# Numerical comparison of MCMC methods for Quantum Tomography

Author: **Danila Mokeev**
Supervisors: **Estelle Massart, Andrew Thompson**
Readers: **Tameem Adel, Pierre-Antoine Absil**
Academic year 2023–2024
Master [120] in Mathematical Engineering

# Acknowledgements

I would like to thank Estelle Massart, Andrew Thompson, and Tameem Adel for their help in the making of this thesis. Their guidance and great ideas were valuable in its success. I also want to thank Matthieu Génévriez for his help in making the Quantum Tomography section more rigorous, and researchers Pierre Alquier and The Tien Mai for providing the source code for the prob-estimator. Finally, I want to thank my family who supported me throughout the university years, and my friends, who made them more enjoyable.

# Contents

# Abbreviations and symbols

| | |
|---|---|
| MCMC | Markov chain Monte Carlo |
| POVM | Positive Operator-Valued Measure |
| MHS | Metropolis-Hastings with Student-t prior algorithm |
| MHGS | Metropolis-Hastings with Gibbs with Student-t prior algorithm |
| $\|\psi\rangle$ | Quantum state $\psi$ in the *bra-ket* formulation |
| $M^\dagger$ | Conjugate transpose of matrix $M$ |
| $\langle M \rangle$ | Expected value of a matrix/operator $M$ |
| $M \otimes N$ | Kronecker product between matrices $M$ and $N$ |
| $M \odot N$ | Hadamard (element-wise) product between matrices $M$ and $N$ |
| $U(a,b)$ | Uniform distribution defined on the open interval $(a,b)$ |
| $N(\mu, \sigma^2)$ | Normal distribution with mean $\mu$ and variance $\sigma^2$ |
| $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ | Multivariate normal distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ |
| $N(\mu, \sigma^2)^{d \times r}$ | Matrix of size $d \times r$ where each entry is independently drawn from a univariate normal distribution with mean $\mu$ and variance $\sigma^2$ |
| $Gamma(\alpha, \beta)$ | Gamma distribution with shape $\alpha$ and rate $\beta$ |
| $\mathcal{D}ir(\alpha_1, \ldots, \alpha_d)$ | Dirichlet distribution with concentration parameters $\alpha_1, \ldots, \alpha_d > 0$ |

# Chapter 1

# Introduction

## 1.1 Motivation

A fundamental problem in physics is the reconstruction of the state of a system given measurements describing it. In classical mechanics, this is, at least in theory, always possible due to the deterministic nature of the system. This allows one to make multiple measurements on a system, usually the position and momentum, to exactly determine its state. In Quantum Mechanics, however, the situation is very different: measuring the system disturbs it (in Quantum Mechanics we talk about wavefunction collapse), making it impossible to gain more information by repeating measurements. We are also fundamentally limited by what we can measure: the Heisenberg uncertainty principle only allows the measurement of either the position or the momentum, but not both at the same time. Finally, the no-cloning theorem also forbids the system to be copied right before the measurement unless we know the state of the system.

*Quantum Tomography* or *Quantum State Tomography* is a process that allows to reconstruct the state of a system in the context of Quantum Mechanics. It tackles the previously described issues by replicating the initial state of the system multiple times, and then measuring once each replica. Note that there is a subtle difference between copying the system and preparing multiple systems with the same steps, as the latter does *not* require any knowledge about the state, only the *steps* necessary to produce it.

The term "tomography" means "to describe an object based on sections, slices" [Wik24i]. Many types of tomographies exist and are used in very diverse environments, with notable examples in medical applications, such as Computed Tomography (CT) or Magnetic Resonance Imaging (MRI). Quantum Tomography works in a similar way: we use a set of slices (observables), each of them probing a particular aspect of that state, to reconstruct the quantum state through a tomographic process. An intuitive analogy is the process of shining light on each side of a 3D object, and reconstructing that object from the obtained shadows. This results in an inverse problem, whereby starting from partial and noisy data (and in this case fundamentally probabilistic), we can obtain the true state of the system.

## 1.2 Quantum Tomography

As previously described, the goal of Quantum Tomography is the reconstruction of the system state based on measurements. All of this system information is stored in a matrix, called the *density matrix* $\rho \in \mathbb{C}^{d\times d}$, with $d = 2^n$ and $n$ the number of qubits. Due to the physical properties of the system, this matrix is Hermitian ($\rho = \rho^\dagger$), positive semi-definite ($\rho > 0$) and satisfying $\operatorname{tr}(\rho) = 1$. The rank of the matrix also has importance: pure states are of rank 1, while mixed states are of any rank between 2 and $d$. A pure state is described by the wavefunction, which is usually represented by a vector (or equivalently a rank-1 matrix), of potentially infinite dimension. Mixed states, on the other hand, are a generalization of pure states and are represented by a density matrix. They correspond to a probabilistic mixture of pure states. It is important to note that physicists are mostly interested in density matrices of low rank.

Quantum Tomography is based on Born's rule, which links a projective measurement $P_m$ associated to an *observable* $O$ to the probability $p(m)$ of obtaining the associated eigenvalue $m$ given a density matrix $\rho$. This can generally be written as

$$p(m) = \operatorname{tr}(\rho P_m) \tag{1.1}$$

By repeating the measurement on many different but identically prepared systems for each observable, we are able to obtain statistically significant estimates for $p(m)$, which combined with the projectors $P_m$, allows us to estimate the density matrix with any appropriate method.

## 1.3 Overview of existing methods

We first provide in 1.1 a summary of existing methods with the papers using them, followed by a more detailed explanation below.

| Method | Papers |
|---|---|
| Inversion | [VR89; RMH10; Alq+13] |
| Pauli basis expansion | [Cai+16] |
| Maximum Likelihood | [Guţ+20; Ban+99; Jam+01; Lvo04; Blu10a; SYH20; Hra+04] |
| Compressed Sensing | [Gro+10; Gro11; Fla+12; Kol11] |
| Sampling-based | [Fer14; KF15; GCC16; Kra+13] |
| Metropolis-Hastings based | [MA17; Mai22; Luk+20; Cot+13; Blu10b] |
| Langevin | [Ade+24] |

Table 1.1: Overview of existing methods with associated papers

Well-established methods exist to approximate $\rho$. We can split the existing methods into 5 main categories: inversion, Pauli basis expansion, Maximum Likelihood, Compressed sensing, and finally Bayesian methods, which are mostly sampling-based algorithms.

Inversion methods are very simple, as they directly rely on the Born rule: $\hat{\rho}$ is computed by solving the linear system of equations defined by

$$\forall \mathbf{a} \in \mathcal{A}, \forall \mathbf{s} \in \mathcal{S} : \hat{p}_{\mathbf{a},\mathbf{s}} = \mathrm{tr}(\hat{\rho} P_{\mathbf{s}}^{\mathbf{a}})$$

where $\mathcal{A}$ denotes the set of indices for observables, $\mathcal{S}$ the set of indices for projective measurements, $\hat{p}_{\mathbf{a},\mathbf{s}}$ the empirical probability and $P_{\mathbf{s}}^{\mathbf{a}}$ the projection operator. This approach was introduced in [VR89], and then used in [RMH10], but also in [Alq+13], where they additionally use a rank penalization term. This method is easy to understand, however, it returns a matrix $\hat{\rho}$ which does not satisfy the properties of a density matrix.

A second approach, introduced in [Cai+16], works by estimating the coefficients in the Pauli basis expansion. Indeed, as the observables of interest we use in Quantum Tomography are the Kronecker product of Pauli matrices (which are usually called $\{\sigma_x, \sigma_y, \sigma_z\} \in \mathbb{C}^{2\times 2}$), we can also approximate $\rho$ using the resulting observable as an element of a base. If we have $\mathcal{B} = \{\sigma_b = \sigma_{b_1} \otimes \cdots \otimes \sigma_{b_n}, b \in \{I, x, y, z\}^n\}$, then the density matrix $\rho$ can be approximated as

$$\rho = \sum_{b \in \{I,x,y,z\}^n} \rho_b \sigma_b \tag{1.2}$$

In the case of [Cai+16], the coefficients $\rho_b$ are estimated using an average of the measured eigenvalues with each observable.

Another common approach used by many [Guţ+20; Ban+99; Jam+01; Lvo04; Blu10a; SYH20; Hra+04] is Maximum Likelihood (ML) estimation, which tries to maximize an objective function $\mathcal{L}$ to find the best $\rho$ which matches the data. A possible loss function could be

$$\mathcal{L}(\rho; \mathbf{D}) \propto \prod_{\mathbf{a}\in\mathcal{A}} \prod_{\mathbf{s}\in\mathcal{S}} \left[\mathrm{tr}\left(\rho P_{\mathbf{s}}^{\mathbf{a}}\right)\right]^{n_{\mathbf{a},\mathbf{s}}}, \tag{1.3}$$

where $n_{\mathbf{a},\mathbf{s}}$ is the number of occurences of $\mathbf{s}$ in measurements by $\mathbf{a}$. The density matrix is calculated as

$$\hat{\rho}_{\mathrm{ML}} = \mathrm{argmin}_{\rho}\, \mathcal{L}(\rho; \mathbf{D}) \tag{1.4}$$

A downside however often mentioned for this method is the cost, notably for $n \geq 10$.

There is also a certain number of papers [Gro+10; Gro11; Fla+12; Kol11] which treat the problem of solving 1.3 as a pure optimization problem, with the extra contraint that not all observables are available (compressed sensing). They are however less relevant to us, as we (mostly) consider ourselves to be in the complete measurement case.

The methods described so far do not make any assumptions about the structure of $\rho$. They may enforce the properties of a density matrix, but not use the low-rank information, with the only exception being papers that include a rank regularization term in the loss function (such as [Alq+13]). To tackle this issue, a good idea is to venture into Bayesian methods, which naturally define a *prior* on $\rho$. They also have the advantage of providing a distribution over it, rather than a point estimate, as most optimization-based methods do. This gives us uncertainty quantification.

Bayesian methods rely on *Bayes* theorem, which states that given data $\mathbf{D}$, which in our case are the empirical probabilities $p_{\mathbf{a},\mathbf{s}}$ and projectors $P^{\mathbf{a}}_{\mathbf{s}}$, and parameters $\theta$, in our case the density matrix $\rho$, we can calculate the posterior distribution $\pi(\rho|\mathbf{D})$:

$$\pi(\rho|\mathbf{D}) \propto \pi(\mathbf{D}|\rho)\pi(\rho) \tag{1.5}$$

where $\pi(\mathbf{D}|\rho)$ is the likelihood of $\mathbf{D}$ given $\rho$ and $\pi(\rho)$ the prior distribution over it. There is also a normalizing term, however, it is normally omitted as it does not modify the result. Obtaining samples from this posterior is what allows the approximation, and the algorithm used for sampling is the key part of the various approaches. The most common estimator then used is the sample mean, also sometimes called the *Bayesian Mean Estimator* (BME) or the Gibbs estimator:

$$\hat{\rho}_{\text{BME}} = \int \hat{\rho}\pi(\hat{\rho}|\mathbf{D})d\hat{\rho} \tag{1.6}$$

Markov Chain Monte Carlo (MCMC) methods are a big class of sampling methods and are the common tool of choice. In particular, [MA17; Mai22; Cot+13; Luk+20; Blu10b] all propose algorithms derived from the very classical Metropolis-Hastings, an algorithm that performs a random walk from an initial point, with an acceptance/rejection step. Some of them, in particular [Mai22; Cot+13; Luk+20], use an approach called "preconditioned Crank-Nicholson", which allows for higher acceptance rates and consequently faster convergence. There are also other strategies, such as Sequential Monte Carlo [Fer14; KF15; GCC16] or Sequential Importance Sampling [Kra+13]. Finally, there is also a new method introduced in [Ade+24], where the authors use the gradient information from the posterior, resulting in Langevin sampling. This technique also provides faster convergence times.

The work in the thesis will focus on the prob-estimator introduced in [MA17], as well as the projected Langevin approach introduced in [Ade+24]. They differ in the algorithms they use, Metropolis-Hastings and Langevin sampling, but also in the prior they choose. A complete description of both of these methods is available in chapter 3.

## 1.4 Goals and contributions

In this thesis, our contribution is twofold: first, we investigate how the prob-estimator introduced in [MA17] compares to the projected Langevin algorithm from [Ade+24]; second, we try to understand how much impact the used prior has in comparison to the MCMC algorithm that we use to perform the sampling. For the latter, we introduce 2 new algorithms, Metropolis-Hastings with Student-t prior (MHS) and Metropolis-Hastings with Gibbs with Student-t prior (MHGS), which mix the algorithm from [MA17] with the prior from [Ade+24]. This allows us to evaluate the advantages that a gradient-based method brings, as well as the effect of a Student-t prior on the result. We will focus on providing numerical results, with all the code to reproduce the experiments available in the near future on Github[1].

[1] www.github.com/daqwes/thesis

## 1.5 Structure

This thesis will be structured as follows.

In Chapter 2, we will provide the needed background to understand Quantum Tomography in the context of numerical experiments, as well as an introduction to Markov chain Monte Carlo methods, with an emphasis on the methods relevant to this thesis.

In Chapter 3, we will review in detail the 2 main algorithms used in the numerical experiments in the context of this work: the prob-estimator and the Projected Langevin algorithm.

In Chapter 4, we will numerically compare the accuracy of both algorithms across various experimental setups.

Finally, in Chapter 5, we will introduce 2 new algorithms, MHS and MHGS, to better understand how the choice of the algorithm and prior impacts the accuracy. This will allow us to shine a light on the potential benefits of the gradient information and the Student-t prior, brought by the Projected Langevin method.

# Chapter 2

# Background

In this chapter, we will introduce the relevant background needed for this thesis. We will start with Quantum Information, the qubit and Quantum Tomography in sections 2.1, 2.2, 2.3, and will follow with Markov chain Monte Carlo methods in section 2.4.

## 2.1 Quantum information

Quantum information is a field that explores how information can be encoded and processed using the principles of quantum mechanics. While classical computers perform operations using *bits* (hence either 0 or 1), quantum computers operate on *qubits*, which allows for a superposition of 0 and 1. This means that a *quantum state* can be in an infinite combination between 0 and 1. This property is more clearly explained using probabilities, where the likelihood of measuring the qubit in a certain state will be proportional to the norm squared of the coefficients of the superposition. It is important to note that when we perform the measurement, only one of the states, 0 or 1, will be visible.

This measurement process is very important in quantum mechanics, as contrary to classical mechanics, we can not simply observe the system. There are 4 main properties that limit us: fundamentally, a quantum system is probabilistic - measuring it once does not give complete information about the state; in addition, a measurement makes the system wavefunction collapse - repeating the measurement only yields the same state again; the no-cloning theorem, which does not allow to simply copy the state; finally, the Heisenberg uncertainty principle, which states that either component of certain pairs of conjugate operators (such as the position and momentum, for example) can be measured, but not both. Given that, we have to find ways to measure the quantum state of the system, and this is where Quantum Tomography fits into the picture.

**Bra-ket notation** The *bra-ket* notation is going to be used in the following sections, so it is important to first introduce its meaning. It originates from quantum mechanics, where we often manipulate complex vectors (finite or infinite). The ket notation $|a\rangle$ corresponds to a column vector $a$. The bra notation $\langle a|$ corresponds to the conjugate transpose of $|a\rangle$. This means that $\langle a|b\rangle$ is an inner product between $|a\rangle$ and $|b\rangle$, and $|a\rangle\langle b|$ an outer product. The notation $\langle a|M|b\rangle$ corresponds to a quadratic form between vectors $a$ and $b$, and matrix $M$.

## 2.2 The qubit

The *qubit* is the fundamental unit of computation in quantum information. It contrasts from classical bits by allowing a continuum of possible states: a linear combination between 0 and 1. We usually use the orthonormal basis states with the *bra-ket* formulation to represent it: $|0\rangle = \begin{bmatrix}1\\0\end{bmatrix}$ and $|1\rangle = \begin{bmatrix}0\\1\end{bmatrix}$. The resulting state $\psi$ is written as follows:

$$\psi = \alpha|0\rangle + \beta|1\rangle \tag{2.1}$$

where $\alpha, \beta \in \mathbb{C}$ are probability amplitudes, with the constraint that $|\alpha|^2 + |\beta|^2 = 1$. This last equality is needed to satisfy the second axiom of probability, as after the measurement either of outcomes must be observed.

An intuitive way to visualize a qubit is to use a *Bloch sphere*. First, we must do a change of coordinates, by going from cartesian to Hopf coordinates. With this in mind, it can be shown that $|\psi\rangle$ can be rewritten as

$$|\psi\rangle = e^{i\gamma}\left(\cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle\right) \tag{2.2}$$

with $\theta, \gamma, \phi \in \mathbb{R}$. The term $e^{i\gamma}$ has no observable effects, allowing us to remove it[1]. This leaves us with 2 parameters $\theta$ and $\phi$, which finally allows us to represent the qubit on a sphere:

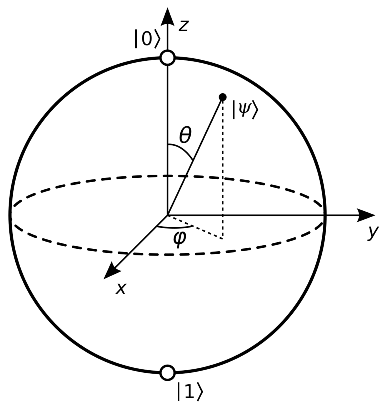


Figure 2.1: Bloch sphere representation of a qubit [Wik24h]

While the classical bit only exists in 2 points, the north and south poles, the qubit extends the quantum state to any position on the surface of the cube. [Wik24h; NC10]

[1]This is because the projective measurement calculates the probability amplitude *squared*

The visual explanation unfortunately fades when we consider multiple qubits: in that case, it is much easier to talk about a $2^n$ dimensional basis. For example, with 2 qubits, 4 basis elements describe a quantum state:

$$|\phi\rangle = \alpha\,|00\rangle + \beta\,|01\rangle + \gamma\,|10\rangle + \delta\,|11\rangle \tag{2.3}$$

where $\alpha, \beta, \gamma, \delta \in \mathbb{C}$ and normalize to 1, and $|\phi\rangle \in \mathbb{C}^4$.

