Technische Universität München
Fakultät für Physik

Quantum Science and Technology Master's thesis

Quantum tensor networks for sampling

Simon Botond

7. Juli 2025

Erstgutachter (Themensteller): Dr. Jeanette Miriam-Lorentz
Zweitgutachter: Alex Goessmann

Abstract

Efficient sampling from complex probability distributions is a cornerstone of modern machine learning, enabling critical tasks such as probabilistic inference, generative modeling, and optimization. Classical sampling methods, including brute force or Markov Chain Monte Carlo methods, face prohibitive computational challenges as the dimensionality and logical complexity of target distributions grow. This thesis investigates the potential of quantum computing—specifically amplitude amplification algorithms—to overcome these limitations and deliver practical advantages for machine learning sampling tasks. Through a systematic methodology, logical connectives and graphical models are mapped to quantum circuits via the Tensor network decomposition of their exponential family distributions. Simulations on up to ≈ 30 qubits (variables and ancillas together) demonstrate the scalability of quantum amplitude amplification, while comparative analyses with classical baselines quantify speedup and a possible turnover point. The results reveal that quantum-assisted sampling achieves quadratic speedups for structured distributions, though hardware constraints and noise pose significant challenges. This work bridges quantum computing and machine learning by formalizing a framework for integrating quantum sampling into classical workflows, offering a pathway toward scalable probabilistic inference in AI systems.

Contents

Chapter 1

Introduction

Motivation and Background

Sampling is an essential part of a machine learning algorithm. Whenever the program makes a decision, handles predictions with uncertainty or gives solutions to a problem, it is a form of sampling from a vast set of information based on past data and / or knowledge. Sampling therefore is fundamental in enabling probabilistic inference, uncertainty quantification, and optimization in high-dimensional spaces. As models are increasingly sophisticated — incorporating rich logical structures or large graphical dependencies — their complexity escalates and the computational demands of sampling algorithms grow exponentially. Classical methods, while foundational, struggle to balance accuracy and efficiency in high-dimensional or multimodal distributions. Quantum computing, with its inherent parallelism and interference capabilities suggest that quantum algorithms could offer speedups for certain sampling tasks, motivating a systematic investigation of their applicability and impact for machine learning [1][2]. This thesis explores the intersection of quantum computing and machine learning, focusing on how quantum amplitude amplification can revolutionize sampling tasks in artificial intelligence. The following sections elaborate on the motivation, challenges, opportunities, objectives, and structure of this work.

Classical Sampling in Machine Learning: Challenges

Classical sampling algorithms, such as brute-force sampling, rejection sampling, and Markov Chain Monte Carlo (MCMC) methods; face significant barriers in contemporary machine learning applications. High-dimensional distributions with complex dependencies or sharp peaks result in slow mixing times, autocorrelation, and exponential resource scaling. For example, sampling from Bayesian networks with hundreds of variables or training deep generative models with multimodal posteriors often becomes computationally intractable [3]. These challenges are exacerbated by the "curse of dimensionality," where the volume of the sampling space grows exponentially with the number of variables. Even state-of-the-art methods like Hamiltonian

Monte Carlo or variational inference require trade-offs between approximation accuracy and computational cost, leaving room for fundamentally new computational paradigms.

Quantum Computing: Opportunities for Sampling

Quantum computing introduces novel strategies for sampling through principles such as superposition or entanglement, enabling new computational paradigms for sampling. Recent research demonstrates that quantum algorithms can sample from complex distributions with fewer resources than classical counterparts, particularly for problems with combinatorial structure or where classical simulation is intractable or even computationally prohibitive [4][5]. Quantum amplitude amplification, a generalization of Grover's search algorithm [6] enables quadratic speedups in identifying "good" solution states, and with repeated application also amplifying them within unstructured search spaces. Thus by encoding probability distributions into quantum states, this technique can efficiently sample from distributions that would be unfeasible for classical systems. Also, recent advances in quantum hardware, such as improved gate fidelities and coherence times suggest that near-term devices may soon support practical implementations [7]. However, practical deployment depends on circuit depth, noise resilience, hardware capabilities and seamless integration with classical machine learning workflows, posing as critical challenges for quantum procedures to effectively overcome classical ones [8][9].

Thesis Objectives and Contributions

This thesis aims to bridge the gap between theory and practice in quantum-assisted sampling for machine learning, through the following contributions:

- A systematic methodology for mapping logical connectives and graphical models to tensor networks formalization of exponential family distributions, enabling their representation as quantum circuits.
- A framework for integrating quantum sampling into the TNREASON library, enabling hybrid quantum-classical inference for nested probabilistic models.
- Implementation and simulation of amplification-based quantum sampling routines on said mapped distributions.
- Compare quantum sampling with classical baselines, quantifying speed, accuracy, and scalability, and analyzing theoretical speedup over classical counterparts.

- Resource estimation studies for near-term quantum hardware, assessing the procedure's empirical performance, providing actionable insights for algorithm deployment on NISQ platforms.

Thesis Structure

The remainder of this work is organized as follows:

- **Chapter 2** reviews foundational concepts in logical connectives, graphical models, and exponential familiy distributions, with their tensor network representations.
- **Chapter 3** introduces quantum computing, defining quantum gates and measurements, tensor network mapping, then turning to quantum amplitude amplification.
- **Chapter 4** destribes my implementation of mapping a one dimensional logical formula to a quantum circuit, and performing the previously defined quantum measurement and amplitude amplification.
- **Chapter 5** benchmarks quantum sampling against classical methods, with analyzing scalability, and comparing theoretical speedups, and discusses practical deployment constraints.
- **Chapter 6** explores broader implications, limitations, and future research directions, finally concludes with a synthesis of contributions and an outlook on quantum sampling in AI.

