nabla_\theta V$, où V représente une certaine fonction d'énergie potentielle. Dans ce cas, le système a tendance à rester piégé dans certaines régions de l'espace des phases, à savoir au voisinage des minima locaux de V. Typiquement, A et B seraient les voisinages de deux minima locaux distincts de V. Le but est alors de simuler efficacement des trajectoires de transition, définis comme des trajectoires qui, à partir d'une condition initiale fixe y_0 située dans le puits de potentiel initial A, atteignent l'ensemble B avant le temps $T \geqslant 0$.

Ce problème a attiré beaucoup d'attention et de nombreuses méthodes ont été développées pour échantillonner efficacement les trajectoires de transition. On peut distinguer deux grandes classes : les techniques d'échantillonnage d'importance [47, 25, 94] et les techniques de fractionnement [34, 8, 28, 29]. Nous avons appliqué des méthodes génératives issues de la littérature d'apprentissage automatique pour échantillonner les trajectoires de transition. Plus de détails sont donnés dans la section 4.1 (avec un résumé des contributions de cette thèse dans la section 1.4.3).

5.3.2 Contributions

Ce travail a débuté pendant l'école d'été CEMRACS au CIRM. Il est pré-publié dans [92] et soumit.

Nous considérons la discrétisation de la dynamique de Langevin suramortie dans le contexte de la dynamique moléculaire par le schéma Euler–Maruyama. Nous nous plaçons dans le cas où le potentiel V a de nombreux minima locaux. Un problème principal dans cette situation est la métastabilité : la trajectoire reste piégée dans des sous-domaines restreints de l'espace des phases, empêchant un échantillonnage précis. La trajectoires de chemins de transition est particulièrement intéressante. Prenons par exemple la Figure 5.1 où nous traçons deux trajectoires typiques en utilisant la discrétisation Euler–Maruyama de la dynamique de Langevin suramortie avec une fonction potentielle définie en (4.3) introduite dans le Chapitre 4. La trajectoire orange n'est pas une trajectoire de transition car elle reste piégée dans le premier puits A ; alors que la trajectoire rouge passe du puits A au puits B et est donc une trajectoire de transition. De telles trajectoires sont rares, et d'autant plus rares quand la dimension est grande.

L'objectif du Chapitre 4 est d'utiliser des modèles génératifs issus de la littérature de l'apprentissage automatique pour générer des trajectoires de transition. Nous commençons par considérer des approches génératives basées sur ll'apprentissage à partir de données, à savoir des autoencodeurs variationnels, d'abord avec un espace latent bi-dimensionnel, qui s'est avéré trop petit pour rendre compte des fluctuations de la trajectoire entière, conduisant à des trajectoires trop lisses (qu'elles soient reconstruites ou générées). L'incorporation de l'aspect temporel dans l'espace latent permet d'obtenir de meilleures trajectoires reconstruites. Cependant, la génération de nouvelles trajectoires nécessite une modélisation raffinée de la distribution dans l'espace latent, ce qui conduit à des résultats pas très convaincants. Nous nous tournons donc vers des approches sans données, en utilisant des techniques d'apprentissage par renforcement. Cela permet d'apprendre une perturbation sur la force de la dynamique de Langevin suramortie en maximisant une certaine fonction de récompense, rendant la transition d'un puits à un autre plus probable. Les résultats sont visuellement plus réalistes qu'avec les méthodes basées sur les données.