## 2.3 Quantum Tomography

As described in 1.2, Quantum Tomography studies the reconstruction of a quantum state described by a density matrix $\rho$. It builds upon the Born rule, which states that given a system state $|\psi\rangle$ and an observable operator $O$, the probability of occurrence of an eigenvalue $m$ of $O$ will be

$$p(m) = \langle\psi|P_m|\psi\rangle = |\langle m|\psi\rangle|^2 \tag{2.4}$$

where $P_m = |m\rangle\langle m|$ is a projection onto the eigenstate of $O$ with eigenvalue $m$ (and corresponding eigenvector $|m\rangle$). The system is measured using $P_m$ with the outcome being an eigenstate $m$ (to be precise it is measured with the collection $\{P_m\}$ associated to $O$). Note that the term operator will be used interchangeably with the term matrix in what follows, as they are equivalent in this context.

In general, one observable is however not enough as it only measures a specific aspect of a state. We will therefore create a collection of observables $\mathcal{O}$ with $O \in \mathcal{O}$, which will allow us to fully describe our state. In the situation when we have access to the entire set $\mathcal{O}$, we talk about a *complete measurement* case. Of course, doing one measurement with an observable is not informative enough, and in order to obtain any statistically significant estimate for the probability of projector, we measure an ensemble of identically prepared quantum states for each $O$.

In the following sections, we will provide a more thorough mathematical background and describe in more detail the different components that constitute Quantum Tomography.

### 2.3.1 Mathematical description

#### Pure states, mixed states and density matrix

A *pure state* is a quantum state that can be represented using a finite or infinite complex vector $|\psi\rangle$, element of a Hilbert space, and of norm 1. Equivalently, it is a state that can not be expressed as a convex combination of other quantum states [Wik24b].

A *mixed state* $|\phi\rangle$, on the other hand, is a probabilistic mixture of pure states $|\psi_j\rangle$. Mixed states usually arise in situations when we do not know from which states our system state is constituted. This means that we can not anymore represent our system state as a vector, and must resort to a more general form: the density matrix.

A *density matrix* $\rho \in \mathbb{C}^{d\times d}$, with $d = 2^n$ and $n$ the number of qubits, is a matrix that represents a general quantum state. It is written as

$$\rho = \sum_j p_j \left|\psi_j\right\rangle \left\langle\psi_j\right| \tag{2.5}$$

where $p_j$ is the probability of $|\psi_j\rangle$.

By construction, we can see that the properties of a density matrix are that it is Hermitian ($\rho = \rho^\dagger$), positive semi-definite ($\rho > 0$), and satisfying $\mathrm{tr}(\rho) = 1$. The rank of this matrix also allows us to easily distinguish between a pure state of rank 1, and a mixed state of any rank between 2 and $2^n$, where $n$ is the number of qubits.

### Born rule, observable and POVM

The Born rule is a fundamental postulate in Quantum Mechanics that connects the wave function (a possible way to represent the system state) to its measurement. It says that "the probability density of finding a system in a given state when measured, is proportional to the square of the amplitude of the system's wavefunction at that state" [Wik24a].

It is described by a collection of orthogonal projective measurements $\{P_m\}$ with $P_m P_{m'} = \delta_{mm'} P_m$. This collection relates to an *observable*, represented by a Hermitian matrix $O$. This property allows it to have a spectral decomposition

$$O = \sum_m m P_m \tag{2.6}$$

where $P_m$ is the projection onto the eigenspace of $O$ associated to eigenvalue $m$. An observable corresponds to a physical property that can be measured, for example, the position, momentum, or spin of a particle.

In the case where our state $|\psi\rangle$ is pure, the Born rule is formulated as

$$p(m) = \langle\psi|P_m|\psi\rangle \tag{2.7}$$

and can be further generalized to a mixed state, represented with a density matrix $\rho$

$$p(m) = \mathrm{tr}(\rho P_m) \tag{2.8}$$

A small example of such a collection are the projectors operators of the *computational basis* ($|0\rangle$ and $|1\rangle$). The resulting matrices are $P_0 = |0\rangle\langle 0|$ and $P_1 = |1\rangle\langle 1|$. If we have that $|\psi\rangle = a|0\rangle + b|1\rangle$, then $p(0) = \langle\psi|P_0^\dagger P_0|\psi\rangle = \langle\psi|P_0|\psi\rangle = |a|^2$ (by idempotency of a projector matrix).

In the context of Quantum Tomography, the most relevant aspect to measure is the spin: in each direction $\{x, y, z\}$, we use a Pauli matrix to probe whether a qubit is spin up or down [NC10; Wik24a].

Note that, while projective measurements are the most common measurement operator, they are a specific case of the *Positive Operator-Valued Measure* (POVM). This generalized measure does not enforce the orthogonality property and only requires the collection $\{F_m\}$ to be Hermitian, positive semi-definite operators on a Hilbert space, and satisfy the completeness relation

$$\sum_m F_m = I \tag{2.9}$$

### 2.3.2 Data generation

There are multiple ways through which one can obtain data for quantum tomography: using a real-world dataset is a possibility, however for that, you need access (or need to know someone who has) to a quantum computer. An easier approach is simply to simulate the data. It has its downsides, notably the fact that it may not be representative of true data. In our experiments, however, we will stick to synthetic data as it is very flexible and better suited for algorithm research and numerical experiments, due to the low iteration time.

When using synthetic data generation, 2 main options are usually considered: separate qubit or mixed qubit. Note that these names are of our own invention, as there does not seem to be a common consensus in the litterature on how to call them.

#### Separate qubit data generation

Separate qubit data generation (also called the “Pauli basis measurements” in [Guţ+20]) is a process through which we can find the probability associated with an outcome $\mathbf{s} = (s_1, s_2 \ldots, s_n) \in \mathcal{R}^n := \{-1, 1\}^n$ for an observable $\mathbf{a} = (a_1, a_2 \ldots, a_n) \in \mathcal{E}^n := \{x, y, z\}^n$ (we adopt the same notations as in [MA17] here and make them more specific than in the introduction 1.3).

The observable $\sigma_{a_i}$ for a qubit can be in $\{\sigma_x, \sigma_y, \sigma_z\}$, where $\sigma_i \in \mathbb{C}^{2\times 2}$ is one of the Pauli matrices. These matrices are written as follows:

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{2.10}$$

They distinguish themselves by being Hermitian, unitary, and having eigenvalues $\{-1, 1\}$. They correspond to measuring the projection of the spin on the qubit, an intrinsic property of a particle analogous to angular momentum, along a particular dimension [Wik24g].
We thus have a total of $3^n$ possible experimental observables and $2^n$ possible outcomes for each observable. Based on the observable and outcome, we can create the operator $P^{\mathbf{a}}_{\mathbf{s}} = P^{a_1}_{s_1} \otimes \cdots \otimes P^{a_n}_{s_n}$, which corresponds to the projector for this $(\mathbf{a}, \mathbf{s})$ pair. The term $P^{a_i}_{s_i}$ corresponds to the orthogonal projection associated to the eigenvalue $s_i$ in the diagonalization of $\sigma_{a_i}$, which in turn can be rewritten as $\sigma_{a_i} = -1P^{a_i}_{-1} + 1P^{a_i}_{1}$ due to its Hermitian property. Note that under this notation, $\mathbf{a}$ and $\mathbf{s}$ are both vectors *and* indices. Reminding ourselves of the POVM and Born's rule we saw in 2.3.1, we can formulate the main equation as

$$\forall \mathbf{s} \in \mathcal{R}^n, p_{\mathbf{a},\mathbf{s}} = \Pr(R^{\mathbf{a}} = \mathbf{s}) = \operatorname{tr}(\rho P^{\mathbf{a}}_{\mathbf{s}}) \tag{2.11}$$

where $R^{\mathbf{a}} \in \mathcal{R}^n$ is the random vector outcome of the measurement with $\mathbf{a}$ and $p_{\mathbf{a},\mathbf{s}}$ the probability linked to that pair. We can see that we recover the projective measurement (or more generally the POVM), only extended to the multi-qubit case by the Kronecker product applied to the projectors $P_{\mathbf{s}}^{\mathbf{a}}$.

An important element to take into account when working with this method in practice is that, while it is the theoretically correct approach, it requires on the order of $6^n = 2^n 3^n$ operations, making it costly.

### Mixed qubit data generation

Mixed qubit data generation (also called "Pauli observables" in [Guţ+20]) is an alternative approach, where instead of calculating the probability for each observable/outcome pair, we will directly approximate the expected value linked to this observable. Its calculation is much more straightforward: if we call $A_m$ the observable of which we calculate the expected value, then $A_m = \sigma_{m_1} \otimes \sigma_{m_2} \cdots \otimes \sigma_{m_n}$ with $n$ the number of qubits and $m_i$ identifying the Pauli matrix for qubit $i$ for combination $m$ (the notation is the same as in [Ade+24]). The measured value for each observable is $\hat{p}_m \in [-1, 1]$. This results in a total of $4^n$ combinations ($\sigma_{m_i} \in \{I, \sigma_x, \sigma_y, \sigma_z\}$), hence less costly than the separate qubit process.

## 2.4 Markov Chain Monte Carlo methods

Markov Chain Monte Carlo (MCMC) methods are a class of sampling algorithms. Sampling from a probability distribution is relevant when we want to compute some statistic about it, for example, the mean. In most cases, the target distribution $\pi(\mathbf{x})$ has high dimensionality as we work in the space of parameters of statistical models. MCMC is particularly important in the context of *Bayesian inference*. In this case, $\pi(\mathbf{x})$ corresponds to the posterior distribution from which we want to sample, and whose probability density function usually corresponds to a complex expression.

Various methods exist for sampling from simple distributions, the most famous ones being the inverse transform method, rejection sampling, or importance sampling. All these methods however have drawbacks: inverse transform requires an analytical solution for the cumulative density function, which is usually not possible to obtain for complex probability density functions as it requires a proper integral; rejection sampling and importance sampling do not have this problem, but they suffer from the curse of dimensionality - once your dimension grows, the sampling becomes very inefficient. MCMC methods solve these issues (to a certain degree), with some methods being more suitable than others depending on the specific situation.

We will now cover several aspects that makeup MCMC: Bayesian inference in 2.4.1, the theoretical guarantees behind the MCMC methods in 2.4.2, and finally the 3 main algorithms we will be concerned about in this thesis, Metropolis-Hastings in 2.4.3, Gibbs sampling in 2.4.4 and Langevin sampling in 2.4.5.

### 2.4.1 Bayesian inference

As its name suggests, Bayesian inference relies on Bayes theorem. In the context of algorithms, it states that given data $\mathbf{D}$ and parameters $\boldsymbol{\theta}$, the *posterior* distribution $\pi(\boldsymbol{\theta}|\mathbf{D})$ is calculated as

$$\pi(\boldsymbol{\theta}|\mathbf{D}) = \frac{\pi(\mathbf{D}|\boldsymbol{\theta})\pi(\boldsymbol{\theta})}{\pi(\mathbf{D})} = \frac{\pi(\mathbf{D}|\boldsymbol{\theta})\pi(\boldsymbol{\theta})}{\int_{\boldsymbol{\theta}} \pi(\mathbf{D}|\boldsymbol{\theta})\pi(\boldsymbol{\theta})d\boldsymbol{\theta}} \tag{2.12}$$

where $\pi(\mathbf{D}|\boldsymbol{\theta})$ is the *likelihood* of data given the parameters and $\pi(\boldsymbol{\theta})$ the *prior* distribution we put on the parameters. The denominator

$$\pi(\mathbf{D}) = \int_{\boldsymbol{\theta}} \pi(\mathbf{D}|\boldsymbol{\theta})\pi(\boldsymbol{\theta})d\boldsymbol{\theta} \tag{2.13}$$

is called the *marginal*, and corresponds to a normalizing constant. It is in practice intractable to calculate as it involves performing a very high dimensional integral numerically, which is unstable. The good news is that in the context of MCMC methods, the normalizing factor is not important and we can safely remove it. This gives the following posterior:

$$\pi(\boldsymbol{\theta}|\mathbf{D}) \propto \pi(\mathbf{D}|\boldsymbol{\theta})\pi(\boldsymbol{\theta}) \tag{2.14}$$

Getting samples from $\pi(\boldsymbol{\theta}|\mathbf{D})$ allows us to compute integrals, a technique called Monte Carlo integration. For example, if we have a function $g$ defined on $\mathbb{R}^n$ and we want to approximate its integral on a subset $\Omega \subseteq \mathbb{R}^n$, we can sample $N$ values uniformly $\{\boldsymbol{\theta}_i\}_{i=1}^N \in \Omega$ and calculate

$$\int_\Omega g(\boldsymbol{\theta}) d\boldsymbol{\theta} \approx \frac{1}{N} \sum_{i=1}^{N} g(\boldsymbol{\theta}_i) \tag{2.15}$$

The law of large numbers ensures that with $N \to \infty$, the discrete sum converges to the true integral [Wik24f].

In statistics, integrals are of particular interest to us as they correspond to expectations. If we have function $f(\boldsymbol{\theta})$ defined on some set $\boldsymbol{\Theta}$, then

$$\mathbb{E}_{\boldsymbol{\theta} \sim \pi(\boldsymbol{\theta}|\mathbf{D})} \left[ f(\boldsymbol{\theta}) \right] = \int_{\boldsymbol{\theta}} f(\boldsymbol{\theta}) \pi(\boldsymbol{\theta}|\mathbf{D}) d\boldsymbol{\theta} \approx \frac{1}{N} \sum_{i=1}^{N} f(\boldsymbol{\theta}_i) \tag{2.16}$$

where $\{\boldsymbol{\theta}_i\}_{i=1}^N$ are samples from $\pi(\boldsymbol{\theta}|\mathbf{D})$. This allows us to compute estimates for a random variable (with $f(\boldsymbol{\theta}) = \boldsymbol{\theta}$), vector, or matrix, the last one being the most relevant in Quantum Tomography. In that case, the parameters correspond to the density matrix $\rho$, and the data to the projectors $P_{\mathbf{s}}^{\mathbf{a}}$ and the empirical probabilities $\hat{p}_{\mathbf{a},\mathbf{s}}$. By first putting a prior on $\rho$, we can obtain a sample estimate of $\rho$, balanced by the data we have. This is where the term *inference* comes from, as we infer $\rho$ from the data and prior.

## 2.4.2 Theoretical guarantees

In contrast to Monte Carlo methods where the samples are independent, MCMC builds a *Markov chain* of samples $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(R)}$, which are dependent (we use $\mathbf{x}$ instead of $\boldsymbol{\theta}$ to be as general as possible). As a reminder, a sequence $\mathbf{X}^{(1)}, \mathbf{X}^{(2)} \ldots \mathbf{X}^{(R)}$ of random variables is a Markov chain if

$$\Pr(\mathbf{X}^{(r+1)} \in A | \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(r)}) = \Pr(\mathbf{X}^{(r+1)} | \mathbf{x}^{(r)}) \tag{2.17}$$

where $\mathbf{x}^{(i)}$ is a realization of $\mathbf{X}^{(i)}$ on domain $A$ (which is usually multidimensional hence the bold notation, however in our definitions it will be one-dimensional). One question remains: how can we make sure that samples we obtain from some Markov chain indeed come from the distribution we sample from? To answer this, we need to review several properties of Markov chains.

A *transition kernel* $k$ is a function that fully characterizes a Markov chain. It gives the probability of transitioning from one state to another in a chain: in the continuous case, this corresponds to conditional function $k(\mathbf{x}^{(r+1)}|\mathbf{x}^{(r)})$ and the discrete case, it is a transition matrix $P$. We say that our chain in *invariant* to a distribution $\pi(\mathbf{x})$ if

$$\pi(\mathbf{x}^*) = \int k(\mathbf{x}^*|\mathbf{x}) \pi(\mathbf{x}) d\mathbf{x} \tag{2.18}$$

where we transition from state $\mathbf{x}$ to $\mathbf{x}^*$. This corresponds to saying that our chain admits a *stationary* distribution, or equivalently that the *marginal* distributions of $\mathbf{X}^{(r)}$ and $\mathbf{X}^{(r+1)}$ are the same. The distribution $\pi(\mathbf{x})$, called the *target* density, corresponds to the distribution we want to sample from in the context of MCMC. It is important to note that not all chains admit a stationary distribution, and a few properties must be respected. We will cover the discrete case for brevity reasons, but this of course extends to the continuous one [Rig24; Wik24d].

- **Irreducibility**: a Markov chain is *irreducible* if it explores the entire sample space, and does not get stuck in local regions. Formally, if $\mathbf{X}^{(r)} \in \mathbb{N}$, this corresponds to the condition

$$\Pr(\tau_j < \infty | \mathbf{x}^{(0)} = j') > 0 \quad \forall j, j' \in \mathbb{N} \tag{2.19}$$

  where $\tau_j = \inf\{r \geq 1 : \mathbf{X}^{(r)} = j\}$ is the first passage time for which the chain is equal to $j$ (defined as $\infty$ if $\mathbf{X}^{(r)} \neq j$ for every $r \geq 1$) and $\Pr(\tau_j < \infty | \mathbf{x}^{(0)} = j')$ the probability of return to $j$ in a finite number of steps.

- **Aperiodicity**: a Markov chain is *aperiodic* if it does not have any deterministic cycles. Formally, we say that a state $j$ is aperiodic if the set $\{r \geq 1 : [P^r]_{jj} > 0\}$ has no common divisor other than 1. A chain is aperiodic if all of its states are aperiodic.

- **Harris recurrence**: a Markov chain is *recurrent* if it visits any region of the sample space sufficiently often. Formally, a state $j$ of a discrete irreducible Markov chain is recurrent if and only if

$$\Pr(\tau_j < \infty | \mathbf{x}^{(0)} = j) = \Pr(\mathbf{X}^{(r)} = j \text{ for infinitely many } r | \mathbf{x}^{(0)} = j) = 1 \tag{2.20}$$

  In other words, it means that this state is visited infinitely often. If all states are recurrent, then the chain is called recurrent. If additionally, this chain admits an invariant distribution, then it is Harris positive (also called positive recurrent).

- **Ergodicity**: a Markov chain in *ergodic* if every state is ergodic. A state $j$ is ergodic if it is aperiodic and positive recurrent. Intuitively, this property means that we can reach every state with a probability greater than 0.