Chapter 2

Foundations

2.1 Logical Connectives and Probabilistic Models

Artificial Intelligence (AI) and Machine Learning (ML) inherently involve sampling problems during training, learning, and optimization. Data and knowledge can be represented through diverse models, each with distinct advantages. Probabilistic graphical models—such as Bayesian networks and Markov Random Fields—have become dominant in real-world applications due to their natural handling of uncertainty and explicit representation of variable dependencies via graph structures [10]. Conversely in – the still well used – logical approaches data is naturally encoded as logical propositions with inference traditionally retrieving deterministic conclusions, though modern extensions integrate probabilistic interpretations and sampling [11]. With extending the practical usage of logics, the field of Statistical Relational AI bridges this gap, unifying logical relations with statistical models to handle uncertainty systematically. This synthesis, often termed *neuro-symbolic AI* [12], leverages the structured reasoning of logic and the uncertainty modeling of graphical models. For instance, logical rules can be embedded as soft constraints in probabilistic frameworks, enabling hybrid inference that balances symbolic precision with statistical robustness.
Both probabilistic and logical systems assume a factored structure, where system states are assignments to a set of variables. This structure admits a natural tensor representation, where variables correspond to tensor indices and their assignments to tensor entries. For example, a probability distribution over binary variables $X_1, \ldots, X_n$ can be encoded as a rank-n tensor $\mathcal{T}[X_1, \ldots, X_n]$, with each entry storing the probability of a specific joint assignment. Similarly, logical formulas map to tensors via one-hot encodings, where entries indicate formula satisfaction (1, TRUE) or violation (0, FALSE) of the given single logical connectives.

Tensor methods provide a unifying formalism for reasoning algorithms across probabilistic and logical frameworks. Marginalization in Bayesian networks corresponds to tensor contraction, while logical inference reduces to multilinear operations on formula-encoded tensors. This shared mathematical foundation enables seamless integration of probabilistic graphical models and symbolic logic, paving the way for scalable quantum sampling algorithms that exploit tensor network decompositions.

2.1.1 Propositional Logic in AI

Propositional logic is a fundamental tool in artificial intelligence for representing and manipulating knowledge in a formal, symbolic manner. In this framework, knowledge is encoded as a set of atomic propositions, each of which can be either true or false. These atomic propositions can be combined using logical connectives such as AND ($\wedge$), OR ($\vee$), XOR ($\oplus$), NOT ($\neg$), IMPLIES ($\rightarrow$) and BIJECTION ($\leftrightarrow$) to form more complex statements or formulas, the symbolization of which is:

$$[[A \wedge B] \rightarrow C] = A \textit{ and } B \textit{ implies } C, \tag{2.1}$$

meaning that our premises are:

- **Formula:** $[[A \wedge B] \rightarrow C]$
- **Premise 1:** $X \rightarrow C$
- **Premise 2:** $[A \wedge B] = X$, where X serves as an intermediate variable
- **Conclusion:** C

so that to have the formula true, we need the premises – defined by atomic (indivisible) statements – to be true.
Logical reasoning in AI often involves determining the satisfiability of a set of formulas, performing inference (e.g., deducing new facts from known ones), and checking entailment between statements. For example, given a knowledge base consisting of rules and facts, an AI system can use logical inference mechanisms such as Modus Ponens or Resolution to derive new conclusions. This symbolic approach underpins many classical AI systems, including expert systems, rule-based engines, and automated theorem provers.

Importantly, logical connectives can be mapped to indicator functions or binary variables, which facilitates their integration into probabilistic models. For instance, the truth value of a conjunction of variables can be represented as the product of their indicator variables. This mapping provides a bridge between symbolic logic and statistical modeling, enabling hybrid approaches that combine the strengths of both paradigms.

2.1.2 Graphical Models: Bayesian and Markov Networks

Probabilistic graphical models (PGMs) provide a powerful framework for representing the conditional dependencies among random variables in complex systems. There are two principal types of graphical models: Bayesian networks and Markov random fields (also known as Markov networks).

Bayesian networks are directed acyclic graphs (DAGs) where each node corresponds to a random variable, and directed edges encode conditional dependencies. The joint probability distribution factorizes according to the graph structure:

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid \mathrm{Pa}(X_i)) \tag{2.2}$$

where $\mathrm{Pa}(X_i)$ denotes the set of parent nodes of X_i. This factorization enables efficient inference and learning, especially when the graph is sparse.

Markov random fields are undirected graphs where nodes represent random variables and edges encode direct probabilistic interactions. The joint distribution is expressed as a product of potential functions over cliques (fully connected subsets of nodes):

$$P(X_1, \ldots, X_n) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(X_C) \tag{2.3}$$

where $\mathcal{C}$ is the set of cliques, ψ_C are non-negative potential functions, and Z is the partition function ensuring normalization. It also can be defined through a partition function and an energy function:

$$P(X) = \frac{1}{Z} exp \left(\sum_i w_i \phi_i(X) \right) \tag{2.4}$$

This energy based representation perfectly overlaps with the Exponential family representation.
Both Bayesian networks and Markov networks can represent high-dimensional distributions compactly, capturing complex dependencies among variables. Many logical knowledge bases and rule systems can be embedded within graphical models, providing a bridge between symbolic and probabilistic reasoning. For example, logical rules can be encoded as hard or soft constraints in the potentials of a Markov network or as conditional probability tables in a Bayesian network.

2.2 Exponential Family Distributions

2.2.1 Definition and Properties

The exponential family is a broad class of probability distributions that share a common functional (energy based) form, making them particularly amenable to statistical inference and learning [13][14].

Canonical parameterization

A probability distribution $p(x \mid \theta)$ belongs to the exponential family if it can be parametrized as:

$$p(x) = \exp\{\langle \theta, \phi(x) \rangle - A(\theta)\} \tag{2.5}$$

where θ is the vector of natural (canonical) parameters, $\phi(x)$ is the vector of sufficient statistics, and $A(\theta)$ is the log-partition function (also called the cumulant function) that ensures normalization, as per:

$$A(\theta) = \log \int\limits_{\chi^m} \exp\langle \theta, \phi(x) \rangle \nu dx$$

where we presume that the integral is finite, so this defnition ensures that $p(x)$ is properly normalized.