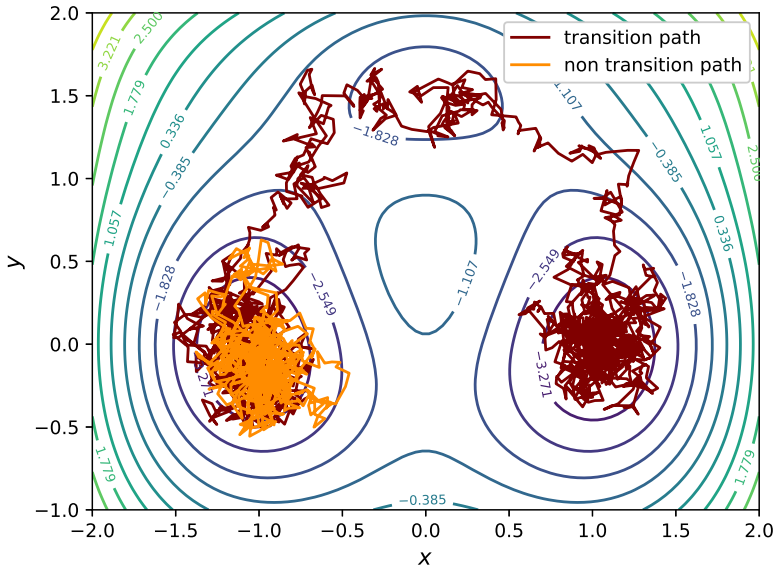


Figure 5.1 – Trajectoire de transition et trajectoire de non-transition dans le potentiel bidimensionnel donné par (4.3).

BIBLIOGRAPHY

[1] A. Abdulle, D. Cohen, G. Vilmart, and K. C. Zygalakis. High weak order methods for stochastic differential equations based on modified equations. *SIAM J. Sci. Comput.*, 34(3):A1800–A1823, 2012.

[2] A. Abdulle, G. Vilmart, and K. C. Zygalakis. High order numerical approximation of the invariant measure of ergodic SDEs. *SIAM J. Numer. Anal.*, 52(4):1600–1622, 2014.

[3] A. Abdulle, G. Vilmart, and K. C. Zygalakis. Long time accuracy of Lie-Trotter splitting methods for Langevin dynamics. *SIAM J. Numer. Anal.*, 53(1):1–16, 2015.

[4] S. Ahn, A. Korattikara, and M. Welling. Bayesian posterior sampling via stochastic gradient Fisher scoring. ICML'12, page 1771–1778, Madison, WI, USA, 2012. Omnipress.

[5] M. P. Allen and D. J. Tildesley. *Computer Simulation of Liquids*. Oxford University Press, Inc., 2nd edition, 2017.

[6] M. Allouche, S. Girard, and E. Gobet. Tail-GAN: Simulation of extreme events with ReLU neural networks. *HAL preprint*, 03250663, 2021.

[7] J. An and S. Cho. Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE*, 2(1):1–18, 2015.

[8] D. Aristoff, T. Lelièvre, C. G. Mayne, and I. Teo. Adaptive multilevel splitting in molecular dynamics simulations. *ESAIM: Proceedings and Surveys*, 48:215–225, 2015.

[9] J. Baker, P. Fearnhead, E. B. Fox, and C. Nemeth. Control variates for stochastic gradient MCMC. *Stat. Comput.*, 29(3):599–615, 2019.

[10] D. Bakry, I. Gentil, and M. Ledoux. *Analysis and geometry of Markov diffusion operators*. Springer, 2014.

[11] R. Bardenet, A. Doucet, and C. Holmes. Towards scaling up Markov chain Monte Carlo: an adaptive subsampling approach. In *International Conference on Machine Learning (ICML)*, Proceedings of the 31st International Conference on Machine Learning (ICML), pages 405–413, Beijing, China, 2014.

[12] D. P. Bertsekas. *Convex optimization algorithms*. Athena Scientific, Belmont, MA, 2015.

[13] A. Beskos, N. Pillai, G. Roberts, J. Sanz-Serna, and A. Stuart. Optimal tuning of the hybrid Monte Carlo algorithm. *Bernoulli*, 19(5A):1501–1534, 2013.

[14] R. N. Bhattacharya. On the functional central limit theorem and the law of the iterated logarithm for Markov processes. *Z. Wahrsch. Verw. Gebiete*, 60(2):185–201, 1982.

[15] J. Bierkens, P. Fearnhead, and G. Roberts. The zig-zag process and super-efficient sampling for Bayesian analysis of big data. *Ann. Statist.*, 47(3):1288–1320, 2019.