We can finally state the main result that allows us to use Markov chains for MCMC [RC04]:

**Theorem 1** *A Markov chain Monte Carlo (MCMC) method converges to a distribution $\pi(\mathbf{x})$ if the underlying Markov chain is ergodic with an equilibrium distribution $f(\mathbf{x}) = \pi(\mathbf{x})$.*

A closely related theorem that provides a guarantee similar to the law of large numbers for Markov chains is the *Ergodic* Theorem [Rig24]:

**Theorem 2** *Let the Markov chain $(\mathbf{X}^{(r)})_{r\geq 1}$ be Harris positive with stationary distribution $\pi$, and $g$ be integrable with respect to $\pi$. Then,*

$$\frac{1}{R}\sum_{r=1}^{R} g(\mathbf{X}^{(r)}) \to \int g(\mathbf{x})\pi(\mathbf{x})d\mathbf{x} \quad \textit{when } R \to \infty \tag{2.21}$$

*almost surely.*

### 2.4.3 Metropolis-Hastings algorithm

The *Metropolis-Hastings* algorithm is the most well-known and used MCMC algorithm. The Markov chain of samples is built by doing a random walk, with an acceptance/rejection step. It works as follows: given that we are currently located at $\mathbf{x} \in \mathbb{R}^n$, we will sample a potential candidate $\mathbf{x}^*$ from $q(\mathbf{x}^*|\mathbf{x})$ where $q$ is called the *proposal* (or *jumping*) distribution. This $q$ is analogous to the kernel we saw in 2.4.2, although not exactly as we will see later (see equation 2.26). Next, we will compute the *acceptance* probability of this sample:

$$\alpha(\mathbf{x}^*, \mathbf{x}) = \min\left(1, \frac{\pi(\mathbf{x}^*)q(\mathbf{x}|\mathbf{x}^*)}{\pi(\mathbf{x})q(\mathbf{x}^*|\mathbf{x})}\right) \tag{2.22}$$

where $\pi(\mathbf{x})$ is the posterior distribution we would like to sample from. Note that we can simplify this criterion in case the proposal density is symmetric ($q(\mathbf{x}^*|\mathbf{x}) = q(\mathbf{x}|\mathbf{x}^*)$):

$$\alpha(\mathbf{x}^*, \mathbf{x}) = \min\left(1, \frac{\pi(\mathbf{x}^*)}{\pi(\mathbf{x})}\right) \tag{2.23}$$

This happens often in practice, as the Gaussian distribution is a common choice of proposal. Another comment is that, as mentioned in section 2.4.1, the normalizing constant of the posterior is not important: it gets simplified in the numerator and denominator as it does not depend on $\mathbf{x}$, nor $\mathbf{x}^*$ (only the data $\mathbf{D}$, see equation 2.12).

After this computation, we will accept $\mathbf{x}^*$ with probability $\alpha$. In practice, this involves, for example, sampling a value $u \sim U(0, 1)$, and accepting if $u \leq \alpha$. This procedure is repeated until the desired number of samples is obtained, with some part of the samples that we discard at the beginning. This period is called the *burn-in* period and corresponds to the duration when the samples we obtain are not part of the equilibrium distribution. We usually talk about *convergence* when the samples come from $\pi$ (the posterior distribution we want to sample from), or that the Markov chain has *mixed*.

A pseudocode for the algorithm could be:

**Algorithm 1:** Metropolis-Hastings algorithm

```
   Input  : $\mathbf{x}^{(0)} \in \mathbb{R}^n, T \in \mathbb{N}$
   Output: $\mathbf{x}^{(1)} \in \mathbb{R}^n, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(T)}$
1  for $t \leftarrow 1 : T$ do
2      Sample $\mathbf{x}^* \sim q(\mathbf{x}|\mathbf{x}^{(t-1)})$
3      $\alpha = \min\left(1, \frac{\pi(\mathbf{x}^*)q(\mathbf{x}^{(t-1)}|\mathbf{x}^*)}{\pi(\mathbf{x}^{(t-1)})q(\mathbf{x}^*|\mathbf{x}^{(t-1)})}\right)$
4      Sample $\mathbf{x}^* \sim U(-0.5, 0.5)$
5      if $u \leq \alpha$ then
           // Accept $\mathbf{x}^*$
6          $\mathbf{x}^{(t)} \leftarrow \mathbf{x}^*$
7      else
           // Reject $\mathbf{x}^*$
8          $\mathbf{x}^{(t)} \leftarrow \mathbf{x}^{(t-1)}$
9      end
10 end
```

where $T$ is the number of iterations.

Note that in most implementations, it is common practice to use $\log(\alpha)$ for the calculation of the acceptance probability, instead of $\alpha$ directly. This avoids numerical problems due to divisions by small numbers and underflow/overflow that might occur when handling double-precision floats. In addition, using the logarithm combines nicely with distributions based on the exponential family, allowing for a log(exp) simplification.

**Convergence of Metropolis-Hastings** We have in section 2.4.2 talked about convergence for Markov chains to the posterior distribution, but have not talked about how that would work in practice. A very handy way is to use the *detailed balance* condition.

**Theorem 3** *A Markov chain* $(\mathbf{X}^{(r)})_{r\geq 1}$ *with transition kernel* $k$ *satisfies the detailed balance if there exists a function* $f$ *such that*

$$k(\mathbf{x}|\mathbf{x}^*)f(\mathbf{x}) = k(\mathbf{x}^*|\mathbf{x})f(\mathbf{x}^*) \tag{2.24}$$

In this context, we also talk about *reversibility*:

**Theorem 4** *A Markov chain* $(\mathbf{X}^{(r)})_{r\geq 1}$ *is reversible if*

$$\Pr(\mathbf{X}^{(r)}|\mathbf{X}^{(r+1)}) = \Pr(\mathbf{X}^{(r+1)}|\mathbf{X}^{(r)}) \tag{2.25}$$

Finally, we can reach an important and useful result for MCMC:

**Theorem 5** *If a Markov chain* $(\mathbf{X}^{(r)})_{r\geq 1}$ *satisfies the detailed balance with* $\pi$ *a probability density function, then* $\pi$ *is the stationary density, and chain is reversible.*

This result means that, as long as we satisfy the detailed balance with our MCMC algorithm, we will converge to the posterior distribution $\pi$. In the case of Metropolis-Hastings, the transition kernel is

$$k\left(\boldsymbol{x}^* \mid \boldsymbol{x}\right) = \alpha\left(\boldsymbol{x}^*, \boldsymbol{x}\right) q\left(\boldsymbol{x}^* \mid \boldsymbol{x}\right) + \delta_{\boldsymbol{x}}\left(\boldsymbol{x}^*\right) \int q(\boldsymbol{s} \mid \boldsymbol{x})\{1-\alpha(\boldsymbol{s} \mid \boldsymbol{x})\} \mathrm{d}\boldsymbol{s} \tag{2.26}$$

where $\delta_{\mathbf{x}}(\mathbf{x}^*)$ is a point mass at $\mathbf{x}$. Putting this kernel into the detailed balance condition 2.24, one can derive that the equality indeed holds. The only element that we have not discussed so far is the ergodicity condition. Although it depends on $\pi$ and $q$, it is typically true under very mild conditions, which can be checked in, for example, [RC04].

A final element to take into account with theoretical convergence guarantees is that in real-world experiments, the algorithm may take a very long time to converge, making it impractical for real use. In these situations, one could for example investigate different proposals, change the duration of the burn-in period, or even swap Metropolis-Hastings for another algorithm. Various metrics exist to quantify the convergence, such as the effective sample size or $\hat{R}$, but also visual checks, which are effective for checking if the chain has mixed. It is also common practice to run several chains from different starting points and assess the overall performance. More details about practical considerations can be found in [Gel+13].

### 2.4.4 Gibbs sampling algorithm

The Gibbs sampling algorithm takes a different approach compared to Metropolis-Hastings in that it splits the joint distribution $\pi(\mathbf{x}) = \pi(x_1, x_2, \ldots, x_n)$ with $\mathbf{x} \in \mathbb{R}^n$ into one-dimensional conditionals, and then step samples from each conditional $\pi(x_i|\mathbf{x}_{-i}) = \pi(x_i|x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$ iteratively. It can also be generalized to blocks, where instead of sampling from a one-dimensional distribution, we sample from a joint conditional.

It is, in its simplest form, an *unadjusted* algorithm, as it does not involve an accept/reject step. A pseudocode for it can be the following:

**Algorithm 2:** Gibbs sampling algorithm

```
   Input  : $\mathbf{x}^{(0)}, T \in \mathbb{N}$
   Output: $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(T)}$
1  for $t \leftarrow 1 : T$ do
2      $\mathbf{x}^{(t)} \leftarrow \mathbf{x}^{(t-1)}$
3      for $i \leftarrow 1 : n$ do
           // Sample from the conditional and set $i$-th element to this
              value
4          $\mathbf{x}_i^{(t)} \sim \pi(x_i|\mathbf{x}_{-i}^{(t)})$
5      end
6  end
```

Note that the inner loop updates the current $\mathbf{x}^{(t)}$, and not a copy. This implies that at iteration $i$, we are using the conditioned value set at $i-1$ for $x_{i-1}^{(t)}$, and not the one set at $x_{i-1}^{(t-1)}$.

The main advantages of Gibbs sampling is that we do not need to tune the proposal distributions, as well as the fact that all samples are accepted. The downside of course is that we need to be able to derive and sample from the conditional, especially doing so efficiently as otherwise this approach becomes very costly. A more general algorithm called *Metropolis-within-Gibbs* involves sampling from the conditional using Metropolis-Hastings, and in that case, this may involve an accept/reject step (an example of which is the prob-estimator in section 3.1 or the MHGS algorithm in section 5.1).

In the case of standard Gibbs, we may view the conditionals we sample from as proposal distributions $q(\mathbf{x}^*|\mathbf{x})$. This in turn shows us why this algorithm is unadjusted, as the acceptance probability $\alpha$ is always equal to 1 [Rig24]. If we call $\mathbf{x}^* = (x_1^{(t)}, \ldots, x_i^*, \ldots, x_n^{(t)})$ and $\mathbf{x} = (x_1^{(t)}, \ldots, x_i^{(t)}, \ldots, x_n^{(t)})$, then

$$\alpha(\mathbf{x}^*|\mathbf{x}) = \frac{\pi(\mathbf{x}^*)q(\mathbf{x}|\mathbf{x}^*)}{\pi(\mathbf{x})q(\mathbf{x}^*|\mathbf{x})} = \frac{\pi(x_i^*|\mathbf{x}_{-i}^*)\pi(\mathbf{x}_{-i}^*)\pi(x_i|\mathbf{x}_{-i}^*)}{\pi(x_i|\mathbf{x}_{-i})\pi(\mathbf{x}_{-i})\pi(x_i^*|\mathbf{x}_{-i})} = \frac{\pi(x_i^*|\mathbf{x}_{-i})\pi(\mathbf{x}_{-i})\pi(x_i|\mathbf{x}_{-i})}{\pi(x_i|\mathbf{x}_{-i})\pi(\mathbf{x}_{-i})\pi(x_i^*|\mathbf{x}_{-i})} = 1 \tag{2.27}$$

where we have used the fact that $\mathbf{x}_{-i}^* = \mathbf{x}_{-i}$.

### 2.4.5 Unadjusted Langevin algorithm

One big downside with Metropolis-Hastings or Gibbs sampling is that they can take a long time to converge, especially in a very high-dimensional setting. The Unadjusted Langevin algorithm (ULA) tackles this problem by using the gradient information of the posterior. The downside, of course, is that we must be able to calculate it, but in the case we can, it can drastically speed up the convergence.

The algorithm draws its name from *Langevin diffusion* (also called *Îto diffusion*), a stochastic differential equation (SDE), written as follows:

$$\mathrm{d}\mathbf{X}_t = -\nabla V(\mathbf{x})\mathrm{d}t + \sqrt{2}\mathrm{d}\mathbf{B}_t \tag{2.28}$$

where $\mathbf{X}_t$ is the position of a particle in a potential $V(\mathbf{x})$ and $\mathbf{B}_t$ is the time derivative of the standard Brownian motion. An SDE is a differential equation where one or more of the terms is a stochastic process and results in a solution which is also a stochastic process. In the case of Langevin diffusion, the solution happens to be a stationary process $p_\infty(\mathbf{x}) \propto \exp(-V(\mathbf{x}))$, which is very handy in the context of MCMC. If we set $V(\mathbf{x}) = -\log(\pi(\mathbf{x}))$, then we can simulate the SDE and obtain samples from $p_\infty = \pi$:

$$\mathrm{d}\mathbf{X}_t = \nabla \log(\pi(\mathbf{x}))\mathrm{d}t + \sqrt{2}\mathrm{d}\mathbf{B}_t \tag{2.29}$$

Of course, to simulate, we must discretize the SDE in some way, and in this case, the *Euler-Maruyama approximation* is what is commonly used. This results in

$$\mathbf{X}^{(k+1)} - \mathbf{X}^{(k)} = \tau\nabla \log \pi(\mathbf{X}^{(k)}) + \sqrt{2\tau}(\mathbf{B}^{(k+1)} - \mathbf{B}^{(k)}) \tag{2.30}$$

where $\tau$ is a step size that can be changed over time and $(\mathbf{B}^{(k+1)} - \mathbf{B}^{(k)}) \sim N(\mathbf{0}, \mathbf{I})$. We can then rewrite it as follows:

$$\mathbf{X}^{(k+1)} = \mathbf{X}^{(k)} + \tau\nabla \log \pi(\mathbf{X}^{(k)}) + \sqrt{2\tau}\boldsymbol{\xi}^{(k)} \tag{2.31}$$

where $\boldsymbol{\xi}^{(k)} \sim N(\mathbf{0}, \mathbf{I})$. Note that the true solution can be computed with $\mathbf{X}(k\tau)$. The pseudocode can be written as follows:

**Algorithm 3:** Unadjusted Langevin algorithm

```
  Input  : $\mathbf{x}^{(0)}, K, \tau^{(1)}, \dots, \tau^{(K)}$
  Output: $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(K)}$
1 for $k \leftarrow 1 : K$ do
2     $\boldsymbol{\xi}^{(k)} \sim N(\mathbf{0}, \mathbf{I})$
3     $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \tau^{(k)}\nabla \log \pi(\mathbf{x}^{(k)}) + \sqrt{2\tau^{(k)}}\boldsymbol{\xi}^{(k)}$
4 end
```

with $K$ the number of iterations and $\tau^{(k)}$ the step size.

ULA can be seen as noisy gradient ascent, which is why it performs better than Metropolis-Hastings or Gibbs. By utilizing the gradient information, we are able to sample more often the regions of high density. The cost of computing the gradient is usually compensated by faster convergence times, making it often an algorithm of choice if the gradient of the posterior can be computed. The algorithm is relevant in this thesis due to its use in the Projected Langevin method (see section 3.2). Note that there also exists a method called *Metropolis-adjusted Langevin algorithm* (MALA), which generalizes this approach with an accept/reject step, but we will not go into the details here [Ans24; Wik24e].

# Chapter 3

# Overview of relevant methods

In this chapter, we will introduce two MCMC algorithms that are in relevant in the context of this thesis: the *prob-estimator* (in 3.1) and the *Projected Langevin algorithm* (in 3.2). They will be useful in Chapter 4 where we will be comparing them in numerical experiments and Chapter 5 where we will be presenting 2 new algorithms derived from them.

Note that both papers provide theoretical convergence bounds for the method they propose, however, we will not describe them here as they are not relevant in the context of our experiments. Also note that the notations we use in this chapter differ from the ones used in the MCMC chapter 2.4, as we prefer to keep to the notations used in the original papers.

## 3.1 Prob-estimator

Introduced in [MA17], the *prob-estimator* is a Metropolis-Hastings-based algorithm (more details about it can be found in section 2.4.3), where the approximation of the density matrix $\rho$ is done using a sum of rank 1 matrices and sparse coefficients, leading to a rank 1 approximation. The exact solution is written as follows:

$$\rho = \sum_{i=1}^{d} \gamma_i V_i V_i^\dagger \tag{3.1}$$

where $V_i \in \mathbb{C}^{d\times 1}$ are normalized vectors, $\gamma_i \in \mathbb{C}$ are the sparse coefficients with $\sum_{i=1}^{d} \gamma_i = 1$, and $\rho \in \mathbb{C}^{d\times d}$ the density matrix with $d = 2^n$ and $n$ the number of qubits.

We can check that the properties of a density matrix are respected: it is a sum of Hermitian matrices, which also makes $\rho$ Hermitian and positive semi-definite, and

$$\mathrm{tr}(\rho) = \sum_{i=1}^{d} \gamma_i \mathrm{tr}(V_i V_i^\dagger) = 1 \tag{3.2}$$

as $\mathrm{tr}(V_i V_i^\dagger) = 1$ due to the normalization.

This way of calculating $\rho$ is analogous to taking an eigenvector decomposition

$$\rho = U \Lambda U^\dagger \tag{3.3}$$

where $U = (U_1|\dots|U_d)$ with $U_i$ orthogonal to each other and of norm 1, and $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \dots, \lambda_d)$ with $\sum_{i=1}^{d} \lambda_i = 1$ (due to $\rho$ being a density matrix).

The advantage of the approach from equation 3.1 is that it does not require the $V_i$ to be orthonormal (as they are in the case of a decomposition of a symmetric/Hermitian matrix).

As we are using an MCMC method, we need to formulate the posterior $\pi(\nu|\mathbf{D})$ from which we will be sampling, with $\nu = \sum_{i=1}^{d} \gamma_i V_i V_i^\dagger$ our parameter matrix and the data $\mathbf{D}$ comprised of the projectors $P_{\mathbf{s}}^{\mathbf{a}}$ and empirical probabilities $\hat{p}_{\mathbf{a},\mathbf{s}}$. In the case of the prob-estimator, the authors use a pseudo-likelihood written as follows

$$\pi(\mathbf{D}|\nu) = \mathcal{L}(\nu, \mathbf{D}) = \exp(-\lambda \ell(\nu, \mathbf{D})) \tag{3.4}$$

where $\lambda$ is a weight parameter for the likelihood and $\ell(\nu, \mathbf{D})$ the loss function

$$\ell(\nu, \mathbf{D}) = \sum_{\mathbf{a}\in\mathcal{E}^n} \sum_{\mathbf{s}\in\mathcal{R}^n} \left[\mathrm{tr}(\nu P_{\mathbf{s}}^{\mathbf{a}}) - \hat{p}_{\mathbf{a},\mathbf{s}}\right]^2 \tag{3.5}$$

The authors use separate qubit data generation, thus all details regarding notation can be checked in 2.3.2. Note that the name “prob-estimator” comes from using this particular loss, as it quantifies how far the probabilities and the empirical frequencies in the sample are.