Mean parameterization

In addition to the 'natural' parameterization by canonical parameters and sufficient statistics, every exponential family admits a dual, alternative, so-called mean parameterization. The mean parameters μ_α are associated with the sufficient statistic ϕ_α as it is defined as the expected values of the sufficient statistics under the distribution:

$$\mu_\alpha = \mathbb{E}_p[\phi_\alpha(x)], \tag{2.6}$$

thus a mean parameter can be defined for each sufficient statistic, creating a set of all realizable mean parameters, known as the *marginal polytope* or *mean parameter space*:

$$\mathcal{M} := \{\mu \in \mathbb{R}^d \mid \exists\, p \; s.t. \; \mathbb{E}[\phi_\alpha(X)] = \mu_\alpha\} \tag{2.7}$$

A fundamental property of exponential families is that the mapping from canonical parameters θ to mean parameters μ is given by the gradient of the log-partition function:

$$\mu = \nabla_\theta A(\theta), \tag{2.8}$$

and, for minimal exponential families, this mapping is bijective between the interior of the mean parameter space and the natural parameter space. This duality underlies variational inference, maximum likelihood estimation, and moment-matching procedures In practice, mean parameters often correspond to marginal probabilities (e.g., node and edge marginals in graphical models), and many inference algorithms (such as mean field and variational methods) operate directly in the mean parameter space. For example marginalization can be understood as transfroming from one parametrization to the other.

Key properties of exponential family distributions include:

- **Sufficient statistics**: The function $\phi(x)$ captures all information about the data relevant to parameter estimation.

- **Conjugacy**: Many exponential family distributions admit conjugate priors, simplifying Bayesian inference.
- **Tractable moments**: Moments and cumulants of the distribution can be computed as derivatives of $A(\theta)$.
- **Generalization**: Many common distributions, such as the Bernoulli, multinomial, normal, and Poisson, and Markov Logic Networks as well are members of the exponential family.

In the context of AI and machine learning, exponential familiy representations can provide a general, unified framework that encompasses many commonly used distributions, such as probabilistic graphical models and logic-based systems. By choosing appropriate sufficient statistics (such as indicator functions for logical formulas), one can encode logical constraints and probabilistic dependencies within a single, tractable mathematical formalism.

2.2.2 Examples Relevant to Logical Connectives

Many logical knowledge bases, Markov logic networks (MLNs), and rule-based systems can be naturally expressed within the exponential family framework. In MLNs, for example, each logical formula ϕ_i is associated with a weight θ_i, and the probability of a world (i.e., a complete assignment of truth values) is given by:

$$P(X) \propto \exp\left(\sum_i \theta_i \phi_i(X)\right) \tag{2.9}$$

as seen in eq. 2.4. Here, $\phi_i(X)$ is an indicator function that evaluates to 1 if the formula is satisfied by X, and 0 otherwise. The weights θ_i determine the strength of each logical constraint: large positive values enforce hard constraints, while smaller values allow for soft, probabilistic reasoning.

This representation allows for the seamless integration of symbolic logic and statistical learning. Logical connectives can be encoded as sufficient statistics in the exponential family, and their weights can be learned from data. As the weights become large, the model approaches a purely logical system; as they decrease, the model becomes more probabilistic, capturing uncertainty and noise in the data.

2.3 Tensor Networks

2.3.1 Tensor Network Basics

A *tensor* is a multi-dimensional array generalizing scalars (0th order), vectors (1st order), and matrices (2nd order) to higher dimensions. Formally, an nth-order tensor

$\mathcal{T} \in \mathbb{R}^{d_1 \times \cdots \times d_n}$ has n indices, each ranging over dimensions $d_1, \ldots, d_n$. Tensors enable compact representation of high-dimensional functions, such as joint probability distributions over many variables.

A *tensor network* is a structured factorization of a high-order tensor into a collection of lower-order tensors connected via contractions over shared indices. Graphically, nodes represent tensors, and edges denote contracted indices.

Key operations in tensor networks include:

- **Contraction**: Summing over shared indices, e.g., $\mathcal{C}[i,j] = \sum_k \mathcal{A}[i,k]\mathcal{B}[k,j]$ for matrix multiplication.
- **Decomposition**: Approximating $\mathcal{T}$ as a network of smaller tensors via formats (like Canonical-Polyadic (CP), Matrix Product State (MPS), or tensor train (TT)).
- **Normalization**: Enforcing constraints (e.g., non-negativity) to represent valid probability distributions.

Tensor networks excel at capturing locality and hierarchy in data. For example, a tree tensor network (TTN) mirrors the structure of a Bayesian network, with each node representing a conditional probability table and edges encoding variable dependencies.

2.3.2 Tensor Networks for Probabilistic Models

The joint probability distribution of a graphical model can be encoded as a tensor where each entry $\mathcal{T}[x_1, \ldots, x_n]$ stores $P(X_1 = x_1, \ldots, X_n = x_n)$. Tensor networks factorize this high-dimensional tensor into localized components reflecting the model's conditional independence structure.

Example: Bayesian Network as a Tensor Network
Consider a Bayesian network over binary variables X_1, X_2, X_3 with edges $X_1 \to X_2$ and $X_2 \to X_3$. The joint distribution is:

$$P(X_1, X_2, X_3) = P(X_1)P(X_2|X_1)P(X_3|X_2) \tag{2.10}$$

Which factorizes to a tensor network:

$$\mathcal{T}[X_1, X_2, X_3] = \sum_Y \mathcal{A}[X_1]\mathcal{B}[X_1, X_2, Y]\mathcal{C}[X_2, X_3, Y] \tag{2.11}$$

where Y is an auxiliary index connecting the two conditional tensors $\mathcal{B}[X_1, X_2, Y]$ and $\mathcal{C}[X_2, X_3, Y]$, with $\mathcal{A}[X_1]$ being the prior tensor to X_1.