[16] C. M. Bishop. *Pattern recognition and machine learning.* Information Science and Statistics. Springer, New York, 2006.

[17] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

[18] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural network. In *International conference on machine learning*, pages 1613–1622. PMLR, 2015.

[19] P. G. Bolhuis, D. Chandler, C. Dellago, and P. L. Geissler. Transition path sampling: Throwing ropes over rough mountain passes, in the dark. *Annual Review of Physical Chemistry*, 53(1):291–318, 2002.

[20] S. Bond-Taylor, A. Leach, Y. Long, and C. G Willcocks. Deep generative modelling: A comparative review of VAEs, GANs, normalizing flows, energy-based and autoregressive models. *arXiv preprint*, 2103.04922, 2021.

[21] D. Bone, M. S. Goodwin, M. P. Black, C. C. Lee, K. Audhkhasi, and S. Narayanan. Applying machine learning to facilitate autism diagnostics: pitfalls and promises. *Journal of autism and developmental disorders*, 45(5):1121–1136, 2015.

[22] N. Bou-Rabee and H. Owhadi. Long-run accuracy of variational integrators in the stochastic context. *SIAM J. Numer. Anal.*, 48(1):278–297, 2010.

[23] S. Brooks, A. Gelman, G. Jones, and X. L. Meng. *Handbook of Markov Chain Monte Carlo.* Chapman & Hall/CRC Handbooks of Modern Statistical Methods. CRC Press, 2011.

[24] N. Brosse, A. Durmus, and E. Moulines. The promises and pitfalls of stochastic gradient Langevin dynamics. *Advances in Neural Information Processing Systems*, 31, 2018.

[25] J. A. Bucklew. *Introduction to Rare Event Simulation.* Springer Series in Statistics. Springer-Verlag, New York, 2004.

[26] E. Cancès, F. Legoll, and G. Stoltz. Theoretical and numerical comparison of some sampling methods for molecular dynamics. *ESAIM: Mathematical Modelling and Numerical Analysis*, 41(2):351–389, 2007.

[27] R. Caruana, S. Lawrence, and C. Giles. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 2000.

[28] F. Cérou, A. Guyader, T. Lelièvre, and D. Pommier. A multiple replica approach to simulate reactive trajectories. *J. Chem. Phys.*, 134(5):054108, 2011.

[29] F. Cérou, A. Guyader, and M. Rousset. Adaptive Multilevel Splitting: Historical perspective and recent results. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 29(4):043108, 2019.

[30] P. Chaudhari and S. Soatto. Stochastic gradient descent performs variational inference converges to limit cycles for deep networks. In *International Conference on Learning Representations*, pages 1–10, 2018.

[31] T. Chen, E. Fox, and C. Guestrin. Stochastic Gradient Hamiltonian Monte Carlo. In E. P. Xing and T. Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1683–1691, Bejing, China, 22–24 Jun 2014. PMLR.

[32] L. Chizat and F. Bach. On the global convergence of gradient descent for over-parameterized models using optimal transport. *Advances in neural information processing systems*, 31, 2018.

[33] J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio. A recurrent latent variable model for sequential data. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[34] F. Cérou and A. Guyader. Adaptive Multilevel Splitting for rare event analysis. *Stochastic Analysis and Applications*, 25(2):417–443, 2007.

[35] A. S. Dalalyan and L. Riou-Durand. On sampling from a log-concave density using kinetic Langevin diffusions. *Bernoulli*, 26(3):1956–1988, 2020.

[36] A. Das, D. C. Rose, J. P. Garrahan, and D. T. Limmer. Reinforcement learning of rare diffusive dynamics. *J. Chem. Phys.*, 155:134105, 2021.

[37] A. Debussche and E. Faou. Weak backward error analysis for SDEs. *SIAM J. Numer. Anal.*, 50(3):1735–1752, 2012.