The prior $\pi(\nu) = \pi_1(V_1, \dots, V_d)\pi_2(\gamma_1, \dots, \gamma_d)$ is defined in 2 parts:

- For $\pi_1(V_1, \dots, V_d)$, the set $V_1, \dots, V_d$ is i.i.d uniformly distributed on the unit sphere. In practice, this involves first sampling from a standard normal multivariate $\tilde{V}_i \sim N(\mathbf{0}, \mathbf{I})$, and then normalizing it $V_i = \tilde{V}_i/||\tilde{V}_i||$.

- For $\pi_2(\gamma_1, \dots, \gamma_d)$, the set $\gamma_1, \dots, \gamma_d$ is sampled from a Dirichlet distribution $\mathcal{D}ir(\alpha_1, \dots, \alpha_d)$ with parameter $\alpha_i > 0$. In order to promote sparsity among $\gamma_1, \dots, \gamma_d$ and get a rank-1 approximation of $\rho$, the parameters are all chosen to be the same, small value: $\alpha_1, \dots, \alpha_d = \alpha = 1/d$. This results in a typical drawing of one $\gamma_i$ close to 1, while others close to 0. In practice, the authors do not sample the Dirichlet distribution directly, and instead use an equivalent formulation: $\gamma_i = Y_i/(\sum_{j=1}^{d} Y_j)$ where $Y_i$ is sampled from a Gamma distribution $Y_i \sim \mathit{Gamma}(\alpha, 1)$.

The posterior is then written as

$$\pi(\nu|\mathbf{D}) \propto \exp(-\lambda \ell(\nu, \mathbf{D}))\pi(\nu) \tag{3.6}$$

and estimated with

$$\hat{\rho}_{\text{prob}} = \int \nu \pi(\nu|\mathbf{D}) d\nu \tag{3.7}$$

As specified above, the prior is split into 2 parts. This is due to the way the algorithm is structured: inside the main loop of Metropolis-Hastings (which goes over iterations $1 \to T$), we also iterate over each dimension $1 \to d$, first for $\gamma$, and then for $V$. This can be seen as doing Metropolis-within-Gibbs.

The pseudocode for the algorithm is available below:

**Algorithm 4:** Prob-estimator algorithm

```
Input  : $d = 2^n, \lambda \in \mathbb{R}, \alpha \in \mathbb{R}, T \in \mathbb{N}, V^{(0)} \in \mathbb{C}^{d\times d}, Y^{(0)} \in \mathbb{C}^{d\times 1}$
Output: $\hat{\rho} \in \mathbb{C}^{d\times d}$
1  $\hat{\rho} \leftarrow \mathbf{0}$
2  $\gamma^{(0)} \in \mathbb{R}^{d\times 1} \leftarrow Y^{(0)}/(\sum_{i=1}^{d} Y_i^{(0)})$
3  for $t \leftarrow 1 : T$ do
4      $Y^{(t)} \leftarrow Y^{(t-1)}$
       // Update $\gamma$
5      for $i \leftarrow 1 : d$ do
6          $\tilde{Y} \leftarrow Y^{(t)}$
7          Sample $y \sim U(-0.5, 0.5)$
8          $\tilde{Y}_i \leftarrow Y_i^{(t)} \exp(y)$
9          $\tilde{\gamma} \leftarrow \tilde{Y}/(\sum_{j=1}^{d} \tilde{Y}_j)$
10         $Y_i^{(t)} \leftarrow$
           $\begin{cases} \tilde{Y}_i & \text{with probability} \min\{R(\tilde{Y}, Y^{(t)}, \tilde{\gamma}, \gamma^{(t)}, V^{(t-1)}, \lambda, \alpha), 1\} \\ Y_i^{(t)} & \text{otherwise} \end{cases}$ // $R$ defined below
11         $\gamma^{(t)} \leftarrow Y^{(t)}/(\sum_{k=1}^{d} Y_k^{(t)})$ // Update all components
12     end
13
14     $V^{(t)} \leftarrow V^{(t-1)}$
       // Update $V$
15     for $i \leftarrow 1 : d$ do
16         $\tilde{V} \leftarrow V^{(t)}$
17         Sample $V \sim N(\mathbf{0}, \mathbf{I})$
18         $\tilde{V}_i \leftarrow (V + V_i^{(t)})/||V + V_i^{(t)}||_2$ // Sample from the unit sphere
19         $V_i^{(t)} \leftarrow$
           $\begin{cases} \tilde{V}_i & \text{with probability} \min\{A(\tilde{V}, V^{(t)}, \gamma^{(t)}, \lambda), 1\} \\ V_i^{(t)} & \text{otherwise} \end{cases}$ // $A$ defined below
20     end
21
       // Compute a running average estimate
22     $\hat{\rho} \leftarrow \frac{1}{t}\sum_{i=1}^{d} \gamma_i^{(t)} V_i^{(t)} (V_i^{(t)})^\dagger + (1 - \frac{1}{t})\hat{\rho}$
23 end
```

where $\lambda$ is a weight parameter for the likelihood, $\alpha$ a scale parameter for the prior, $T$ the number of iterations and $V^{(0)}, Y^{(0)}$ the initial samples. There is also a burn-in period and a log-transform trick involved, not indicated above for brevity reasons.

The log acceptance rates are the following:

$$\log R(\tilde{Y}, Y^{(t)}, \tilde{\gamma}, \gamma^{(t)}, V^{(t-1)}, \lambda, \alpha) = -\lambda\ell(\sum_{i=1}^{d} \tilde{\gamma}_i V_i^{(t-1)}(V_i^{(t-1)})^\dagger, \mathbf{D}) + \lambda\ell(\sum_{i=1}^{d} \gamma_i^{(t)} V_i^{(t-1)}(V_i^{(t-1)})^\dagger, \mathbf{D}) \tag{3.8}$$

$$+ (\alpha - 1)\log\left(\frac{\tilde{Y}_i}{Y_i^{(t)}}\right) - \tilde{Y}_i + Y_i^{(t)} \tag{3.9}$$

and

$$\log A(\tilde{V}, V^{(t)}, \gamma^{(t)}, \lambda) = -\lambda\ell(\sum_{i=1}^{d} \gamma_i^{(t)} \tilde{V}_i \tilde{V}_i^\dagger, \mathbf{D}) + \lambda\ell(\sum_{i=1}^{d} \gamma_i^{(t)} V_i^{(t)}(V_i^{(t)})^\dagger, \mathbf{D}) \tag{3.10}$$

The curious reader can find the detailed derivations in the appendix 6.

Note that there is an element which is ambiguous in the prob-estimator. The lines 7 and 8 of pseudocode 4 correspond to the sampling from $Gamma(\alpha, 1)$. It is however unclear on why this would correspond to obtaining $\tilde{Y}_i \sim Gamma(\alpha, 1)$. There exists a classical procedure where one uses the inverse transform method and the fact that if $X_i \sim \exp(\lambda)$, then

$$\sum_{i=1}^{t} X_i \sim Gamma(t, \lambda) \tag{3.11}$$

however this only works in the case where $t \in \mathbb{N}$. The Wikipedia page [Wik24c] mentions that the case with $0 < \alpha < 1$ (assumed with $\beta = 1$) is the most challenging to sample from, with new research [LMS15] still being done on this topic. The Python library `numpy` [Har+20] does it using a form of rejection sampling[1], similar to [LMS15]. These approaches are however very different from what is done in the prob-estimator, and as the authors do not indicate why it is correct, it is difficult to understand the reasoning behind it.

[1]https://github.com/numpy/numpy/blob/main/numpy/random/src/distributions/distributions.c#220

## 3.2 Projected Langevin algorithm

Introduced in the [Ade+24] paper (unpublished at the moment of submission of this thesis), the Projected Langevin algorithm is based on an Unadjusted Langevin algorithm (ULA) (more details about it can be found in section 2.4.5). The authors utilize the Burer-Monteiro factorization (introduced in [BM03]) $\rho = YY^\dagger$ with $Y \in \mathbb{C}^{d\times r}$ (with $d = 2^n$ and $n$ the number of qubits), which corresponds to a low rank (of rank $r$) factorization of $\rho$. This corresponds to considering that the density matrix has a maximum rank of $r$. This also allows us to reduce the dimension of the parameter space from $2d^2$ to $2dr$ by running the MCMC method directly on the $Y$ space. It is important to note that the physical constraints of the problem are still respected, as we assume the matrix $Y$ to belong to the complex hypersphere $C\mathbb{S}^{d\times r} = \{Y \in \mathbb{C}^{d\times r} : ||Y||_F = 1\}$, leading to a unit trace. The Hermitian and positive semi-definite properties of the density matrix are obtained by construction.

In addition to the factorization, the paper proposes a new prior, called the spectral scaled Student-t distribution defined as

$$\nu_\theta(Y) = C_\theta \det(\theta^2 I_d + YY^\dagger)^{-(2d+r+2)/2} \tag{3.12}$$

where $\theta$ is a scaling parameter.

This prior is an extension of a prior proposed by [Dal20] to the complex domain, where the Student-t distribution is a generalization of the Gaussian distribution, allowing for fatter tails. This feature is what promotes sparsity in the eigenvalues of $Y$, favoring a low rank (most samples are close to 0, but occasionally some are large).

The used likelihood is similar to the prob-estimator, and is written as

$$L(Y, \mathbf{D}) = \sum_{i=1}^{M} (\hat{p}_m - \operatorname{tr}(A_m YY^\dagger))^2 \tag{3.13}$$

with $A_m$ the Hermitian matrix characterizing the $m^{th}$ measurement (or observable) and $\hat{p}_m \in [-1, 1]$ a measured value, which together correspond to the data $\mathbf{D}$. The authors use mixed qubit data generation, for which more details can be found in section 2.3.2.

Combining these terms, we can obtain a posterior $\hat{\nu}_{\lambda,\theta}(Y, \mathbf{D}) = \exp(-f_{\lambda,\theta}(Y, \mathbf{D}))$ with

$$f_{\lambda,\theta}(Y, \mathbf{D}) = \lambda \sum_{i=1}^{M} (\hat{p}_m - \operatorname{tr}(A_m YY^\dagger))^2 + \frac{2d + r + 2}{2} \log\det(\theta^2 I_d + YY^\dagger) \tag{3.14}$$

The exponential in this case is useful to us because it allows us to simplify the calculations. ULA requires us to set the potential $V(x) = -\log(\pi(x))$ where $\pi(x)$ is the posterior from which we want to sample, and using this formulation we obtain a log(exp) simplification.

The authors also avoid dealing with complex values by using an isomorphism $\psi : \mathbb{C}^{d\times r} \to \mathbb{R}^{2d\times 2r}$

$$\psi : M^R + iM^I \to \frac{1}{\sqrt{2}} \begin{pmatrix} M^R & -M^I \\ M^I & M^R \end{pmatrix} \tag{3.15}$$

This results in the following rewrite of $f$:

$$\tilde{f}_{\lambda,\theta}(Y,\mathbf{D}) = \lambda\sum_{i=1}^{M}(\hat{p}_m - \sqrt{2}\text{tr}(\tilde{A}_m\tilde{Y}\tilde{Y}^T))^2 + \frac{2d+r+2}{4}\log\det(\frac{\theta^2}{\sqrt{2}}I_{2d} + \sqrt{2}\tilde{Y}\tilde{Y}^T) + \tilde{C} \quad (3.16)$$

where $\tilde{Y} = \psi(Y)$, $\psi(YY^\dagger) = \sqrt{2}\psi(Y)\psi(Y)^T$ and $\tilde{C}$ a normalization constant. The posterior becomes $\hat{\mu}_{\lambda,\theta}(\tilde{Y},\mathbf{D}) = \exp(-\tilde{f}_{\lambda,\theta}(\tilde{Y},\mathbf{D}))$. We can then also obtain the gradient of $\tilde{f}_{\lambda,\theta}$:

$$\nabla\tilde{f}_{\lambda,\theta}(\tilde{Y},\mathbf{D}) = -2\sqrt{2}\lambda\sum_{i=1}^{M}(\hat{p}_m - \sqrt{2}\text{tr}(\tilde{A}_m\tilde{Y}\tilde{Y}^T))^2(\tilde{A}_m + \tilde{A}_m^{(t)})\tilde{Y} + \frac{2d+r+2}{\theta^2}\left(I_{2d} + \frac{2}{\theta^2}\tilde{Y}\tilde{Y}^T\right)^{-1}\tilde{Y} \quad (3.17)$$

They also use a trick (Sherman-Morrison-Woodbury formula) to compute the inverse more efficiently:

$$\left(I_{2d} + \frac{2}{\theta^2}\tilde{Y}\tilde{Y}^T\right)^{-1} = I_{2d} - \tilde{Y}\left(\frac{\theta^2}{2}I_{2r} + \tilde{Y}^T\tilde{Y}\right)^{-1}\tilde{Y}^T \quad (3.18)$$

Putting all these components together, we can write the pseudocode for the resulting algorithm:

**Algorithm 5:** Projected Langevin algorithm

```
   Input  : $T \in \mathbb{N}, Y^{(0)} \in \mathbb{C}^{d\times r}, \{\eta^{(k)} | k \in 1 \dots T\}, \beta \in \mathbb{R}, \theta \in \mathbb{R}, \lambda \in \mathbb{R}$
   Output: $\tilde{Y} \in \mathbb{R}^{2d\times 2r}$
1  $\tilde{Y}^{(0)} \leftarrow \psi(Y^{(0)})$
2  for $k \leftarrow 1 : T$ do
3      $w_R^{(k)}, w_I^{(k)} \sim N(0,1)^{d\times r}$ // Sample from the standard normal of size $d \times r$
4      $w^{(k)} \leftarrow w_R^{(k)} + i w_I^{(k)}$
5      $\tilde{w}^{(k)} \leftarrow \psi(w^{(k)})$
6      $\tilde{Y}^{(k)} \leftarrow \tilde{Y}^{(k-1)} - \eta^{(k)}\nabla f(\tilde{Y}^{(k-1)}, \theta, \lambda) + \frac{\sqrt{2\eta^{(k)}}}{\beta}\tilde{w}^{(k)}$
7      $\tilde{Y} \leftarrow \frac{1}{k}\tilde{Y}^{(k)} + (1 - \frac{1}{k})\tilde{Y}$
8  end
```

where $\eta^{(k)}$ is the step size, $\theta$ a scaling parameter for the prior, $\beta$ a scaling parameter for the noise term and $T$ the number of iterations. There is also a burn-in period which is not indicated here.

The main differences from the original algorithm described in 2.4.5 are that instead of doing ascent, we are doing descent, and that we sample $d \times r$ scalar entries from a standard normal distribution, instead of sampling $r$ $d$-dimensional vectors from a multivariate normal. Note that we also use a running average in this version (this was not done is the original paper), to have a fairer comparison with the prob-estimator.

The main motivation behind this approach comes from the faster convergence times, especially in high-dimensional settings where Metropolis-Hastings suffers. The proposed prior allows for a computable gradient in a reasonable time, which bootstraps the use of Langevin sampling. A second benefit comes from the Burer-Monteiro factorization, which allows to use, if available, the information about the rank of the density matrix. In general, the parameter $r$ in this factorization corresponds to an upper bound on the rank of $\hat{\rho}$, the approximation of the true density matrix $\rho$. This however also means that we may give too much freedom to the approximation if we know that the true $\rho$ is of low rank. Setting $r$ to the true rank allows us to constrain the search space.

# Chapter 4

# Numerical experiments

We will now focus on the first main contribution of this thesis, namely numerical experiments to compare the prob-estimator (which we will refer to as prob in figures) and the Projected Langevin algorithm (which we will refer to as PL or langevin in figures).

The goal of these experiments is to see how Metropolis-Hastings compares to Langevin sampling with different experimental settings in the context of Quantum Tomography. In particular, we will use the $L_2$ squared (or Frobenius squared) error as the main metric to assess performance in 5 main contexts: convergence plots in 4.2, influence of the burn-in period in 4.3, impact of the number of shots in 4.4, impact of the number of measurements in 4.5 and finally the impact of knowledge about the rank of the matrix (in particular for Langevin) in 4.6. The term "shots" here refers to the number of times the quantum system is replicated and measured. The term "measurements" (albeit somewhat ambiguous) refers to the number of observables we use.

As mentioned in the introduction 1.4, the code written for these experiments will be made available later on GitHub. All of it is in Python, with the prob-estimator reimplemented from R and the Projected Langevin algorithm from Matlab, with the original source code provided by the authors in both cases ([MA17] and [Ade+24] respectively). The prob-estimator originally uses separate qubit data generation and Projected Langevin mixed qubit, however, we will often use only one of the data generation procedures to perform the experiments (this will be indicated). Per our tests, the results are usually very close between them (for example by using separate qubit for Projected Langevin or vice versa), so either data generation approach should be valid for any experiment, regardless of the used algorithm.

Before describing in more detail each experiment, it is important to note that there are some discrepancies for the prob-estimator between the paper and the associated (original) implementation in R. These details may impact the obtained results, therefore it is relevant to keep them in mind.

- The original paper states the proposed $\tilde{V}_i$ does not depend on the previous sample ($\tilde{V}_i$ is sampled uniformly from the uniform distribution). This however contradicts their implementation, which considers $\tilde{V}_i = V_i^{(t)} + V$, where $V$ is sampled from the unit sphere. Both approaches seem valid: in the former case, while we do not sample locally, we still explore the parameter space as it is constrained to be the unit sphere;

in the latter case, we also remain on the unit sphere due to normalization. In our implementation, we decided to align on their code.

- Their paper does not mention the use of a running average, but this is a technique the authors apply in the code. In this case, we also align on their code and reflect it for Projected Langevin, to allow for a fair comparison.
- In their code, the first sample for the MCMC algorithm is not random and comes from an estimation using the inverse method. This provides a warm start and can affect the accuracy of the algorithm. In our experiments, we use the same random first sample for both algorithms to avoid any potential bias.