Example: Markov Random Field as a Tensor Network
Consider a Markov random field (MRF) over binary variables X_1, X_2, X_3 with cliques $\{X_1, X_2\}$ and $\{X_2, X_3\}$. Its joint distribution is:

$$P(X) = \frac{1}{Z}\psi_{12}(X_1, X_2)\psi_{23}(X_2, X_3) \tag{2.12}$$

This corresponds to a tensor network:

$$\mathcal{T}[X_1, X_2, X_3] = \sum_{Y} \psi_{12}[X_1, X_2, Y]\psi_{23}[Y, X_2, X_3] \tag{2.13}$$

where Y is an auxiliary index connecting the two clique tensors. This meets the Tensor Network representation of Propositional Logical formulas as well, where we can define the unique Logical connectives through the $\psi_{X_n,X_m,Y}$ functions, with X_n, X_m being the variables for the connective and the Y should be the 'ancilla' connecting the connectives to have the Logical formula in the end.

Key Applications:

- **Marginalization**: Computing $P(X_i)$ reduces to contracting all indices except X_i. For a TTN, this scales linearly in n rather than exponentially.
- **Sampling**: Ancestral sampling in Bayesian networks corresponds to sequential contractions in directed tensor networks.
- **Learning**: Maximum likelihood estimation becomes tensor factorization, e.g., using alternating least squares (ALS).

2.4 Tensor Network Representation of Exponential Families

Exponential family distributions, due to their structured parameterization and sufficient statistics, are naturally suited for representation as tensor networks. This correspondence enables efficient manipulation, marginalization, and sampling—crucial for high-dimensional probabilistic models and for mapping to quantum circuits.

2.4.1 Slice Tensor Decomposition

A central technique for representing high-order tensors arising from exponential family models is *slice tensor decomposition.* In this approach, a high-dimensional tensor $\mathcal{T}[X_1, \ldots, X_n]$ encoding the joint distribution is factorized into a sum of lower-rank tensors (slices) along one or more modes:

$$\mathcal{T}[X_1, \ldots, X_n] = \sum_{k=1}^{r} \lambda_k \, \mathcal{A}_k[X_1] \otimes \mathcal{B}_k[X_2, \ldots, X_n] \tag{2.14}$$

where λ_k are scalar coefficients, and $\mathcal{A}_k$, $\mathcal{B}_k$ are subtensors or vectors corresponding to particular slices. This decomposition preserves the conditional independence structure of the underlying graphical or logical model and allows for efficient storage and computation, especially when the tensor admits a low-rank structure.

Slice decomposition is particularly powerful for models with factorizable sufficient statistics, such as those arising in Markov logic networks or graphical models, where each logical formula or clique potential can be associated with a tensor slice. The resulting network of slices can then be contracted (multiplied and summed over shared indices) to recover marginals or conditionals.

2.4.2 Operational Mapping: Step-by-Step Example

To illustrate, consider a simple exponential family model over three binary variables X_1, X_2, X_3. The joint distribution can be encoded as a rank-3 tensor $\mathcal{T}[X_1, X_2, X_3]$. Suppose the model factorizes as:

$$\mathcal{T}[X_1, X_2, X_3] = \sum_{k=1}^{r} \mathcal{S}_k[X_1] \cdot \mathcal{U}_k[X_2] \cdot \mathcal{V}_k[X_3] \tag{2.15}$$

where each $\mathcal{S}_k, \mathcal{U}_k, \mathcal{V}_k$ is a vector (or slice) over its respective variable, and r is the decomposition rank. This format is equivalent to the canonical polyadic (CP) decomposition and is especially efficient when r is small compared to the full tensor size.

In practice, such decompositions can be obtained via algebraic methods (e.g., alternating least squares) or by exploiting the structure of the model (e.g., using the factor graph or logical formulae). The resulting tensor network can then be contracted to compute marginals, conditionals, or to generate samples.

Chapter 3

Quantum Sampling Algorithms

3.1 Quantum Gates, Circuits, and Measurement

Quantum gates, circuits, and measurements together provide the operational foundation for all quantum algorithms. Gates manipulate the amplitudes and phases of quantum states, circuits implement complex transformations, and measurement extracts classical information, enabling quantum algorithms to outperform their classical counterparts in sampling, search, and inference [15].

This section summarizes the essential elements relevant for later procedures, such as amplitude amplification and the collective quantum sampling algorithm.

3.1.1 Qubit States and Quantum Registers

A single qubit is described by a state vector in a two-dimensional Hilbert space,

$$|a\rangle = v_0 |0\rangle + v_1 |1\rangle ,$$

where v_0 and v_1 are complex amplitudes satisfying $|v_0|^2 + |v_1|^2 = 1$. The basis states $|0\rangle$ and $|1\rangle$ are represented as column vectors:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

For multi-qubit systems, the overall state is given by the tensor product of individual qubit states. For example, a two-qubit state is

$$|\psi\rangle = v_{00} |00\rangle + v_{01} |01\rangle + v_{10} |10\rangle + v_{11} |11\rangle .$$

3.1.2 Quantum Gates: Basic Building Blocks

Quantum gates are unitary operations acting on one or more qubits. They generalize classical logic gates but can create superposition and entanglement, enabling quantum parallelism.

Single-Qubit Gates:

- **Pauli-X (NOT) Gate:** Flips $|0\rangle$ to $|1\rangle$ and vice versa.
$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$
- **Pauli-Y and Pauli-Z Gates:** Y combines a bit and phase flip; Z applies a phase flip to $|1\rangle$.
$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$
- **Hadamard (H) Gate:** Creates superposition. Applied to $|0\rangle$, it yields $(|0\rangle + |1\rangle)/\sqrt{2}$.
$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$
- **Phase Gates:** modifies the phase of the quantum state, but the measurement probabilities stay.
$$P(\varphi) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\varphi} \end{bmatrix}$$
- **Rotation Gates:** $R_x(\theta)$, $R_y(\theta)$, and $R_z(\theta)$ rotate the qubit state around the respective axes by angle θ.
$$R_{P_g}(\theta) = e^{-iP_g\theta/2} = \cos\frac{\theta}{2} \cdot 1 - i \sin\frac{\theta}{2} \cdot P_g$$

Multi-Qubit Gates:

- **SWAP Gate:** Exchanges the states of two qubits.
$$Swap = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
- **CNOT (Controlled-NOT) Gate:** A two-qubit gate that flips the target qubit if the control qubit is $|1\rangle$. Essential for generating entanglement.
$$CNot = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