[38] A. Dieuleveut, A. Durmus, and F. Bach. Bridging the gap between constant step size stochastic gradient descent and Markov chains. *The Annals of Statistics*, 48(3):1348–1382, 2020.

[39] N. Ding, Y. Fang, R. Babbush, C. Chen, R. D. Skeel, and H. Neven. Bayesian sampling using stochastic gradient thermostats. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3203–3211. Curran Associates, Inc., 2014.

[40] N. Dionelis, M. Yaghoobi, and S. A. Tsaftaris. Tail of distribution GAN (tailGAN): Generative Adversarial-Network-based boundary formation. In *2020 Sensor Signal Processing for Defence Conference (SSPD)*, pages 1–5. IEEE, 2020.

[41] J. Dolbeault, C. Mouhot, and C. Schmeiser. Hypocoercivity for kinetic equations with linear relaxation terms. *C. R. Math. Acad. Sci. Paris*, 347(9-10):511–516, 2009.

[42] J. Dolbeault, C. Mouhot, and C. Schmeiser. Hypocoercivity for linear kinetic equations conserving mass. *Trans. Amer. Math. Soc.*, 367(6):3807–3828, 2015.

[43] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid Monte Carlo. *Phys. Lett. B*, 195(2):216–222, 1987.

[44] A. Durmus, A. Enfroy, E. Moulines, and G. Stoltz. Uniform minorization condition and convergence bounds for discretizations of kinetic Langevin dynamics. *arXiv preprint*, 2107.14542, 2021.

[45] A. Durmus, E. Moulines, and E. Saksman. On the convergence of Hamiltonian Monte Carlo. *arXiv preprint*, 1705.00166, 2017.

[46] A. Durmus, U. Simsekli, E. Moulines, R. Badeau, and G. Richard. Stochastic gradient Richardson–Romberg Markov chain Monte Carlo. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[47] W. H. Fleming. Exit probabilities and optimal stochastic control. *Appl. Math. Optim.*, 4(4):329–346, 1977/78.

[48] D. Frenkel and B. Smit. *Understanding Molecular Simulation: From Algorithms to Applications*. Academic Press, 2002.

[49] A. Friedman. *Stochastic differential equations and applications. Volume 1*. Academic Press, 1975.

[50] S. Fujimoto, H. van Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1587–1596. PMLR, 10–15 Jul 2018.

[51] Y. Gal. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.

[52] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, (6):721–741, 1984.

[53] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[54] M. Gürbüzbalaban, X. Gao, Y. Hu, and L. Zhu. Decentralized stochastic gradient Langevin dynamics and Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 22(239):1–69, 2021.

[55] E. Hairer, C. Lubich, and G. Wanner. *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*, volume 31 of *Springer Series in Computational Mathematics*. Springer-Verlag, 2006.

[56] C. Hartmann, R. Banisch, M. Sarich, T. Badowski, and C. Schütte. Characterization of rare events in molecular dynamics. *Entropy*, 16(1):350–376, 2014.

[57] C. Hartmann and C. Schütte. Efficient rare event simulation by optimal nonequilibrium forcing. *Journal of Statistical Mechanics: Theory and Experiment*, 2012(11):P11004, 2012.

[58] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57:97–109, 1970.

[59] F. Heber, Z. Trstanova, and B. Leimkuhler. Tati-thermodynamic analytics toolkit: Tensorflow-based software for posterior sampling in machine learning applications. *arXiv preprint arXiv:1903.08640*, 2019.

[60] M. Hein, M. Andriushchenko, and J. Bitterwolf. Why ReLU networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 41–50, 2019.

[61] F. Hérau. Hypocoercivity and exponential time decay for the linear inhomogeneous relaxation Boltzmann equation. *Asymptot. Anal.*, 46(3-4):349–359, 2006.

[62] G. E. Hinton. Learning translation invariant recognition in a massively parallel networks. In *International Conference on Parallel Architectures and Languages Europe*, pages 1–13. Springer, 1987.