## 4.1 Parameter choice

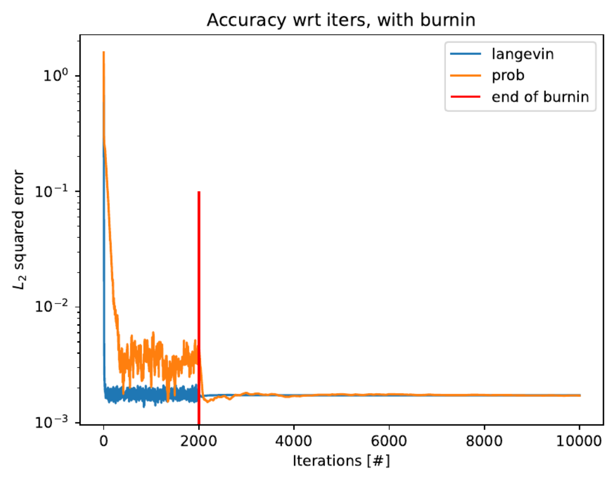
We will use $n$ to denote the number of qubits, $m$ the number of shots, $n_{\text{meas}}$ the number of measurements, $n_{\text{iter}}$ the number of iterations, $n_{\text{burnin}}$ the number of burn-in iterations, and $\text{rank}(\rho)$ the rank of the true $\rho$. If not specified otherwise, the usual setting is $n = 3$, $m = 2000$, $n_{\text{meas}} = 3^n$ for independent qubit data generation and $n_{\text{meas}} = 4^n$ for mixed qubits (complete measurement case), $n_{\text{iter}} = 10000$, $n_{\text{burnin}} = 2000$ and $\text{rank}(\rho) = 2$. The choice of a small value for $n$ is related to computation time. The matrix dimensions scale exponentially with the number of qubits, and increasing $n$ to $n = 4$ or $n = 5$ becomes expensive to run. The number of shots comes from the original source code for both the prob-estimator and Projected Langevin papers, and after tests, it seems to be informative enough (see section 4.4 for more details). The initial sample is the same for both methods and corresponds to a random matrix of rank $d = 8$. More details on how it is generated can be found in the appendix 6.

The parameters used for the prob-estimator are $\lambda = m/2$ and $\alpha = 1/2$, as it is the case in the original R code. For the Langevin sampling, we use $\lambda = m/2, \theta = 0.1, \beta = 100, \eta = 0.05/m$, where $\beta$ and $\theta$ were chosen using a grid search, $\eta$ set sufficiently small for the algorithm to converge and $\lambda$ coming from the original code.

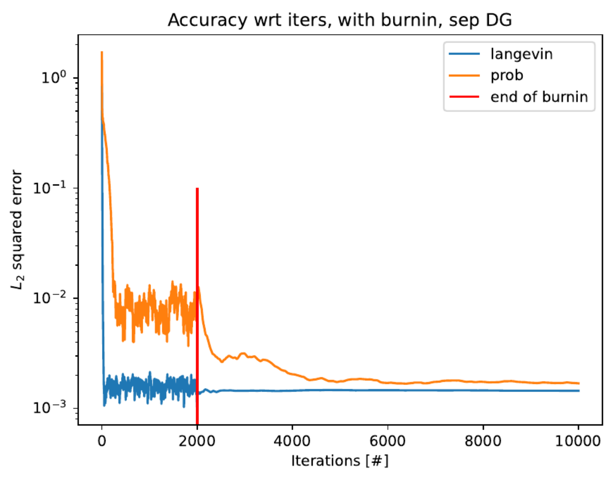
## 4.2 Convergence comparison

### Baseline convergence

We will start the comparison with a simple convergence plot:



(a) Mixed qubit



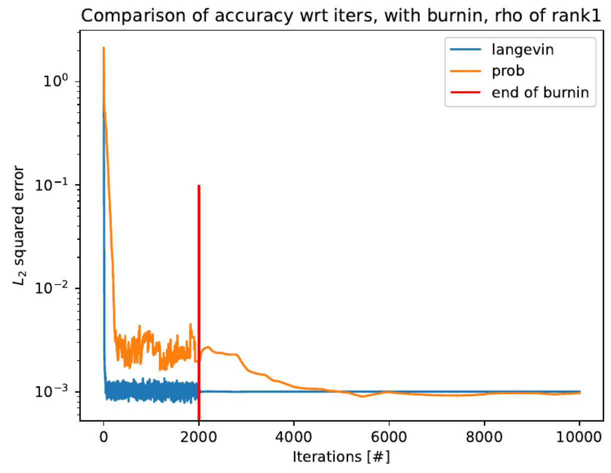
(b) Separate qubit

Figure 4.1: Convergence plot with $n_{\text{iter}} = 10000$ and $n_{\text{burnin}} = 2000$ for $n = 3$

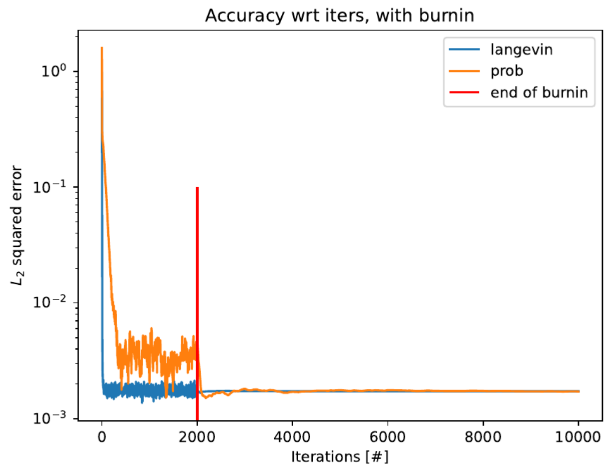
As we can see in both these plots, the 2 algorithms achieve the same accuracy, however, Projected Langevin converges much faster. This confirms the hypothesis that the gradient is beneficial. It is also a confirmation that both data generation approaches give similar results. The only difference we see is that the convergence of the prob-estimator is faster with the mixed qubit approach, however, this does not change the overall conclusion between the algorithms. It is also interesting to note how much the running average helps the prob-estimator, as without it, convergence does not seem to occur.

## Convergence across ranks

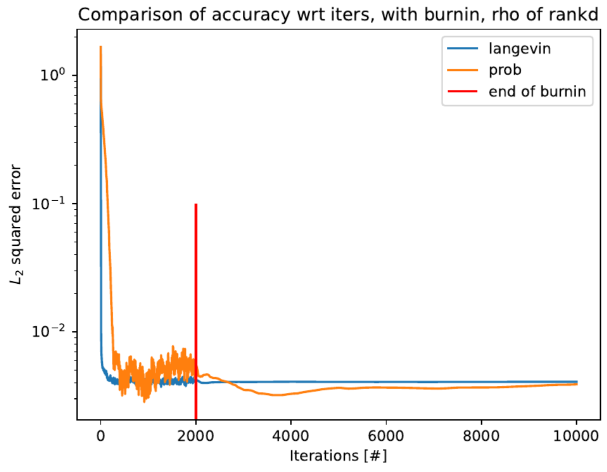
A second element that we would like to check is how these algorithms perform if the rank $\rho$ is different from 2. These are the results we get for 3 combinations, rank 1, rank 2 and rank $d = 8$ using mixed qubit data generation:



(a) $\text{rank}(\rho) = d = 1$



(b) $\text{rank}(\rho) = d = 2$



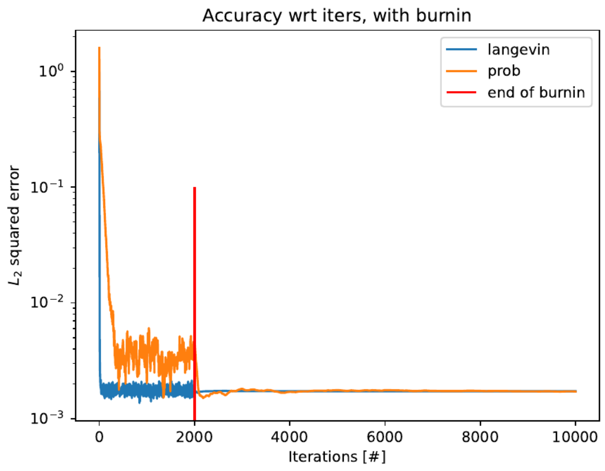
(c) $\text{rank}(\rho) = d = 8$

Figure 4.2: Convergence plot with $n_{\text{iter}} = 10000$ and $n_{\text{burnin}} = 2000$ with $\text{rank}(\rho) \in \{1, 2, d = 8\}$ for $n = 3$

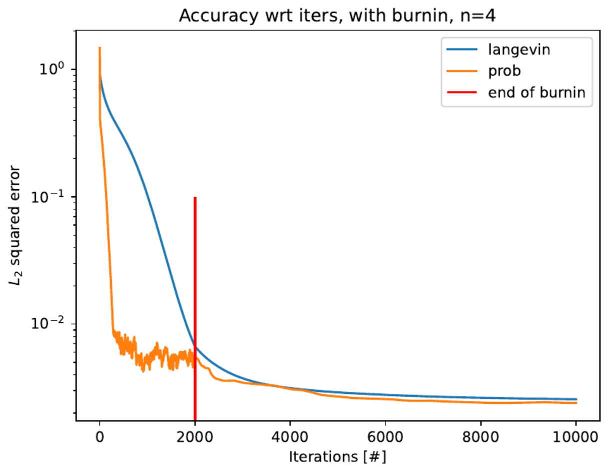
We can see that the result is sensibly the same across all ranks, hence confirming our hypothesis that rank 2 is representative of other ranks. It is interesting to note that rank 1 takes longer to converge for the prob-estimator, which means that the density matrix is more difficult to approximate in this situation. This matches our intuition, as a sum of nonzero matrix random samples (forming a Markov chain in this case) is of full rank. Note that it could also be related to the way the initial sample is generated (see the appendix 6).

## Convergence across qubit count

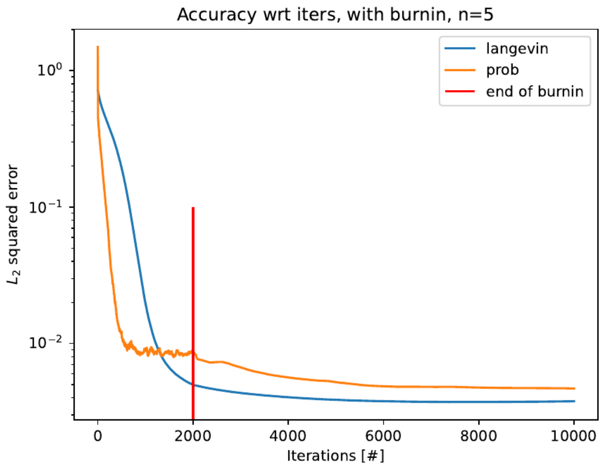
A final element that is interesting to compare is how convergence is impacted by the qubit count. In this case, we will be running both algorithms with $n \in \{3, 4, 5\}$:



(a) $n = 3$



(b) $n = 4$

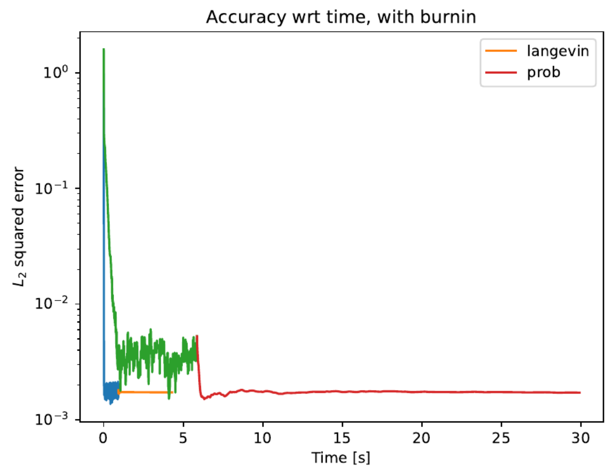


(c) $n = 5$

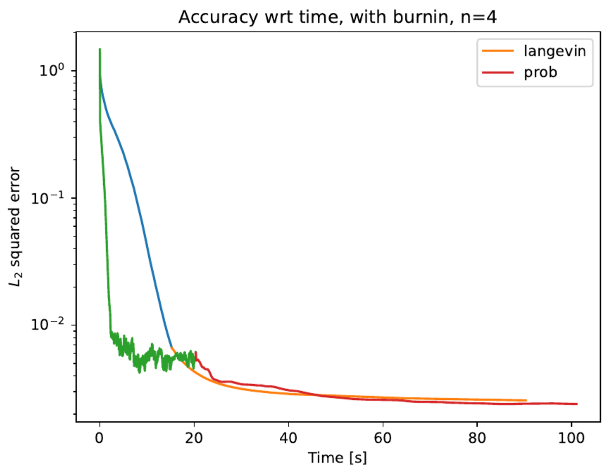
Figure 4.3: Convergence plot with $n_{\text{iter}} = 10000$ and $n_{\text{burnin}} = 2000$ with $n \in 3, 4, 5$

With each qubit increase, we can see that convergence becomes more and more challenging. We also see that for $n = 5$, the prob-estimator converges to a higher accuracy than Langevin (for this number of iterations at least). For $n \in \{4, 5\}$, we decreased $\eta$ to $0.0001/m$ as otherwise Langevin was not converging. This also explains why this curve is smooth during the burn-in period, as the steps we take are very small (on the order of 1e-7). There seems to be a major trade-off appearing for Langevin: either we reduce the step size in order to obtain convergence and accuracy, but pay with convergence time (in iterations), or converge faster, but without being as accurate, or even without obtaining convergence.

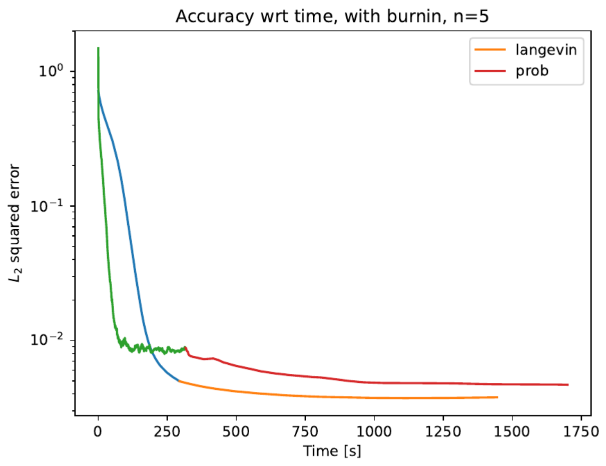
One important aspect this conclusion however omits is the computation time (in seconds). If we look at the same figures from above 4.3, but instead plot the accuracy with respect to time, this is what we obtain:



(a) $n = 3$



(b) $n = 4$



(c) $n = 5$

Figure 4.4: Convergence plot with respect to cumulative computation time with $n_{\text{iter}} = 10000$ and $n_{\text{burnin}} = 2000$ with $n \in 3, 4, 5$

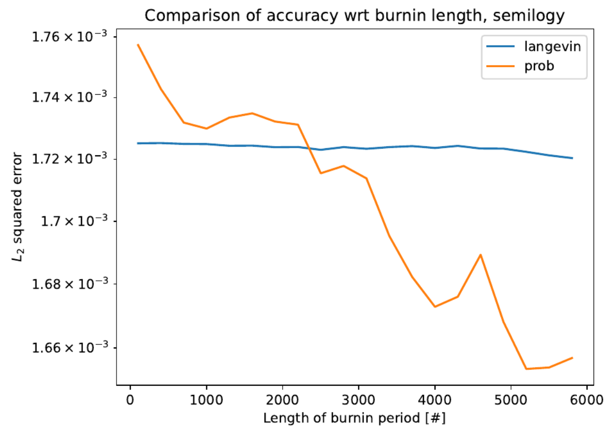
We can see that increasing $n$ and decreasing $\eta$ for Langevin leads to higher computation time for convergence. This is a surprising result, as one might expect Langevin to keep performing consistently better than Metropolis-Hastings when the dimensionality increases. It relates to the original trade-off for Langevin sampling discussed in section 2.4.5: the computation of the gradient should not come at the expense of the convergence time. Here, however, this is what happens: when we increase $n$, the gradient becomes expensive to calculate *and* the number of iterations for the algorithm to converge becomes higher, as we are required to reduce $\eta$. The main cost probably comes from the matrix inversion, although this has not been tested. Note that it is possible that with a better choice of parameters $\theta, \lambda, \beta$ the convergence time (in iterations) would decrease, however it is mostly unlikely given the extent to which it takes time and that the parameters used here were chosen through grid search. More experiments with different combinations of parameters but also larger $n$ would be needed to give a definite answer.

These results hint at the potential advantages of Langevin: it is a method that remains precise when $n$ grows if we keep $\eta$ small enough, however at the expense of computation time. The prob-estimator on the other hand seems to be consistently slow and relatively precise, but without needing any adjustments to a step size parameter. Note that it also remains untested how the same experiment would perform in case of a reduced burn-in duration for the prob-estimator, as given results from section 4.3, the final accuracy seems to be the same across varying burn-in periods. It might be possible for it to converge much faster, even with $n_{\text{burnin}} = 500$ (for $n = 3$ at least).

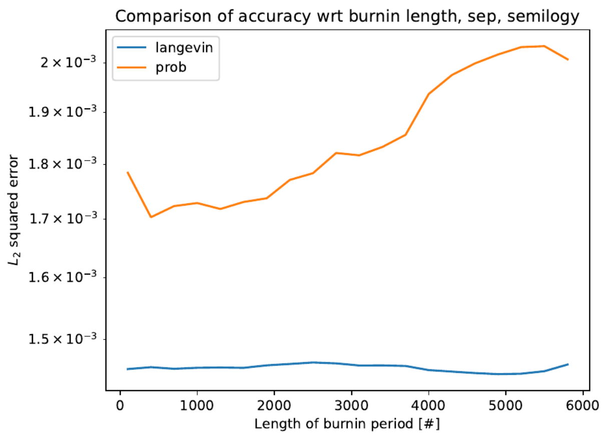
Another interesting question to ask in this context is if Langevin is suitable for larger $n$, as one might expect no convergence to happen with a very small $\eta$ (if we suppose that decreasing it will be needed to obtain convergence). This also requires more experiments to be confirmed.