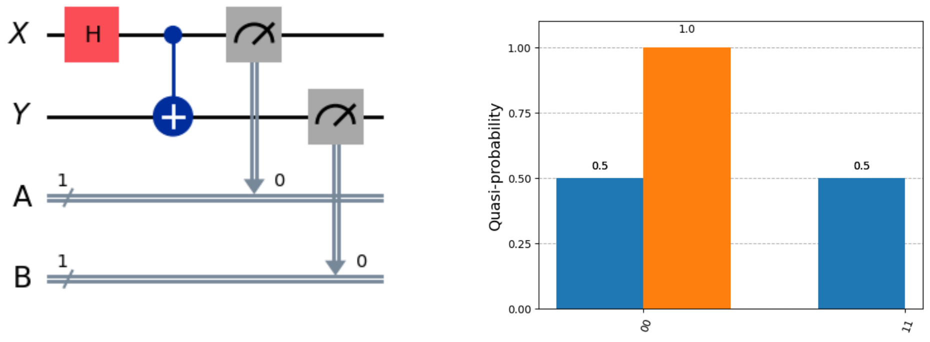
- **Toffoli (CCNOT) Gate:** A three-qubit gate; flips the third (target) qubit if both control qubits are $|1\rangle$. Important for universal reversible computation and error correction.

$$Toffoli = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Quantum gates are represented as unitary matrices. A gate acting on n qubits is a $2^n \times 2^n$ unitary matrix.

3.1.3 Quantum Circuits

A quantum circuit is a sequence of quantum gates applied to qubits, typically followed by measurement. Circuits are often depicted as diagrams, with qubits as horizontal lines and gates as symbols acting on these lines. For example, the circuit for creating a Bell state applies a Hadamard gate to the first qubit, followed by a CNOT with the first as control and the second as target, as seen in the figure below 3.1.



(a) Entangling circuit (b) Result of circuit measurement

Figure 3.1: Example circuit, resulting in $\hat{U}|00\rangle \rightarrow \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. Later it will be changed up to make it fit with the overall color scheme etc, right now it is a placeholder

The action of a circuit on an initial state $|\psi_0\rangle$ is a sequence of unitary transformations:

$$|\psi_{\text{final}}\rangle = U_k \cdots U_2 U_1 |\psi_0\rangle ,$$

where each U_i is a quantum gate.

3.1.4 Measurement in Quantum Mechanics

A measurement is the testing or manipulation of a physical system to yield a numerical result. In quantum mechanics it is when a quantum state collapses to a classical outcome. In the computational basis, measuring a qubit in state $|a\rangle = v_0 |0\rangle + v_1 |1\rangle$ yields $|0\rangle$ with probability $|v_0|^2$ and $|1\rangle$ with probability $|v_1|^2$.

For multi-qubit systems, measurement projects the state onto one of the basis vectors $|k\rangle$, with probability $|v_k|^2$. The outcome is inherently probabilistic, a fundamental departure from classical computation.

Entanglement and Measurement: Measurement on one qubit of an entangled pair instantaneously determines the outcome of the other, a phenomenon with no classical analog. For example, measuring one qubit of the Bell state $(|00\rangle + |11\rangle)/\sqrt{2}$ collapses the other qubit to the same value, although causality must be preserved, hence the no communication theorem should be obeyed [16]. It will be utilized in my implementation to enable backpropagation as a learning procedure.

3.2 Amplitude Amplification: Theory and Practice

Amplitude amplification is a quantum algorithmic technique that generalizes Grover's search [6], providing a quadratic speedup for the identification or sampling of "good" states in a large, unstructured search space. The power of amplitude amplification lies in its ability to systematically increase the probability amplitude associated with desirable outcomes, while keeping the state space normalized, making it a central primitive for quantum-enhanced sampling in machine learning.

3.2.1 Grover's Algorithm and Generalizations

Grover's algorithm is the canonical example of amplitude amplification. In the classical setting, searching for a marked item in an unsorted database of size N requires, on average, $O(N)$ queries, as with random selection, each and every item would have a $\frac{1}{N}$ probability of finding. Grover's quantum approach reduces this to $O(\sqrt{N})$ by exploiting quantum superposition and entanglement.

The algorithm operates in an N-dimensional Hilbert space $\mathcal{H}$, where each basis state $|k\rangle$ represents a possible solution. A Boolean oracle function $\chi : \{0,1\}^n \rightarrow$

$\{0,1\}$ identifies "good" states. The oracle operator $\mathbf{O}$ applies a phase flip to these states, while the diffusion operator $\mathbf{D}$ amplifies their amplitudes.

Brassard et al. [17] extended Grover's idea to arbitrary initial states and general quantum algorithms, formalizing the amplitude amplification operator:

$$\mathbf{Q} = -\mathcal{A}\mathbf{B}_0\mathcal{A}^{-1}\mathbf{B}_\chi$$

where $\mathcal{A}$ prepares the initial state, $\mathbf{B}_\chi$ flips the phase of good states, and $\mathbf{B}_0$ flips the phase of the all-zero state. Repeated application of $\mathbf{Q}$ rotates the quantum state in the two-dimensional subspace spanned by the good and bad components, exponentially increasing the probability of measuring a good state.

3.2.2 Oracle and Diffusion Operator Design

The effectiveness of amplitude amplification hinges on the careful design of the oracle and diffusion operators.

Oracle Operator O: The oracle is a unitary operator that marks the set of good states by flipping their phase. For a basis state $|k\rangle$, in $\mathcal{S} := \{|k\rangle\}_{k=0}^{N-1}$,

$$\mathbf{O}\,|k\rangle = (-1)^{\chi(k)}\,|k\rangle$$

where $\chi(k) = 1$ for good states and 0 otherwise. In practical terms, the oracle is implemented as a quantum circuit that evaluates the Boolean function χ and applies a controlled-Z or multi-controlled Toffoli gate to flip the phase of the target states. For simple logical connectives, such as AND or OR, the circuit construction is straightforward. For more complex constraints, the circuit depth increases, but the principle remains the same: the oracle must be a reversible, unitary operation that encodes the solution set into phase flips.