[63] W. G. Hoover. Canonical dynamics - Equilibrium phase-space distributions. *Phys. Rev. A*, 31(3):1695–1697, 1985.

[64] P. Izmailov, S. Vikram, M. D. Hoffman, and A. G. Wilson. What are Bayesian neural network posteriors really like? In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 4629–4640. PMLR, 2021.

[65] A. Jones and B. Leimkuhler. Adaptive stochastic methods for sampling driven molecular systems. *J. Chem. Phys.*, 135(8):084125, 2011.

[66] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

[67] L. V. Jospin, H. Laga, F. Boussaid, W. Buntine, and M. Bennamoun. Hands-on Bayesian neural networks - A tutorial for deep learning users. *IEEE Computational Intelligence Magazine*, 17(2):29–48, 2022.

[68] K. Kawaguchi. Deep learning without poor local minima. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[69] R. Khasminskii. *Stochastic Stability of Differential Equations*, volume 66 of *Stochastic Modelling and Applied Probability*. Springer, Heidelberg, second edition, 2012.

[70] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[71] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[72] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, and et al. Overcoming catastrophic forgetting in neural networks. *Proc. Natl. Acad. Sci. USA*, 114(13):3521–3526, 2017.

[73] W. Kliemann. Recurrence and invariant measures for degenerate diffusions. *Ann. Probab.*, 15(2):690–707, 1987.

[74] P. E. Kloeden and E. Platen. *Numerical Solution of Stochastic Differential Equations*, volume 23 of *Applications of Mathematics (New York)*. Springer-Verlag, Berlin, 1992.

[75] A. Korattikara, Y. Chen, and M. Welling. Austerity in MCMC land: Cutting the Metropolis–Hastings budget. In E. P. Xing and T. Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 181–189, Bejing, China, 22–24 Jun 2014. PMLR.

[76] A. Korattikara, V. Rathod, K. P. Murphy, and M. Welling. Bayesian dark knowledge. *Advances in Neural Information Processing Systems*, 28, 2015.

[77] A. Kristiadi, M. Hein, and P. Hennig. Being Bayesian, even just a bit, fixes overconfidence in ReLU networks. In *International conference on machine learning*, pages 5436–5446. PMLR, 2020.

[78] A. Krogh and J. Hertz. A simple weight decay can improve generalization. *Advances in Neural Information Processing Systems*, 4, 1991.

[79] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

[80] N. Laptev. AnoGen: Deep anomaly generator, 2018. https://research.facebook.com/file/969101687155819/AnoGen-Deep-Anomaly-Generator.pdf.

[81] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[82] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[83] F. Legoll, M. Luskin, and R. Moeckel. Non-ergodicity of the Nosé-Hoover thermostatted harmonic oscillator. *Arch. Ration. Mech. Anal.*, 184(3):449–463, 2007.

[84] E. L. Lehmann and G. Casella. *Theory of Point Estimation*. Springer Texts in Statistics. Springer-Verlag, New York, second edition, 1998.

[85] B. Leimkuhler and C. Matthews. Rational construction of stochastic numerical methods for molecular sampling. *Appl. Math. Res. Express*, pages 34–56, 2013.

[86] B. Leimkuhler and C. Matthews. *Molecular Dynamics: With Deterministic and Stochastic Numerical Methods*. Interdisciplinary Applied Mathematics. Springer, 2015.

[87] B. Leimkuhler, C. Matthews, and G. Stoltz. The computation of averages from equilibrium and nonequilibrium Langevin molecular dynamics. *IMA J. Numer. Anal.*, 36(1):13–79, 2016.

[88] B. Leimkuhler, C. Matthews, and T. Vlaar. Partitioned integrators for thermodynamic parameterization of neural networks. *Foundations of Data Science*, 1(4):457–489, 2019.

[89] B. Leimkuhler and S. Reich. *Simulating Hamiltonian Dynamics*. Cambridge University Press, 2004.

[90] B. Leimkuhler, M. Sachs, and G. Stoltz. Hypocoercivity properties of adaptive Langevin dynamics. *SIAM J. Appl. Math.*, 80(3):1197–1222, 2020.