## 4.3 Impact of the burn-in duration

So far, we have settled on using $n_{\text{burnin}} = 2000$ for the burn-in duration with a large number of iterations $n_{\text{iter}} = 10000$. An interesting question to ask ourselves in this situation is to which degree this value determines the convergence speed, and whether in general convergence will happen if, for example, we reduce $n_{\text{burnin}}$ . For this experiment, we fix $n_{\text{iter}} = 10000$ but we vary $n_{\text{burnin}}$ from 100 to 6000. Here are the results we obtained (with mixed qubit data generation):



(a) Mixed qubit



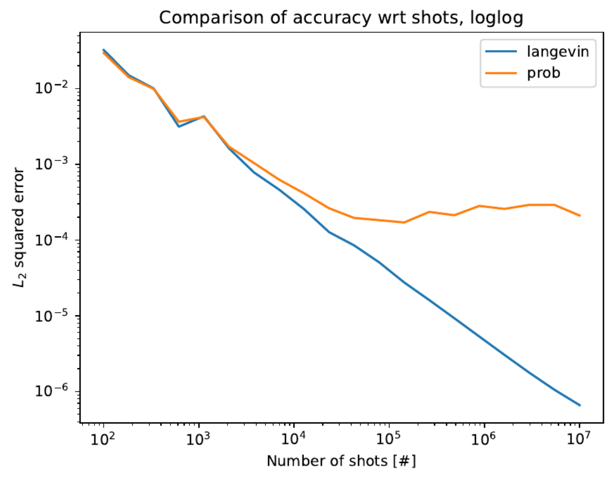
(b) Separate qubit

Figure 4.5: Impact of the iterations used for burn-in on the error with $n = 3$ and $n_{\text{iter}} = 10000$ fixed

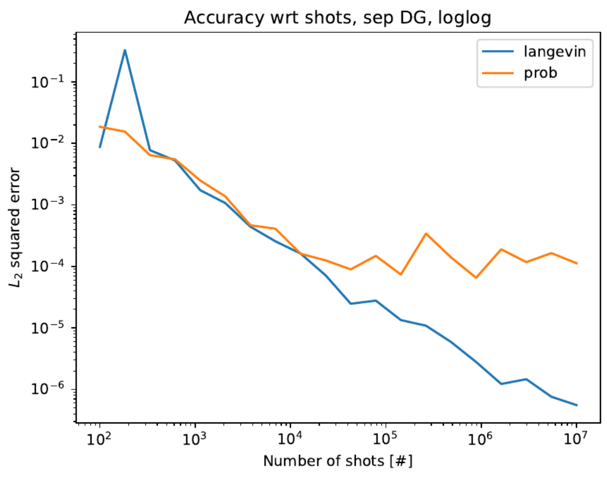
The duration of the burn-in very marginally impacts the final accuracy, as it stays on the same order of magnitude of 1e-3. There is more variation for the prob-estimator, however, this is a result we could have expected, as Projected Langevin converges very fast, while the prob-estimator takes time.

## 4.4 Impact of the number of shots

As discussed in section 4, the value of $m = 2000$ for the number of shots was chosen as it was the number provided in the original source code for both algorithms. One however might ask how this value impacts the accuracy, as the number of shots is very information-dense: performing the measurement on a system many times (or more exactly replicating the system and measuring the clones) provides better estimates for the empirical probabilities $\hat{p}_{\mathbf{a},\mathbf{s}}$. A priori, we expect that for any algorithm, increasing the number of shots will increase the accuracy. In this experiment, we will be increasing $m$ exponentially from 1e2 to 1e7, to replicate an experiment from the Langevin paper [Ade+24]. The number of iterations, as well as the burn-in duration, were reduced to $n_{\text{iter}} = 5000$ and $n_{\text{burnin}} = 1000$.



(a) Mixed qubit



(b) Separate qubit

Figure 4.6: Impact of the number of used shots on the error with $n = 3$

We can see quite a surprising result: while Langevin gets the expected behaviour of scaling linearly in the log-log plot, the prob-estimator does not improve beyond a certain threshold of approximately 1e-4/1e-5 shots. It occurs with both data generation processes, which means that it is not the root of the difference. A possible explanation could be that the $\lambda$ parameter, which depends on the number of shots, makes the likelihood term become very large, with the prior stopping being significant. We have therefore tried changing the value of $\lambda$ to a fixed value across all shots' counts (only for the prob-estimator). This is what we obtained with mixed qubit data generation:

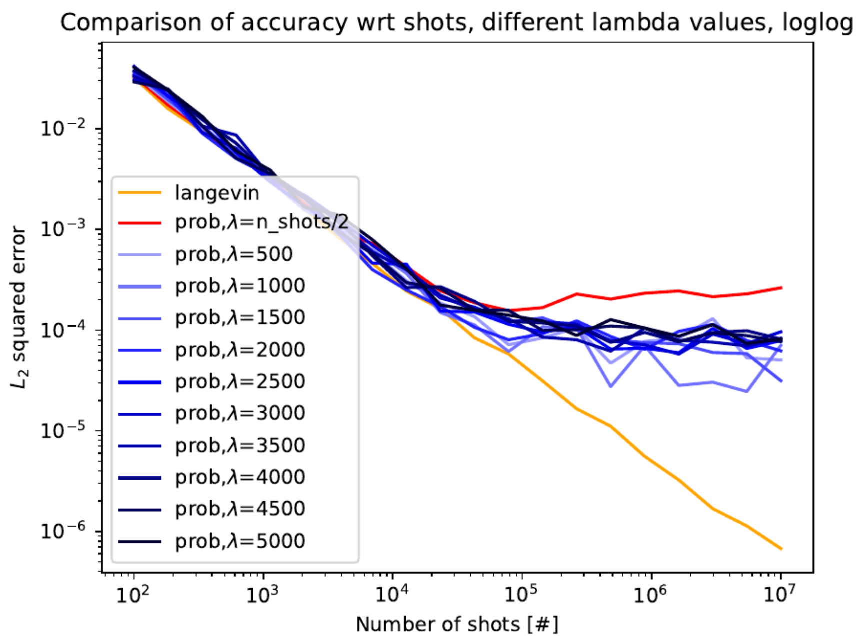
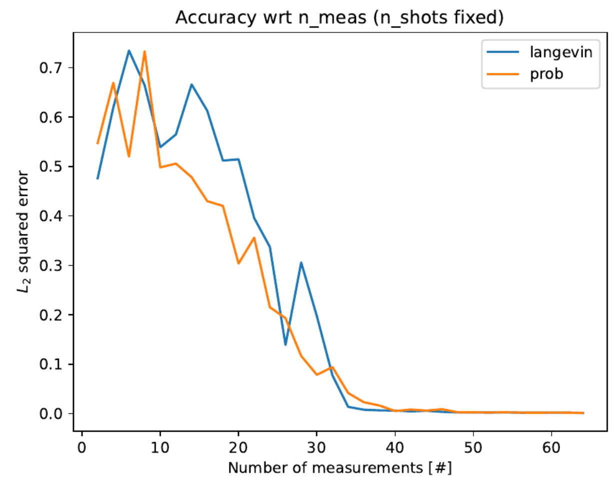


Figure 4.7: Impact of the number of used shots on the error with varying parameter $\lambda$ for the prob-estimator with $n = 3$

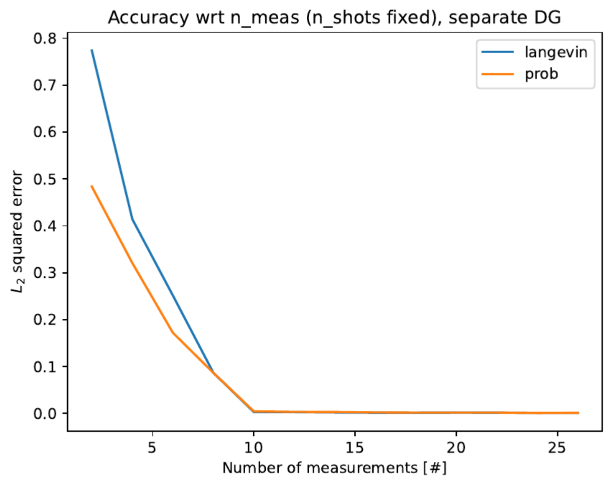
We can see that even with a fixed $\lambda$ for the prob-estimator, the accuracy stagnates at around 1e-4, even for large $m$. We are not exactly sure why this occurs, as we are able to replicate the same behaviour with the original R code (the authors also do not provide metrics beyond $m = 2000$ in their paper). The only conclusion we can draw from this result is that it is an inherent property of the method.

## 4.5 Impact of the number of measurements

The number of measurements, or observables, is another interesting aspect to explore. While in general, we assume a complete measurement experiment, here we will try to reduce the number of observables to see how both algorithms behave in this setup, and how many observables are needed in order to keep accuracy sufficiently high. Note that inferring the state based on a subset of all measurements is analogous to solving an underdetermined system of equations, and can be seen as compressed sensing. We will be increasing $n_{\text{meas}}$ from 1 to either $4^n$ for mixed qubit, or $3^n$ for separate qubit data generation, with the observables randomly chosen at each step. The number of iterations, as well as the burn-in duration, were reduced to $n_{\text{iter}} = 5000$ and $n_{\text{burnin}} = 1000$. Here is what we obtain:



(a) Mixed qubit



(b) Separate qubit

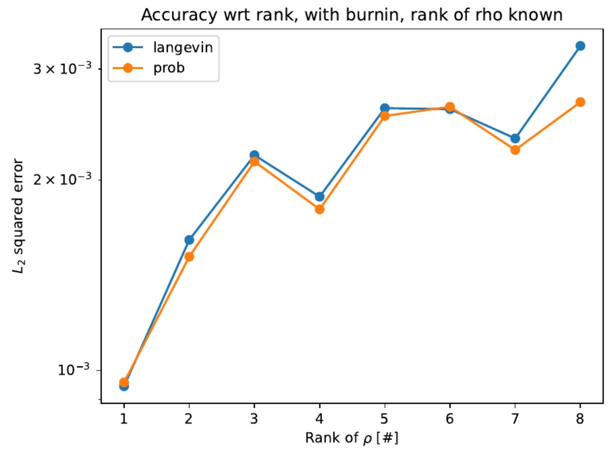
Figure 4.8: Impact of the number of used measurements on the error with $n = 3$

We can see that both algorithms perform similarly with both data generation schemas: for mixed qubit, approximately half of the total observables (around 35) are needed to obtain most of the information; for separate qubit, this value seems to be around 10, closer to a third of the total count.

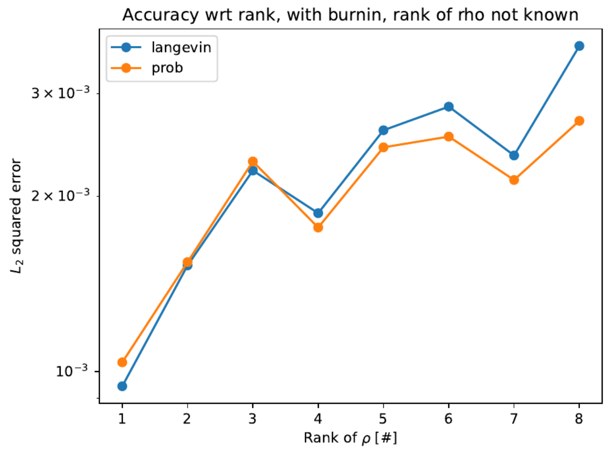
This is not an unexpected result, an approximate approach comes with the downside of being a bit less precise or conveying less information. An interesting plot to obtain related to this one would be the cumulative information gain, which could allow us to answer questions such as "How many observables do we need in order to be x% accurate as the equilibrium distribution?" or "How many observables do we need in order to have x% of information of the equilibrium distribution?".

## 4.6 Impact of knowing the rank

An advantage that the Projected Langevin algorithm has is that it can also incorporate prior information through the Burer-Monteiro factorization, and in particular the rank of the factors. If we know the rank of our true matrix, we can constrain the search space by setting the upper bound of the rank of the sampled matrix $Y$ to be the one of our true matrix $\rho$. In this experiment, we will be comparing the situation where we have information about the rank, in which case the initial sample for Langevin will be of rank of $\rho$, and otherwise, we will keep it fixed as $d = 8$. For both cases, we will increase rank($\rho$) from 1 to $d = 8$ to see how accuracy changes, and use mixed qubit data generation. Here are the results:



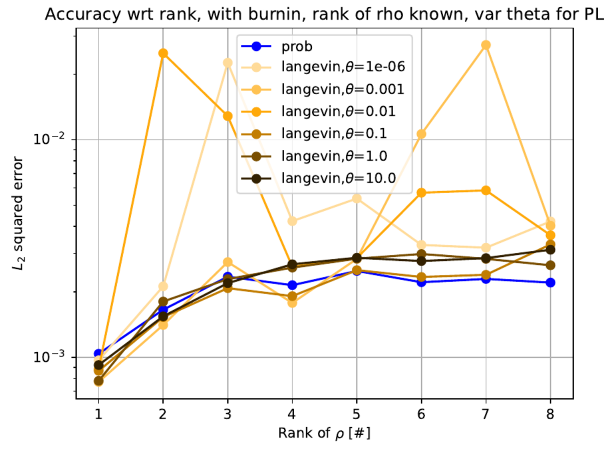
(a) Rank of $\rho$ known



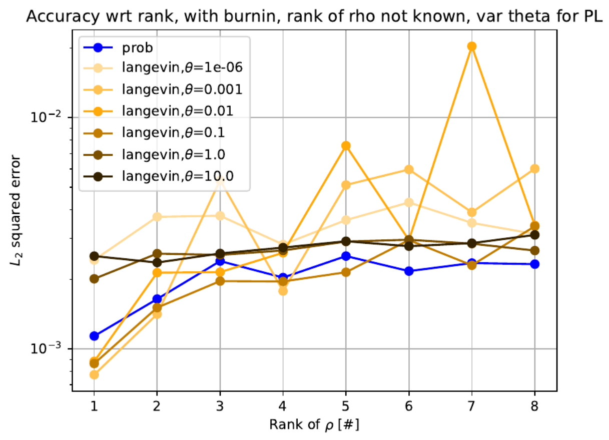
(b) Rank of $\rho$ not known

Figure 4.9: Rank knowledge plot for $n = 3$

One might also wonder how the parameter $\theta$ in the prior affects this result. In figure 4.10, we ran the same experiment with $\theta \in \{1\text{e-}6, 1\text{e-}3, 1\text{e-}2, 1\text{e-}1, 1\text{e}0, 1\text{e}1\}$:



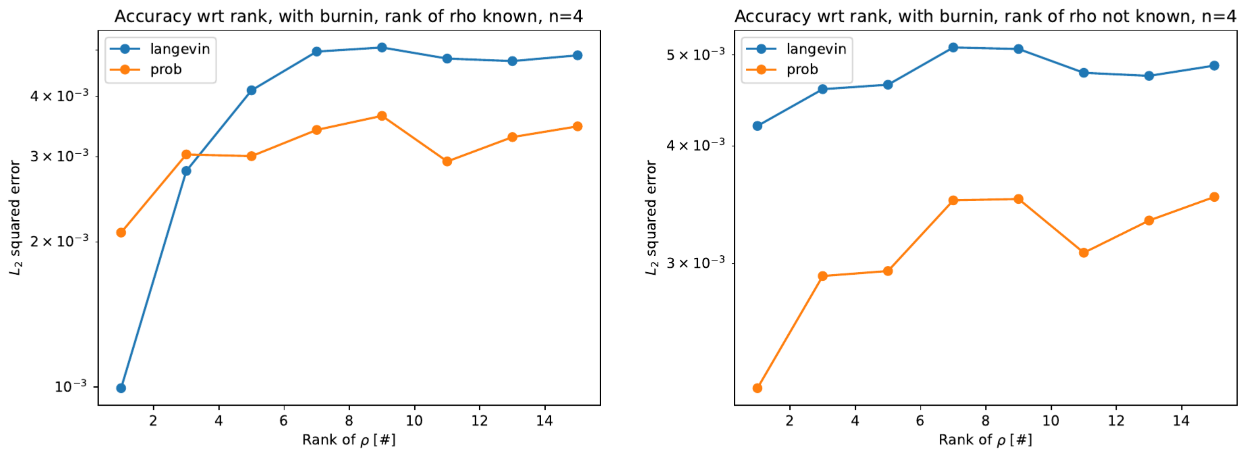
(a) Rank of $\rho$ known



(b) Rank of $\rho$ not known

Figure 4.10: Rank knowledge plot for $n = 3$ and different combinations of $\theta$

We can also try this experiment for $n = 4$ (hence from rank$(\rho) = 1$ to rank$(\rho) = d = 16$), to see if it generalizes to higher qubit counts:



(a) Rank of $\rho$ known

(b) Rank of $\rho$ not known

Figure 4.11: Rank knowledge plot for $n = 4$

These results are surprising, as they show that the information about the rank only leads to a marginal improvement in accuracy (it stays on the same order of magnitude). The most insightful plot is figure 4.10 with the different combinations of $\theta$: we see on the left figure 4.10a, where we know the rank, that the error increases for rank$(\rho) \in \{1, 2, 3\}$ and stabilizes for other ranks around 2e-3/3e-3 (with more variance for smaller $\theta$). On the right figure 4.10b, some parameters have the same curve, but others, in particular, $\theta \in \{1, 10\}$, have a flat curve, with little variation between ranks. The overall conclusion remains however the same regardless of the parameter value: the information gain is minimal and the error is of the same order of magnitude. These results seem to generalize for higher $n$, as we see on figure 4.11.

There are 2 possible conclusions we can draw from this: either the upper bound that the factorization brings plays a small role in how it constrains the search space (with the prior and its parameters having much more impact), however, it is also possible that the way we obtain our first sample is not correct, and even though samples are numerically of different rank, they tend to be too similar (see the appendix 6 for more details on how the first sample is generated). More experiments are needed to investigate these results.

Another interesting insight we can draw from figures 4.9, 4.10 and 4.11 (for both cases, whether we know the rank of $\rho$ or not) is that approximating matrices of high rank tends to be more difficult. It is not clear if this is only linked to the fact that we use a low-rank prior, or simply a property of these matrices. Future work could explore non-low-rank priors in the experiments we propose, which would allow us to estimate the usefulness of low-rank priors.

## 4.7 Summary

The previous experiments allow us to conclude several elements about the difference between the prob-estimator and Projected Langevin.