Diffusion Operator D: The diffusion operator, or "inversion about the mean," amplifies the amplitudes of the marked states. For an initial state $|\psi\rangle = \mathcal{A}\,|0\rangle$, the diffusion operator is

$$\mathbf{D} = 2\,|\psi\rangle\,\langle\psi| - \mathbb{I}$$

This operator reflects the quantum state about the initial state vector. In the special case where $|\psi\rangle$ is the uniform superposition, $\mathbf{D}$ can be implemented by Hadamard gates, a phase flip on $|0\rangle$, and another round of Hadamards. For non-uniform initial states, the diffusion operator is constructed by applying the inverse of the state preparation circuit, a phase flip on $|0\rangle$, and then re-applying the state preparation.

Geometric Interpretation: The interplay between the oracle and diffusion operators can be visualized as a sequence of reflections in a two-dimensional subspace. With a projection operator emerging from the definition of $\mathbf{O}$;

$$\mathcal{P} := \sum_{\chi(k)=1} |k\rangle \langle k|$$

defining the good and bad subspaces respectively:

$$\mathcal{H}_G := Im(\mathcal{P}) = span\{|k\rangle \in \mathcal{S}_{op} \mid \chi(k) = 1\}$$
$$\mathcal{H}_B := Ker(\mathcal{P}) = span\{|k\rangle \in \mathcal{S}_{op} \mid \chi(k) = 0\}$$

If we adopt the convention of $\sum'$ for summation over the solution states, and $\sum''$ for summation over the non-solutions, we can define normalized states as such:

$$|x_G\rangle = \sqrt{\frac{1}{M}} \sum_x{}' |x\rangle$$
$$|x_B\rangle = \sqrt{\frac{1}{N-M}} \sum_x{}'' |x\rangle$$

where $\mathbf{N}$ is the number of states (2^n for n qubits) and $\mathbf{M}$ is the number of solution states. Thus the initial state can be realized as:

$$|\psi\rangle = \sqrt{\frac{N-M}{N}} |x_B\rangle + \sqrt{\frac{M}{N}} |x_G\rangle = \cos\left(\frac{\alpha}{2}\right) |x_B\rangle + \sin\left(\frac{\alpha}{2}\right) |x_G\rangle$$

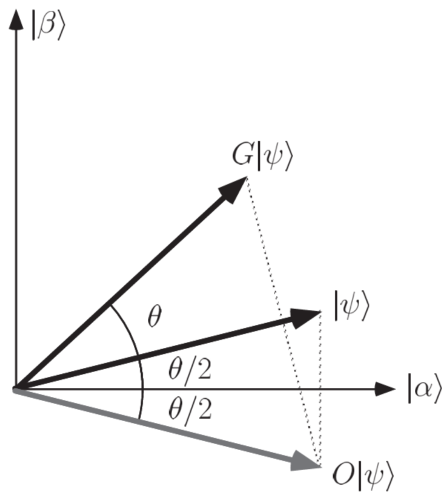
This is then a two-dimensional subspace spanned by the vectors $|x_G\rangle$ and $|x_B\rangle$ is stable under the action of $\mathbf{Q}$. The oracle reflects the state about the bad subspace, while the diffusion operator reflects about the initial state.

The composition of these two reflections is a rotation by α towards the good subspace. This geometric process underlies the quadratic speedup of amplitude amplification.

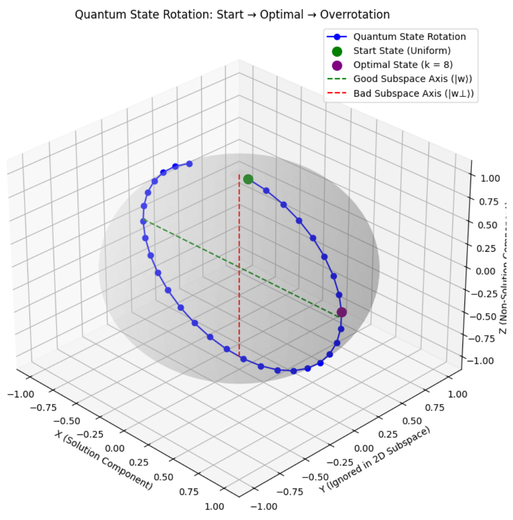
3.2.3 Parameter Update Interpretation

The action of the amplitude amplification operator $\mathbf{Q}$ can be interpreted as two reflections – once over the bad states, and then over the average amplitude of our states (i.e. the register itself in the two-dimensional subspace) – which geometrically is a rotation in the two-dimensional subspace of the Hilbert space spanned by the projections of the initial state onto the good and bad subspaces. The initial state can be written as

$$|\psi\rangle = \sin\left(\frac{\alpha}{2}\right) |\psi_g\rangle + \cos\left(\frac{\alpha}{2}\right) |\psi_b\rangle$$



(a) Grover rotation in the 2 dimensional subspace H_ψ



(b) Repeated applications in the complete state space

where $|\psi_g\rangle$ and $|\psi_b\rangle$ are normalized projections onto the good and bad subspaces respectively, ultimately resulting in the 2 dimensional subspace, which I will refer to as H_ψ. The result of said rotations in H_ψ can be depicted as seen in figure 3.2a. As the state is stable under the action of $\mathbf{Q}$, the rotaion angle of $\frac{\alpha}{2}$ is given. Thus, Each application of $\mathbf{Q}$ rotates the state by $2 \cdot \frac{\alpha}{2} = \alpha$, such that after k iterations,

$$\mathbf{Q}^k |\psi\rangle = \sin\left((2k+1)\frac{\alpha}{2}\right)|\psi_g\rangle + \cos\left((2k+1)\frac{\alpha}{2}\right)|\psi_b\rangle$$

The probability of measuring a good state thus increases quadratically with the number of iterations, reaching a maximum when $(2k+1)\frac{\alpha}{2} \approx \pi/2$, resulting in:

$$k = \lfloor \frac{\pi}{4}\sqrt{\frac{1}{P_g}} \rfloor \tag{3.1}$$

meaning, that after exactly k rotations we will have a maximum amplification of our state. This repeated applications, each resulting in a rotation with angle α, can be seen in a 3-dimensional representation in figure 3.2b, where the axes are the *bad* ($|\psi_b\rangle$) and *good* ($|\psi_g\rangle$) states, spanning H_ψ.