[91] B. Leimkuhler and X. Shang. Adaptive thermostats for noisy gradient systems. *SIAM J. Sci. Comput.*, 38(2):A712–A736, 2016.

[92] T. Lelièvre, G. Robin, I. Sekkat, G. Stoltz, and G. Victorino Cardoso. Generative methods for sampling transition paths in molecular dynamics. *arXiv preprint arXiv:2205.02818*, 2022.

[93] T. Lelièvre, M. Rousset, and G. Stoltz. *Free Energy Computations: A Mathematical Perspective*. Imperial College Press, 2010.

[94] T. Lelièvre and G. Stoltz. Partial differential equations and stochastic methods in molecular dynamics. *Acta Numerica*, 25:681–880, 2016.

[95] C. Li, C. Chen, D. Carlson, and L. Carin. Preconditioned stochastic gradient Langevin dynamics for deep neural networks. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[96] S. Livingstone, M. Betancourt, S. Byrne, and M. Girolami. On the geometric ergodicity of Hamiltonian Monte Carlo. *Bernoulli*, 25(4A):3109–3138, 2019.

[97] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *International Conference on Learning Representations*, 2019.

[98] Y-A. Ma, T. Chen, and E. Fox. A complete recipe for Stochastic Gradient MCMC. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[99] Y-A. Ma, Y. Chen, C. Jin, N. Flammarion, and M. I. Jordan. Sampling can be faster than optimization. *Proceedings of the National Academy of Sciences*, 116(42):20881–20885, 2019.

[100] C. Matthews and J. Weare. Langevin Markov Chain Monte Carlo with stochastic gradients. *arXiv preprint*, 1805.08863, 2018.

[101] J. C. Mattingly, A. M. Stuart, and D. J. Higham. Ergodicity for SDEs and approximations: locally Lipschitz vector fields and degenerate noise. *Stochastic Process. Appl.*, 101(2):185–232, 2002.

[102] J. C. Mattingly, A. M. Stuart, and M. V. Tretyakov. Convergence of numerical time-averaging and stationary measures via Poisson equations. *SIAM Journal on Numerical Analysis*, 48(2):552–577, 2010.

[103] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equations of state calculations by fast computing machines. *J. Chem. Phys.*, 21(6):1087–1091, 1953.

[104] N. Metropolis, A.W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *J. Chem. Phys.*, 21(6):1087–1092, 1953.

[105] P. Metzner, C. Schütte, and E. Vanden-Eijnden. Transition path theory for Markov jump processes. *Multiscale Modeling & Simulation*, 7(3):1192–1219, 2009.

[106] S. Meyn and R. Tweedie. *Markov Chains and Stochastic Stability*. Springer-Verlag, 1993.

[107] G. N. Milstein and M. V. Tretyakov. *Stochastic Numerics for Mathematical Physics*. Scientific Computation. Springer Berlin Heidelberg, 2013.

[108] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing Atari with deep reinforcement learning. *arXiv preprint*, 1312.5602, 2013.

[109] W. Mou, Y.-A. Ma, M. J. Wainwright, P. L. Bartlett, and M. I. Jordan. High-order Langevin diffusion yields an accelerated MCMC algorithm. *Journal of Machine Learning Research*, 22(42):1–41, 2021.

[110] T. Nagapetyan, A. Duncan, L. Hasenclever, S. Vollmer, L. Szpruch, and K. Zygalakis. The true cost of Stochastic Gradient Langevin Dynamics. *arXiv preprint*, 1706.02692, 2017.

[111] R. M. Neal. *Bayesian Learning for Neural Networks*, volume 118. Springer Science & Business Media, 2012.

[112] S. Nosé. A unified formulation of the constant temperature molecular-dynamics methods. *J. Chem. Phys.*, 81(1):511–519, 1984.

[113] Roberts G. O. and Tweedie R. L. Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, 2(4):341 – 363, 1996.