First, we can indeed confirm the hypothesis that Langevin converges faster for $n = 3$ thanks to the gradient information based on the results we saw in section 4.2. This algorithm also seems to be more precise for larger $n$, as shown in figure 4.3c. This algorithm however does not seem to scale when the dimensionality increases, as the computational time becomes nearly equivalent to that of the prob-estimator, due to the small $\eta$ needed in order to converge and the high cost of a gradient calculation. This asks the question whether it makes sense to use Langevin in practical applications where $n$ is large. These conclusions seem to be consistent across the data generation processes, as we saw in figure 4.1, as well as ranks (figure 4.2).

Second, for both algorithms, any burn-in duration produces the same accuracy for $n = 3$, assuming the total iteration count is sufficiently large ($n_{\text{iter}} = 10000$ in this case). As we see in figure 4.5, from $n_{\text{burnin}} = 100$ to $n_{\text{burnin}} = 6000$, the curves stays close to flat.

Third, Projected Langevin scales when the number of shots increases, while the prob-estimator does not beyond a threshold. As we saw in section 4.4, the prob-estimator does not get more accurate after approximately 1e-4/1e-5 shots, while Langevin has a straight line in the log-log plot of figure 4.6. This is further confirmed in figure 4.7, where $\lambda$ is fixed (not depending on $m$) and tested with different values for the prob-estimator. This means that the prob-estimator is not suitable for obtaining high precision estimates of $\rho$ in situations when a large number of shots is available.

Fourth, the number of measurements or observables needed in order to stay accurate is the same for both Projected Langevin and the prob-estimator. Indeed, the data generation process has much more importance, and we observe mixed qubit needing on average half of the total number, while separate qubit only a third, as visible in figure 4.8. This confirms our intuition that an exact method needs fewer observables than an approximate method.

Finally, for Projected Langevin, knowing the true rank of $\rho$ only leads to marginally better accuracy. Indeed, figures 4.9, 4.10 and 4.11 all result in the same conclusion: the rank of $\rho$ has more importance on the accuracy than having information about it.

# Chapter 5

# Algorithm vs prior: what matters the most?

The previous experiments in Chapter 4 showed us that for a small number of qubits $n$, Projected Langevin tends to converge faster in terms of iterations. This raises the question of "why", as [Ade+24] brings not only a new algorithm, but also a new prior. In this chapter, we will introduce 2 new algorithms, Metropolis-Hastings with Student-t prior (MHS) and Metropolis-Hastings with Gibbs with Student-t prior (MHGS) in order to try to understand which part contributes most. By combining Metropolis-Hastings from the prob-estimator with the Student-t prior from Projected Langevin, we can estimate whether the prior is a silver bullet that can solve Bayesian Quantum Tomography, or if Langevin is still a mandatory component for it to work.

This chapter is the second main contribution of this thesis and is structured as follows: in 5.1 we will describe in detail both algorithms, then in 5.2 we will talk about the choice of the proposal and finally in 5.3 we will numerically evaluate how these algorithms perform.

## 5.1 Introducing new algorithms: MHS and MHGS

### Metropolis-Hastings with Student-t prior

*Metropolis-Hastings with Student-t* (MHS), as its name suggests, is a Metropolis-Hastings-based algorithm. It is a vanilla version of the original algorithm, as the modifications are minor. The main difference comes from the fact that we work directly on the matrix space, rather than vectors or scalars. More precisely, we utilize the Burer-Monteiro factorization from the Projected Langevin approach (as seen in section 3.2) to work directly with $Y \in \mathbb{R}^{2d\times 2r}$, where we do the reverse isomorphic transformation $\psi^{-1}$ to map back to $Y \in \mathbb{C}^{d\times r}$ to calculate $\hat{\rho}$. We also use the spectral scaled Student-t prior from Langevin (defined on $\mathbb{C}$ here):

$$\nu_\theta(Y) = C_\theta \det(\theta^2 I_d + YY^\dagger)^{-(2d+r+2)/2} \tag{5.1}$$

When converted to the real domain, this results in the same posterior as in Projected

Langevin $\hat{\mu}_{\lambda,\theta}(\tilde{Y}, \mathbf{D}) = \exp(-\tilde{f}_{\lambda,\theta}(\tilde{Y}, \mathbf{D}))$, where

$$\tilde{f}_{\lambda,\theta}(Y, \mathbf{D}) = \lambda \sum_{i=1}^{M} (\hat{p}_m - \sqrt{2}\mathrm{tr}(\tilde{A}_m \tilde{Y}\tilde{Y}^T))^2 + \frac{2d+r+2}{4} \log\det(\frac{\theta^2}{\sqrt{2}} I_{2d} + \sqrt{2}\tilde{Y}\tilde{Y}^T) + \tilde{C} \quad (5.2)$$

In practice, we also use the log-transformation trick as described in section 2.4.3 because it allows for better numerical stability. Assuming a proposal $p_1(\tilde{Y}|\tilde{Y}^{(k-1)})$ (which we will describe in more detail in the next section 5.2) and acceptance rate $A_1$, the pseudocode for the resulting algorithm is the following:

**Algorithm 6:** Metropolis-Hastings with Student-t prior

```
   Input  : $T \in \mathbb{N}, Y^{(0)} \in \mathbb{C}^{d \times r}, \theta \in \mathbb{R}, \lambda \in \mathbb{R}$
   Output: $\tilde{Y} \in \mathbb{R}^{2d \times 2r}$
1  $\tilde{Y}^{(0)} \leftarrow \psi(Y^{(0)})$
2  $\tilde{Y} = \mathbf{0}$
3  for $k \leftarrow 1 : T$ do
4      Sample $\tilde{Y}^* \sim p_1(\tilde{Y}|\tilde{Y}^{(k-1)})$
5      Sample $u \sim U(0,1)$
6      $\alpha \leftarrow \min\left\{\log A_1(\tilde{Y}^*, \tilde{Y}^{(k-1)}, \theta, \lambda), \log(1)\right\}$
7      if $\log(u) \leq \alpha$ then
           // Accept $\tilde{Y}^*$
8          $\tilde{Y}^{(k)} \leftarrow \tilde{Y}^*$
9      else
           // Reject $\tilde{Y}^*$
10         $\tilde{Y}^{(k)} \leftarrow \tilde{Y}^{(k-1)}$
11     end
12     $\tilde{Y} \leftarrow \frac{1}{k}\tilde{Y}^{(k)} + (1 - \frac{1}{k})\tilde{Y}$
13 end
```

where $\lambda$ is a weight parameter for the likelihood, $\theta$ a scale parameter for the prior, $T$ the number of iterations and $Y^{(0)}$ the initial sample.

## Metropolis-Hastings with Gibbs with Student-t prior

*Metropolis-Hastings with Gibbs with Student-t prior* (MHGS) is an extension of MHS and is an attempt to imitate more closely the way the prob-estimator works. Indeed as we saw in section 3.1, the prob-estimator resembles more Metropolis-within-Gibbs than pure Metropolis-Hastings, as it iterates over each dimension for each Metropolis-Hastings iteration. With this in mind, MHGS works by sampling each entry of the matrix $\tilde{Y} \in \mathbb{R}^{2d \times 2r}$ separately. If we assume the proposal $p_2(y|\tilde{Y}_{ij}^{(k-1)})$ and acceptance rate $A_2$, this results in the following algorithm:

**Algorithm 7:** Metropolis-Hastings with Gibbs with Student-t prior

```
Input  : $T \in \mathbb{N}, Y^{(0)} \in \mathbb{C}^{d\times r}, \theta \in \mathbb{R}, \lambda \in \mathbb{R}$
Output: $\tilde{Y} \in \mathbb{R}^{2d\times 2r}$
1  $\tilde{Y}^{(0)} \leftarrow \psi(Y^{(0)})$
2  $\tilde{Y} = \mathbf{0}$
3  for $k \leftarrow 1 : T$ do
4      $\tilde{Y}^{(k)} \leftarrow \tilde{Y}^{(k-1)}$
5      for $i \leftarrow 1 : 2d$ do
6          for $j \leftarrow 1 : 2r$ do
7              Sample $y^* \sim p_2(y|\tilde{Y}_{ij}^{(k)})$
8              Sample $u \sim U(0,1)$
9              $\alpha \leftarrow \min\left\{\log A_2(y^*, \tilde{Y}_{ij}^{(k)}, \theta, \lambda), \log(1)\right\}$
10             if $\log(u) \leq \alpha$ then
                   // Accept $y^*$
11                 $\tilde{Y}_{ij}^{(k)} \leftarrow y^*$
12             else
                   // Reject $y^*$
13                 $\tilde{Y}_{ij}^{(k)} \leftarrow \tilde{Y}_{ij}^{(k)}$
14             end
15         end
16     end
17     $\tilde{Y} \leftarrow \frac{1}{k}\tilde{Y}^{(k)} + (1 - \frac{1}{k})\tilde{Y}$
18 end
```

where $\lambda$ is a weight parameter for the likelihood, $\theta$ a scale parameter for the prior, $T$ the number of iterations and $Y^{(0)}$ the initial sample.

## 5.2 Proposal choice

A very important element of these approaches is the choice of the proposal. As we are trying to be as close as possible to the prob-estimator for the algorithm part, it is not clear which proposal should be chosen. The original algorithm samples from a prior, be it a Gamma distribution for $\gamma$ or the unit sphere for $V$. Sampling from these distributions is, however, only relevant in the rank-1 factorization that the authors use, and not as a general proposal. Following this logic, we should be sampling from the scaled Student-t distribution, however, it is not obvious how this should be done, with further work required to investigate it. In the following sections, we will discuss the considered options.

### Proposal for MHS

For the MHS algorithm, we considered 3 main options for $p_1(\tilde{Y}|\tilde{Y}^{(k-1)})$: the normal distribution centered at the current sample (which is a very common proposal), the normal distribution centered at 0 (independent normal) and the Gamma distribution, in the same way that is done by the prob-estimator. This last method is not the most appropriate given what we mentioned above, however, we kept it as a point of comparison. We used the adjusted version of the acceptance rate for both the independent normal and the centered normal (note that the proposal component in the numerator and denominator simplify for it, as it is symmetrical), but did not do so for the gamma distribution, as it is not clear how this should be done (even though it is asymmetrical). We summarize them below in 5.1:

| Proposal name | Equation |
|---|---|
| Normal distribution, centered | $\tilde{Y}^* \leftarrow \tilde{Y}^{(k-1)} + \sigma\boldsymbol{\eta} \quad \text{with } \boldsymbol{\eta} \sim N(0,1)^{2d\times 2r}$ |
| Normal distribution, independent | $\tilde{Y}^* \leftarrow \sigma\boldsymbol{\eta} \quad \text{with } \boldsymbol{\eta} \sim N(0,1)^{2d\times 2r}$ |
| Gamma distribution, centered | $\tilde{Y}^* \leftarrow \tilde{Y}^{(k-1)} \odot \exp(\sigma\boldsymbol{\xi}) \quad \text{with } \boldsymbol{\xi} \sim U(-0.5,0.5)^{2d\times 2r}$ |

Table 5.1: Proposals considered for MHS

After running a grid search with parameters $\sigma \in \{0.001, 0.01, 0.05, 0.1, 0.5, 1, 10\}$ and $n_{\text{iter}} = 5000$, $n_{\text{burnin}} = 1000$ these are the results we obtain:

| Proposal name | Parameter value | Error ($L_2^2$) |
|---|---|---|
| Centered normal | 0.001 | 0.093 |
| Centered normal | 0.01 | 0.115 |
| Independent normal | 0.05 | 0.316 |
| Gamma | 0.1 | 0.423 |
| Gamma | 1.0 | 0.449 |

Table 5.2: Grid search results over proposals and their scaling parameters for MHS, top 5 entries sorted by error

Given this outcome, we settled on using the centered normal as our proposal with $\sigma = 0.001$. Note that this choice of proposal is not the best, and as stated previously, we should ideally be sampling from the Student-t distribution to align with the prob-estimator approach.

## Proposal for MHGS

For the MHGS algorithm, we decided to remove the independent normal distribution as we deemed it to be less relevant, however, kept the two other methods in their scalar version:

| Proposal name | Equation |
|---|---|
| Normal distribution, centered | $y^* \leftarrow \tilde{Y}_{ij}^{(k)} + \sigma\eta \quad \text{with } \eta \sim N(0,1)$ |
| Gamma distribution, centered | $y^* \leftarrow \tilde{Y}^{(k)} \exp(\sigma u) \quad \text{with } u \sim U(-0.5,0.5)$ |

Table 5.3: Proposals considered for MHGS

We then re-ran the same grid search with $n_{\text{iter}} = 1000$ and $n_{\text{burnin}} = 300$ to obtain the following results:

| Proposal name | Parameter value | Error ($L_2^2$) |
|---|---|---|
| Centered normal | 0.1 | 0.052 |
| Centered normal | 0.05 | 0.063 |
| Centered normal | 0.001 | 0.097 |
| Centered normal | 10.0 | 0.165 |
| Centered normal | 1.0 | 0.166 |

Table 5.4: Grid search results over proposals and their scaling parameters for MHGS, top 5 entries sorted by error

As we can see, the centered normal is once again the best result (in this case with standard deviation $\sigma = 0.1$), which confirms that is it a good default choice for proposals in MCMC.

## 5.3 Comparison with the prob-estimator and the Projected Langevin algorithm

With the proposals being chosen, we can now proceed to compare these 2 new algorithms to the prob-estimator and Projected Langevin. We will be using the convergence plot to assess the performance, using the separate qubit data generation approach in order to avoid any potential bias that might arise with mixed qubit. The parameter values were kept the same as those from the Langevin algorithm, with $\theta = 0.1, \lambda = m/2$. Here is what we obtain:

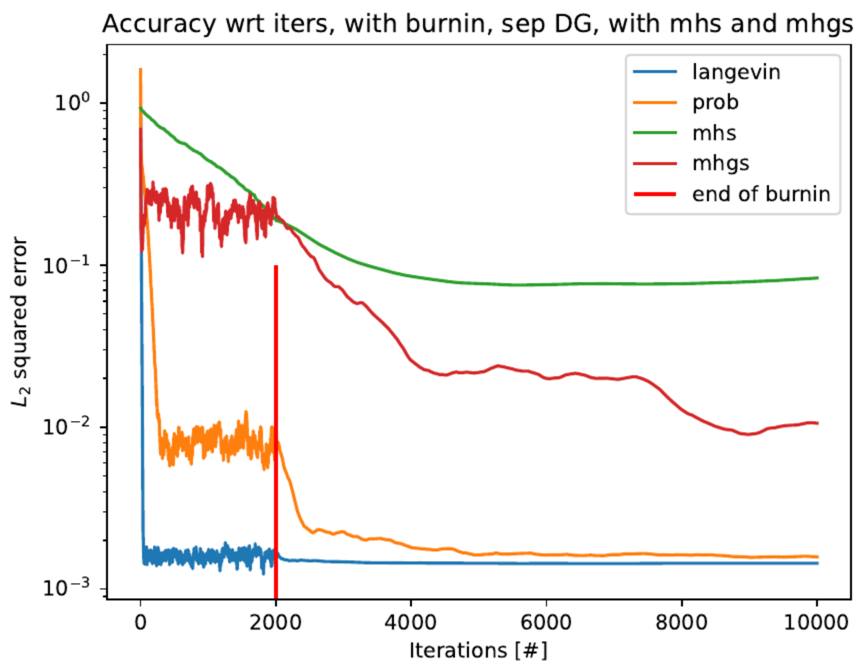


Figure 5.1: Convergence plot with the prob-estimator, Projected Langevin, MHS, and MHGS with $n = 3$ and separate qubit data generation

As we can see, both new methods perform worse than the prob-estimator and Projected Langevin, even with a sufficiently good proposal. The MHGS algorithm seems to perform better than MHS, which validates the idea behind the prob-estimator, and other techniques such as [Mai22; Luk+20], which all sample one-dimensional distributions across dimensions instead of directly sampling from the joint. This however comes at the expense of very long computation times, something that was also visible in section 4.2, figure 4.4a.

The major conclusion to draw however from this result is that the prior is not a solution on its own, as pairing it with another algorithm does not lead to better or similar accuracy. The similar results obtained from two different algorithms indicate that it is unlikely to be a coincidence. This of course makes the assumption that the proposal we use is sufficiently good and does not perform suboptimally. This also implies that if a new method is to be implemented using this prior, it is important, after choosing the algorithm well and tuning its hyperparameters, to thoroughly benchmark its performance in order to make sure it is sufficiently accurate. The prior itself does not provide any guarantees of accuracy.

The prior also does not help with the convergence speed, as we can see MHGS taking a very long time to converge (and possibly not even converging). This means that Langevin sampling is needed in the context of our comparison. Further work is required to verify if this property comes specifically from Langevin or the gradient information in general, in which case methods such as Hamiltonian Monte Carlo may also be valid.

# Chapter 6

# Conclusion and future work

We have in this master thesis explored how Markov chain Monte Carlo methods can help us solve the Quantum Tomography problem. By leveraging the Bayesian framework, we can include the properties of a density matrix into the prior and approximate the posterior by sampling from it.

In particular, we have numerically compared the prob-estimator introduced in [MA17] and the Projected Langevin algorithm introduced in [Ade+24] in different experimental settings. Our results in Chapter 4 demonstrate that

- The Projected Langevin algorithm converges faster than the prob-estimator for $n = 3$, however, the computational time becomes a bottleneck with a larger number of qubits. This is due to a need to reduce the stepsize $\eta$ for the algorithm to converge to the equilibrium distribution, and a high cost for the gradient calculation. The Langevin algorithm nevertheless remains more precise for large $n$, making it likely a better choice.
- The prob-estimator is not appropriate in situations when a large number of shots is done, for example in order to estimate $\rho$ very precisely. This is due to its bad scaling for large $m$, as demonstrated in figure 4.6. In these situations, the Projected Langevin method seems better suited.
- The number of measurements or observables needed in order to estimate $\rho$ to a certain degree of accuracy does not depend on the algorithm, but rather on the data generation process, as seen in section 4.5.
- For Projected Langevin, knowing the true rank of $\rho$ only leads to marginally better results, as demonstrated in section 4.6.

By then introducing 2 new algorithms in section 5.1, we wanted to infer what part of the Projected Langevin method, the algorithm or prior, made it perform better. The results conclude that it is probably a combination of both, and the novel low-rank prior by itself is not a silver bullet for Bayesian Quantum Tomography.