[114] B. Oksendal. *Stochastic Differential Equations: an Introduction with Applications*. Springer Science & Business Media, 2013.

[115] J. Paisley, D. Blei, and M. Jordan. Variational Bayesian inference with stochastic search. *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*, 2, 06 2012.

[116] A. Pakman, D. Gilboa, D. Carlson, and L. Paninski. Stochastic bouncy particle sampler. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2741–2750. PMLR, 2017.

[117] S. Park, M.K. Sener, D. Lu, and K. Schulten. Reaction paths based on mean first-passage times. *J. Chem. Phys.*, 119(3):1313–1319, 2003.

[118] G. A. Pavliotis. *Stochastic Processes and Applications: Diffusion Processes, the Fokker-Planck and Langevin Equations*. Texts in Applied Mathematics. Springer New York, 2014.

[119] P. H. Peskun. Optimum Monte-Carlo sampling using Markov Chains. *Biometrika*, 60(3):607–612, 1973.

[120] S. Pesme, L. Pillaud-Vivien, and N. Flammarion. Implicit bias of SGD for diagonal linear networks: a provable benefit of stochasticity. *Advances in Neural Information Processing Systems*, 34, 2021.

[121] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2014.

[122] M. Quiroz, R. Kohn, M. Villani, and M. Tran. Speeding up MCMC by efficient data subsampling. *J. Amer. Statist. Assoc.*, 114(526):831–843, 2019.

[123] L. Rey-Bellet. Ergodic properties of Markov processes. In *Open quantum systems. II*, volume 1881 of *Lecture Notes in Math.*, pages 1–39. Springer, Berlin, 2006.

[124] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In E.P. Xing and T. Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1278–1286, Bejing, China, 22–24 Jun 2014. PMLR.

[125] H. Ritter, A. Botev, and D. Barber. A scalable laplace approximation for neural networks. In *6th International Conference on Learning Representations, ICLR 2018-Conference Track Proceedings*, volume 6. International Conference on Representation Learning, 2018.

[126] H. Robbins and S. Monro. A stochastic approximation method. *Ann. Math. Statistics*, 22:400–407, 1951.

[127] C. P. Robert. *The Bayesian choice*. Springer Texts in Statistics. Springer, New York, second edition, 2007. From decision-theoretic foundations to computational implementation.

[128] C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer Texts in Statistics. Springer-Verlag, New York, second edition, 2004.

[129] G. O. Roberts and J. S. Rosenthal. Optimal scaling of discrete approximations to Langevin diffusions. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 60(1):255–268, 1998.

[130] G. O. Roberts and J. S. Rosenthal. Optimal scaling for various Metropolis-Hastings algorithms. *Statist. Sci.*, 16(4):351–367, 2001.

[131] G. O. Roberts and R. L. Tweedie. Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, 2(4):341–363, 1996.

[132] P. J. Rossky, J. D. Doll, and H. L. Friedman. Brownian dynamics as smart Monte Carlo simulation. *J. Chem. Phys.*, 69(10):4628–4633, 1978.

[133] G. M. Rotskoff and E. Vanden-Eijnden. Learning with rare data: using active importance sampling to optimize objectives dominated by rare events. *arXiv preprint*, 2008.06334, 2020.

[134] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[135] T. Salimans, D. Kingma, and M. Welling. Markov Chain Monte Carlo and variational inference: Bridging the gap. In *International Conference on Machine Learning*, pages 1218–1226. PMLR, 2015.

[136] C. Schütte, S. Winkelmann, and C. Hartmann. Optimal control of molecular dynamics using Markov state models. *Mathematical Programming*, 134(1):259–282, 2012.

[137] I. Sekkat and G. Stoltz. Removing the mini-batching error in Bayesian inference using Adaptive Langevin dynamics. *arXiv preprint arXiv:2105.10347*, 2021.

[138] X. Shang, Z. Zhu, B. Leimkuhler, and A. J. Storkey. Covariance-controlled adaptive Langevin thermostat for large-scale Bayesian sampling. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28, pages 37–45. Curran Associates, Inc., 2015.

[139] T. Shardlow. Modified equations for stochastic differential equations. *BIT Numerical Mathematics*, 46(1):111–125, 2006.

[140] D. Silver, T. Hubert, J. Schrittwieser, and et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018.

[141] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[142] G. Stoltz. Path sampling with stochastic dynamics: some new algorithms. *J. Comput. Phys.*, 225:491–508, 2007.

[143] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT press, 2018.

[144] D. Talay. Second-order discretization schemes of stochastic differential systems for the computation of the invariant law. *Stochastics and Stochastic Reports*, 29(1):13–36, 1990.

[145] D. Talay. Stochastic Hamiltonian dissipative systems: Exponential convergence to the invariant measure, and discretization by the implicit Euler scheme. *Markov Proc. Rel. Fields*, 8:163–198, 2002.

[146] D. Talay and L. Tubaro. Expansion of the global error for numerical schemes solving stochastic differential equations. *Stochastic Anal. Appl.*, 8(4):483–509, 1990.

[147] Y. W. Teh, A. H. Thiery, and S. J. Vollmer. Consistency and fluctuations for stochastic gradient Langevin dynamics. *Journal of Machine Learning Research*, 17:Paper No. 7, 33, 2016.

[148] M. Tuckerman. *Statistical Mechanics: Theory and Molecular Simulation*. Oxford University Press, 2010.

[149] A. van den Oord, O. Vinyals, and K. Kavukcuoglu. Neural discrete representation learning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[150] A. W. van der Vaart. *Asymptotic Statistics*, volume 3 of *Cambridge Series in Statistical and Probabilistic Mathematics*. Cambridge University Press, Cambridge, 1998.

[151] Y. B. Varolgüneş, T. Bereau, and J. F. Rudzinski. Interpretable embeddings from molecular simulations using Gaussian mixture variational autoencoders. *Machine Learning: Science and Technology*, 1(1):015012, 2020.

[152] L. Verlet. Computer "experiments" on classical fluids. i. thermodynamical properties of Lennard-Jones molecules. *Physical review*, 159(1):98, 1967.

[153] T. Vlaar and B. Leimkuhler. Multirate Training of Neural Networks. *arXiv preprint arXiv:2106.10771*, 2021.

[154] S. J. Vollmer, K. C. Zygalakis, and Y. W. Teh. Exploration of the (non-)asymptotic bias and variance of stochastic gradient Langevin dynamics. *Journal of Machine Learning Research*, 17:Paper No. 159, 2016.

[155] H. Wang and D.-Y. Yeung. A survey on Bayesian deep learning. *ACM Comput. Surv.*, 53(5), 2020.

[156] C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3):279–292, 1992.

[157] M. Welling and Y. W. Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 681–688, USA, 2011. Omnipress.

[158] Y. Wen, P. Vicol, J. Ba, D. Tran, and R. Grosse. Flipout: Efficient pseudo-independent weight perturbations on mini-batches. *arXiv preprint arXiv:1803.04386*, 2018.

[159] P. J. Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural networks*, 1(4):339–356, 1988.

[160] J. Yan, H. Touchette, and G. M. Rotskoff. Learning nonequilibrium control forces to characterize dynamical phase transitions. *Phys. Rev. E*, 105:024115, 2022.

[161] W. Zeng, S. Cao, X. Huang, and Y. Yao. A note on learning rare events in molecular dynamics using LSTM and transformer. *arXiv preprint*, 2107.06573, 2021.

[162] Z. Zhu, J. Wu, B. Yu, L. Wu, and J. Ma. The anisotropic noise in stochastic gradient descent: Its behavior of escaping from sharp minima and regularization effects. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7654–7663. PMLR, 2019.

[163] K. C. Zygalakis. On the existence and the applications of modified equations for stochastic differential equations. *SIAM J. Sci. Comput.*, 33(1):102–130, 2011.