This thesis however only explores a subset of the possible experiments, and many questions remain to be answered. We have only compared the prob-estimator and the Projected Langevin algorithm, but as described in the introduction 1.3, many other algorithms exist for Bayesian Quantum Tomography. Creating a comprehensive overview of methods with their characteristics would allow researchers to make an informed choice for their use case. It would also be interesting to expand all the experiments that we have done to larger $n$ and test all ranks of $\rho$.

We have also talked about convergence in a nonrigorous way, without relying on the convergence bounds that most previous work comes with (including the reference papers). It would be interesting to quantify and compare the obtained numerical results and the expected theoretical bounds.

In Chapter 5 we tried to give an answer on why Projected Langevin performs better by proposing 2 new algorithms. The set of experiments we used to draw our conclusion was however very limited and expanding it to the full suite of chapter 4 would allow to provide more nuance to the answer. The proposed algorithms also form only a small subset of possible priors and algorithms. Exploring other low-rank priors (but also non-low-rank) and algorithms, such as Hamiltonian Monte Carlo for gradient-based methods or preconditioned Crank-Nicholson for non-gradient based, would greatly benefit our understanding of the correctness and robustness of our conclusions.

# Bibliography

[VR89] K. Vogel and H. Risken. “Determination of quasiprobability distributions in terms of probability distributions for the rotated quadrature phase”. In: *Phys. Rev. A* 40 (5 Sept. 1989), pp. 2847–2849. DOI: 10.1103/PhysRevA.40.2847. URL: https://link.aps.org/doi/10.1103/PhysRevA.40.2847.

[Ban+99] K. Banaszek et al. “Maximum-likelihood estimation of the density matrix”. In: *Phys. Rev. A* 61 (1 Dec. 1999), p. 010304. DOI: 10.1103/PhysRevA.61.010304. URL: https://link.aps.org/doi/10.1103/PhysRevA.61.010304.

[Jam+01] Daniel F. V. James et al. “Measurement of qubits”. In: *Phys. Rev. A* 64 (5 Oct. 2001), p. 052312. DOI: 10.1103/PhysRevA.64.052312. URL: https://link.aps.org/doi/10.1103/PhysRevA.64.052312.

[BM03] Samuel Burer and Renato D.C. Monteiro. “A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization”. In: *Mathematical Programming* 95.2 (Feb. 2003), pp. 329–357. ISSN: 1436-4646. DOI: 10.1007/s10107-002-0352-8. URL: https://doi.org/10.1007/s10107-002-0352-8.

[Hra+04] Zdeněk Hradil et al. “3 Maximum-Likelihood Methodsin Quantum Mechanics”. In: *Quantum State Estimation*. Ed. by Matteo Paris and Jaroslav Řeháček. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 59–112.

[Lvo04] A I Lvovsky. “Iterative maximum-likelihood reconstruction in quantum homodyne tomography”. In: *Journal of Optics B: Quantum and Semiclassical Optics* 6.6 (May 2004), S556. DOI: 10.1088/1464-4266/6/6/014. URL: https://dx.doi.org/10.1088/1464-4266/6/6/014.

[RC04] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods*. Springer, 2004. ISBN: 978-1-4757-4145-2. URL: https://link.springer.com/book/10.1007/978-1-4757-4145-2.

[Blu10a] Robin Blume-Kohout. “Hedged Maximum Likelihood Quantum State Estimation”. In: *Phys. Rev. Lett.* 105 (20 Nov. 2010), p. 200504. DOI: 10.1103/PhysRevLett.105.200504. URL: https://link.aps.org/doi/10.1103/PhysRevLett.105.200504.

[Blu10b] Robin Blume-Kohout. “Optimal, reliable estimation of quantum states”. In: *New Journal of Physics* 12.4 (Apr. 2010), p. 043034. ISSN: 1367-2630. DOI: 10.1088/1367-2630/12/4/043034. URL: http://dx.doi.org/10.1088/1367-2630/12/4/043034.

[Gro+10] David Gross et al. “Quantum State Tomography via Compressed Sensing”. In: *Phys. Rev. Lett.* 105 (15 Oct. 2010), p. 150401. DOI: 10.1103/PhysRevLett.105.150401. URL: https://link.aps.org/doi/10.1103/PhysRevLett.105.150401.

[NC10] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[RMH10] J. Řehá ček, D. Mogilevtsev, and Z. Hradil. “Operational Tomography: Fitting of Data Patterns”. In: *Phys. Rev. Lett.* 105 (1 June 2010), p. 010402. DOI: 10.1103/PhysRevLett.105.010402. URL: https://link.aps.org/doi/10.1103/PhysRevLett.105.010402.

[Gro11] David Gross. “Recovering Low-Rank Matrices From Few Coefficients in Any Basis”. In: *IEEE Transactions on Information Theory* 57.3 (Mar. 2011), pp. 1548–1566. ISSN: 1557-9654. DOI: 10.1109/tit.2011.2104999. URL: http://dx.doi.org/10.1109/TIT.2011.2104999.

[Kol11] Vladimir Koltchinskii. “Von Neumann entropy penalization and low-rank matrix estimation”. In: *The Annals of Statistics* 39.6 (2011), pp. 2936–2973. DOI: 10.1214/11-AOS926. URL: https://doi.org/10.1214/11-AOS926.

[Fla+12] Steven T Flammia et al. “Quantum tomography via compressed sensing: error bounds, sample complexity and efficient estimators”. In: *New Journal of Physics* 14.9 (Sept. 2012), p. 095022. DOI: 10.1088/1367-2630/14/9/095022. URL: https://dx.doi.org/10.1088/1367-2630/14/9/095022.

[Alq+13] Pierre Alquier et al. “Rank-penalized estimation of a quantum system”. In: *Physical Review A* 88.3 (Sept. 2013). ISSN: 1094-1622. DOI: 10.1103/physreva.88.032113. URL: http://dx.doi.org/10.1103/PhysRevA.88.032113.

[Cot+13] S. L. Cotter et al. “MCMC Methods for Functions: Modifying Old Algorithms to Make Them Faster”. In: *Statistical Science* 28.3 (Aug. 2013). ISSN: 0883-4237. DOI: 10.1214/13-sts421. URL: http://dx.doi.org/10.1214/13-STS421.

[Gel+13] A. Gelman et al. *Bayesian Data Analysis, Third Edition*. Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis, 2013. ISBN: 9781439840955. URL: https://books.google.be/books?id=ZXL6AQAAQBAJ.

[Kra+13] K. S. Kravtsov et al. “Experimental adaptive Bayesian tomography”. In: *Physical Review A* 87.6 (June 2013). ISSN: 1094-1622. DOI: 10.1103/physreva.87.062122. URL: http://dx.doi.org/10.1103/PhysRevA.87.062122.

[Fer14] Christopher Ferrie. “Quantum model averaging”. In: *New Journal of Physics* 16.9 (Sept. 2014), p. 093035. DOI: 10.1088/1367-2630/16/9/093035. URL: https://dx.doi.org/10.1088/1367-2630/16/9/093035.

[KF15] Richard Kueng and Christopher Ferrie. “Near-optimal quantum tomography: estimators and bounds”. In: *New Journal of Physics* 17.12 (Dec. 2015), p. 123013. ISSN: 1367-2630. DOI: 10.1088/1367-2630/17/12/123013. URL: http://dx.doi.org/10.1088/1367-2630/17/12/123013.

[LMS15] Chuanhai Liu, Ryan Martin, and Nick Syring. *Simulating from a gamma distribution with small shape parameter*. 2015. arXiv: 1302.1884 [stat.CO].

[Cai+16] Tony Cai et al. "Optimal large-scale quantum state tomography with Pauli measurements". In: *The Annals of Statistics* 44.2 (Apr. 2016). ISSN: 0090-5364. DOI: 10.1214/15-aos1382. URL: http://dx.doi.org/10.1214/15-AOS1382.

[GCC16] Christopher Granade, Joshua Combes, and D G Cory. "Practical Bayesian tomography". In: *New Journal of Physics* 18.3 (Mar. 2016), p. 033024. DOI: 10.1088/1367-2630/18/3/033024. URL: https://dx.doi.org/10.1088/1367-2630/18/3/033024.

[MA17] The Tien Mai and Pierre Alquier. "Pseudo-Bayesian quantum tomography with rank-adaptation". In: *Journal of Statistical Planning and Inference* 184 (May 2017), pp. 62–76. ISSN: 0378-3758. DOI: 10.1016/j.jspi.2016.11.003. URL: http://dx.doi.org/10.1016/j.jspi.2016.11.003.

[Dal20] Arnak S. Dalalyan. "Exponential weights in multivariate regression and a low-rankness favoring prior". In: *Annales de l'Institut Henri Poincaré, Probabilités et Statistiques* 56.2 (2020), pp. 1465–1483. DOI: 10.1214/19-AIHP1010. URL: https://doi.org/10.1214/19-AIHP1010.

[Guţ+20] M Guţă et al. "Fast state tomography with optimal error bounds". In: *Journal of Physics A: Mathematical and Theoretical* 53.20 (Apr. 2020), p. 204001. DOI: 10.1088/1751-8121/ab8111. URL: https://dx.doi.org/10.1088/1751-8121/ab8111.

[Har+20] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[Luk+20] Joseph M Lukens et al. "A practical and efficient approach for Bayesian quantum state estimation". In: *New Journal of Physics* 22.6 (June 2020), p. 063038. DOI: 10.1088/1367-2630/ab8efa. URL: https://dx.doi.org/10.1088/1367-2630/ab8efa.

[SYH20] Jun Suzuki, Yuxiang Yang, and Masahito Hayashi. "Quantum state estimation with nuisance parameters". In: *Journal of Physics A: Mathematical and Theoretical* 53.45 (Oct. 2020), p. 453001. ISSN: 1751-8121. DOI: 10.1088/1751-8121/ab8b78. URL: http://dx.doi.org/10.1088/1751-8121/ab8b78.

[Mai22] The Tien Mai. "An efficient adaptive MCMC algorithm for Pseudo-Bayesian quantum tomography". In: *Computational Statistics* 38.2 (July 2022), pp. 827–843. ISSN: 1613-9658. DOI: 10.1007/s00180-022-01264-x. URL: http://dx.doi.org/10.1007/s00180-022-01264-x.

[Ade+24] Tameem Adel et al. "A projected Langevin sampling algorithm for quantum tomography". unpublished. 2024.

[Ans24] Abdul Fatir Ansari. *Monte Carlo Sampling using Langevin Dynamics | Abdul Fatir Ansari*. [Online; accessed 18. May 2024]. Apr. 2024. URL: https://abdulfatir.com/blog/2020/Langevin-Monte-Carlo.

[Rig24] Tommaso Rigon. *Bayesian Computations.* [Online; accessed 16. May 2024]. Mar. 2024. URL: https://tommasorigon.github.io/CompStat.

[Wik24a] Wikipedia contributors. *Born rule — Wikipedia, The Free Encyclopedia.* https://en.wikipedia.org/w/index.php?title=Born_rule&oldid=1222150412. [Online; accessed 10-May-2024]. 2024.

[Wik24b] Wikipedia contributors. *Density matrix — Wikipedia, The Free Encyclopedia.* https://en.wikipedia.org/w/index.php?title=Density_matrix&oldid=1201935092. [Online; accessed 10-May-2024]. 2024.

[Wik24c] Wikipedia contributors. *Gamma distribution — Wikipedia, The Free Encyclopedia.* https://en.wikipedia.org/w/index.php?title=Gamma_distribution&oldid=1224587615. [Online; accessed 20-May-2024]. 2024.

[Wik24d] Wikipedia contributors. *Markov chain — Wikipedia, The Free Encyclopedia.* https://en.wikipedia.org/w/index.php?title=Markov_chain&oldid=1223581210. [Online; accessed 17-May-2024]. 2024.

[Wik24e] Wikipedia contributors. *Metropolis-adjusted Langevin algorithm — Wikipedia, The Free Encyclopedia.* https://en.wikipedia.org/w/index.php?title=Metropolis-adjusted_Langevin_algorithm&oldid=1211312860. [Online; accessed 18-May-2024]. 2024.

[Wik24f] Wikipedia contributors. *Monte Carlo integration — Wikipedia, The Free Encyclopedia.* https://en.wikipedia.org/w/index.php?title=Monte_Carlo_integration&oldid=1217233846. [Online; accessed 16-May-2024]. 2024.

[Wik24g] Wikipedia contributors. *Pauli matrices — Wikipedia, The Free Encyclopedia.* https://en.wikipedia.org/w/index.php?title=Pauli_matrices&oldid=1222373526. [Online; accessed 17-May-2024]. 2024.

[Wik24h] Wikipedia contributors. *Qubit — Wikipedia, The Free Encyclopedia.* https://en.wikipedia.org/w/index.php?title=Qubit&oldid=1222972285. [Online; accessed 13-May-2024]. 2024.

[Wik24i] Wikipedia contributors. *Tomography — Wikipedia, The Free Encyclopedia.* https://en.wikipedia.org/w/index.php?title=Tomography&oldid=1222258721. [Online; accessed 9-May-2024]. 2024.

# Appendix

## Derivation of the acceptance rate for the prob-estimator

### Derivation of $\log(R)$

The first expression $\log(R(\tilde{Y}, Y^{(t)}, \tilde{\gamma}, \gamma^{(t)}, V^{(t-1)}, \lambda, \alpha))$ (which we will write as $\log(R(\nu^*, \nu))$) is derived as follows: first, let us define

$$\tilde{\ell} = \ell(\sum_{i=1}^{d} \tilde{\gamma}_i V_i^{(t-1)} (V_i^{(t-1)})^\dagger, \mathbf{D}) \tag{6.1}$$

$$\ell^{(t)} = \ell(\sum_{i=1}^{d} \gamma_i^{(t)} V_i^{(t-1)} (V_i^{(t-1)})^\dagger, \mathbf{D}) \tag{6.2}$$

In this case of the prob-estimator, we do not have a proposal distribution per se (the sampling from $Gamma(\alpha, 1)$ corresponds to the prior), so the acceptance probability is computed as

$$R(\nu^*, \nu) = \frac{\pi(\nu^*|D)}{\pi(\nu|D)} = \frac{\exp(-\lambda\tilde{\ell})\pi(\nu^*)}{\exp(-\lambda\ell^{(t)})\pi(\nu)} \tag{6.3}$$

The prior $\pi(\nu) = \pi_1(\nu)$ is defined in this case as the Gamma distribution on $Y_i$:

$$\pi_1(y; \alpha, 1) = \frac{1}{\Gamma(\alpha)} y^{\alpha-1} \exp(-y) \tag{6.4}$$

We can now rewrite $\alpha$ with this definition:

$$R(\nu^*, \nu) = \frac{\exp(-\lambda\tilde{\ell})\tilde{Y}_i^{\alpha-1}\exp(-\tilde{Y}_i)}{\exp(-\lambda\ell^{(t)}){Y_i^{(t)}}^{\alpha-1}\exp(-Y_i^{(t)})} \tag{6.5}$$

where we have replaced $y$ by either $\tilde{Y}_i$ or $Y_i^{(t)}$. Finally, we can apply the log transform:

$$\log(R(\nu^*,\nu)) = \log\left(\frac{\exp(-\lambda\tilde{\ell})\tilde{Y}_i^{\alpha-1}\exp(-\tilde{Y}_i)}{\exp(-\lambda\ell^{(t)}){Y_i^{(t)}}^{\alpha-1}\exp(-Y_i^{(t)})}\right) \tag{6.6}$$

$$= \log(\exp(-\lambda\tilde{\ell})\tilde{Y}_i^{\alpha-1}\exp(-\tilde{Y}_i)) - \log(\exp(-\lambda\ell^{(t)}){Y_i^{(t)}}^{\alpha-1}\exp(-Y_i^{(t)})) \tag{6.7}$$

$$= -\lambda\tilde{\ell} + (\alpha-1)\log(\tilde{Y}_i) - \tilde{Y}_i - (-\lambda\ell^{(t)} + (\alpha-1)\log(Y_i^{(t)}) - Y_i^{(t)}) \tag{6.8}$$

$$= -\lambda\tilde{\ell} + \lambda\ell^{(t)} + (\alpha-1)\log(\frac{\tilde{Y}_i}{Y_i^{(t)}}) - \tilde{Y}_i + Y_i^{(t)} \tag{6.9}$$

## Derivation of $\log(A)$

The derivation of $\log(A(\tilde{V}, V^{(t)}, \gamma^{(t)}, \lambda)) = \log(A(\nu^*, \nu))$ is very similar to the one of $\log(R)$, however much simpler since the resulting expression does not involve the prior. The authors do not provide on how the derivation is done, however, the most probable explanation is that the prior is symmetric, hence we can simplify the numerator and denominator. It thus serves a double role of prior *and* proposal. Let's define

$$\tilde{\ell} = \ell(\sum_{i=1}^{d}\gamma_i^{(t)}\tilde{V}_i\tilde{V}_i^{\dagger}, \mathbf{D}) \tag{6.10}$$

$$\ell^{(t)} = \ell(\sum_{i=1}^{d}\gamma_i^{(t)}V_i^{(t)}(V_i^{(t)})^{\dagger}, \mathbf{D}) \tag{6.11}$$

Then, we can calculate $\log(A)$ as:

$$\log(A(\nu^*,\nu)) = \log(\frac{\pi(\nu^*)}{\pi(\nu)}) \tag{6.12}$$

$$= \log(\frac{\exp(-\lambda\tilde{\ell})}{\exp(-\lambda\ell^{(t)})}) \tag{6.13}$$

$$= -\lambda\tilde{\ell} + \lambda\ell^{(t)} \tag{6.14}$$

## Generation of the initial sample

In order to generate the initial sample $\hat{\rho}_0 \in \mathbb{C}^{d\times r}$ for MCMC algorithms, we use the following scheme: we start by generating a random unitary matrix $V$ of size $d \times r$, then we generate $\tilde{D}$ a diagonal matrix, where $\tilde{D}_{ii} \sim Gamma(1/r, 1)$, which we then normalize and create $D = \sqrt{\tilde{D}/\sum_{i=1}^{r}\tilde{D}_{ii}}$. Finally, the generated matrix is calculated as $\hat{\rho}_0 = VD$. This approach is inspired by what is done in the Langevin code and seems analogous to what is suggested in [MA17] to generate a rank-1 matrix. In practice, however, the matrix retains its initial rank $r$.