3.2.4 Scalability and Simulator Limitations

The scalability of amplitude amplification is determined by the complexity of the oracle and diffusion operators, as well as the available quantum resources. For simple logical connectives and small models, both operators can be implemented with shallow circuits, and the algorithm can be simulated efficiently on classical hardware. For more complex logical formulas or high-dimensional models, the circuit depth and qubit count increase, and classical simulation becomes intractable.

In this work, quantum simulations were performed for circuits up to 25 qubits using the Qiskit Aer simulator, balancing accuracy and computational feasibility. For larger problem sizes, efficient simulation may require tensor network-based methods or access to real quantum hardware. The quadratic speedup of amplitude amplification remains, but practical implementation is constrained by current hardware and software limitations.

3.3 From Tensor Networks to Quantum Circuits

Tensor networks unify the representation of probabilistic models with quantum state encoding, as probabilistic models can be represented 2.3.2, while quantum states, such as matrix product states (MPS) and projected entangled pair states (PEPS) are inherently represented as tensor networks [18]. Therefore tensor networks provide a unified framework for representing and manipulating probabilistic models, as both probabilistic algorithms (e.g.inference), and quantum algorithms (such as Amplitude amplification) operate on these networks, providing a pathway to quantum-enhanced sampling.

3.3.1 Quantum Circuit Construction

To map a tensor network to a quantum circuit, each tensor (or slice) is encoded as a quantum state and/or a set of parameterized quantum gates. The contraction of tensors, which in the classical setting corresponds to summing over shared indices, is implemented in the quantum setting by entangling qubits or applying multi-qubit gates. For example, in a network where each variable X_i is binary, a qubit is allocated for each variable, and the entries of the tensor slices determine the amplitudes of basis states, which can be applied as parameterized gate-sets.

The construction proceeds as follows:

1. **Initialization**: Prepare the quantum register in a reference state (e.g., $|0\rangle^{\otimes n}$).

2. **State Preparation**: Apply a sequence of single- and multi-qubit gates to encode the tensor slices, such that the resulting quantum state amplitudes correspond to the (normalized) entries of the target tensor network. For instance, parameterized rotation gates (R_X, R_Y, R_Z) can be used to set the amplitudes according to the tensor entries, while Hadamard gates may be used to create initial superpositions

3. **Entanglement**: Use CNOT and controlled gates to implement contractions between slices, reflecting dependencies in the original probabilistic model. In my model later, entanglement will be inherently present due to the nature of the encoding, with multi-qubit entangling gates (Toffoli) being frequently utilized over ancillas (i.e. shared indices of the Tensor Network).

This mapping is particularly efficient when the tensor network has low rank or sparse structure, as is often the case for models with strong conditional independence. In such scenarios, the number of required gates and the circuit depth can be kept polynomial in the number of variables, making the approach feasible for near-term quantum devices [19].

3.3.2 Gate Decomposition and Resource Estimates

The quantum circuit depth and resource requirements depend on the structure and rank of the underlying tensor network:

- **Qubit Count**: Each variable in the model typically requires one qubit for binary variables (or $\lceil \log_2 d_i \rceil$ qubits for d_i-ary variables) as well as one auxiliary qubit (i.e. ancilla qubit) for connecting the cliques of probabilistic model representations 2.3.2.

- **Gate Complexity**: The number of gates scales with the number of tensor slices and the connectivity of the network. For a CP decomposition of rank r over n variables, the circuit requires $\mathcal{O}(rn)$ parameterized gates [20].

- **Circuit Depth**: For chain-like (MPS) or tree-like (TTN) tensor networks, the depth is $O(n)$ or $O(\log n)$, respectively.

In summary, the mapping from exponential family distributions to tensor networks, and subsequently to quantum circuits, enables scalable quantum sampling for complex probabilistic models. This approach exploits both the structure of the underlying model and the computational power of quantum devices, providing a pathway to quantum advantage in probabilistic inference and machine learning.

Chapter 4

Experimental Evaluation

4.1 Use Case: Logical Connective Sampling

4.1.1 Problem Setup

In this section, we consider a representative logical connective as the target for quantum sampling. The problem is defined as follows: given a logical formula $\mathbf{F}$ (e.g., a conjunction or disjunction of n Boolean variables), our goal is to efficiently sample satisfying assignments ("good" states) from the exponentially large (2^n) space of all possible assignments. For concreteness, let us focus on a n-qubit system encoding a logical connective such as seen in the previous example 2.1, which for the sake of the example will be 6 variables (mapped as qubits), with 19 connectives (ancillas). The set of "good" states, $\mathcal{G}$, consists of all assignments that satisfy the formula. The initial probability of selecting a good state from the uniform distribution is $P_g = M/N = |\mathcal{G}|/2^n$, in the case of my example shown, exactly $P_g = M/N = 1/2^6$. Our aim is to amplify P_g to near-unity using quantum amplitude amplification.

4.1.2 Mapping Logical Formula to a Quantum state

The logical formula $\mathbf{F}$ can be encoded as an exponential family distribution as per 2.2.2:

$$p(x) = \frac{1}{Z} \exp\left\{\theta \cdot \hat{1}[\mathbf{F}]\right\} \tag{4.1}$$

where $\hat{1}[\mathbf{F}]$ is the indicator function for the formula, serving as the sufficient statistics for the exponential family, whilst θ is the canonical parameter, a scalar in our case, and Z is the normalization constant.
I have defined the weight added to the formula be:

- unit, whenever the indicator function gives a result of 'FALSE'/0
- e^{θ}, whenever it is 'TRUE'/1

Using this representation, θ can be understood as the weight of the truth value of the formulas, coming directly from the sufficient statistics being the indicator function of the logical formula, and therefore the weights being:

- $p(x=1) = \frac{Me^{\theta}}{(N-M)+Me^{\theta}} = P_g$
- $p(x=0) = \frac{N-M}{(N-M)+Me^{\theta}} = P_b$

with $N = 2^n$ as all the possible states, M of which are 'Good', i.e. results in the indicator function giving value of 1. This way it gives us a great starting point as for $\theta = 0$, this reduces to the uniform distribution, and as $\theta \to \infty$, the distribution concentrates on the satisfying assignments. Thus our expectation is solely this, that when the locical formula is mapped to a quantum circuit, we start from a uniform distribution of variables, and slowly amplify the probability of the 'Good' states, only the canonical parameter of the formula will change, as it will diverge to infinite values, whilst the sufficient statistics should remain the same, esentially keeping us in the same exponential family with finely tuned variables, to match our amplified quantum states.
Then we need to apply slice decomposition on the defined exponential family to map each intricate connection to the quantum circuit at hand. Following the section 2.4.1 we can slice the logical formula by each connective, thus making it straightforward to iteratively have every detail embedded into our initial state $|\psi\rangle$, prepared by the unitary $\mathcal{A}$, the direct result of the decomposition.

4.1.3 Amplitude Amplification for Exponential Families

Our expectation is that only the canonical parameter of the formula will change, as it will diverge to infinite values, whilst the sufficient statistics should remain the same, esentially keeping us in the same exponential family with finely tuned variables, to match our amplified quantum states. As when the locical formula is mapped to a quantum circuit, we start from a uniform distribution of variables, and slowly amplify the probability of the 'Good' states, thus changing the probability of finding such.

Mathematically, if the initial probability of a good state is P_{g0}, then after k applications of amplitude amplification, the probability becomes $P_k = \sin^2\left((2k+1)\frac{\alpha}{2}\right)$, where $\frac{\alpha}{2} = \arcsin(\sqrt{P_{g0}})$, is the angle of the state in H_ψ. In exponential family terms, this corresponds to a shift in the canonical parameter:

$$\theta_k = \ln\left(\frac{(1-P_{g0})P_k}{P_{g0}(1-P_k)}\right) \tag{4.2}$$

which can be rewritten as:

$$\theta_k = \ln\left(\frac{P_k}{1-P_k}\right) - \ln\left(\frac{P_{g0}}{1-P_{g0}}\right) \tag{4.3}$$

So θ_k is just the difference of log-odds (i.e. *logit*, the inverse of the *sigmoid*) between the current probability $P_k$and the initial probability P_{g0}.

This provides a direct link between quantum amplitude amplification and classical parameter updates in probabilistic models.

Circuit Construction: In practice, the amplitude amplification circuit is constructed as follows:

1. Prepare the initial state $|\psi\rangle$ using the unitary $\mathcal{A}$.
2. Apply the oracle $\mathbf{O}$.
3. Apply the diffusion operator $\mathbf{D}$.
4. Repeat the sequence $\mathbf{Q} = \mathbf{DO}$ for the optimal number of iterations.
5. Measure the final state in the computational basis.

The oracle and diffusion operators are implemented as modular subroutines, allowing for flexibility in defining different logical connectives or constraints.

4.1.4 Quantum Circuit Implementation

The exponential family representation is mapped to a quantum circuit through the slice decomposition of its Tensor network representation.
as follows:

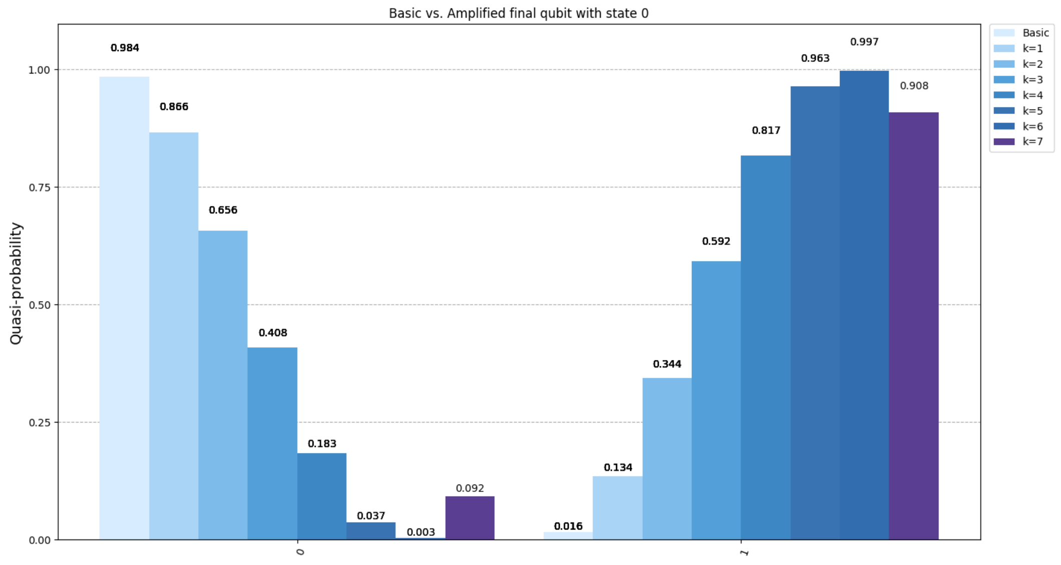
- **State Preparation:** Apply Hadamard gates to all qubits to create a uniform superposition over all 2^n states, and then apply CNOT/Toffoli gates to encode the logical connectives, which are projected to an ancilla.
- **Oracle Construction:** Implement an oracle O_ϕ that flips the phase of the "good" states (those satisfying ϕ). This is realized using a equence including Z gates corresponding to the binary structure of ϕ.
- **Diffusion Operator:** Construct the diffusion operator D as an inversion about the mean, typically using the inversible state preparation unitary $\mathcal{A}^+$, a multi-qubit Z gate on $|0\rangle^{\otimes n}$, and another round $\mathcal{A}$.
- **Amplitude Amplification:** Apply the operator $Q = DO_\phi$ for k iterations, where k is chosen to maximize the probability of measuring a good state 3.2.3.

The pseudocode for each part of the procedure can be seen in reference appendix for pseudocode.

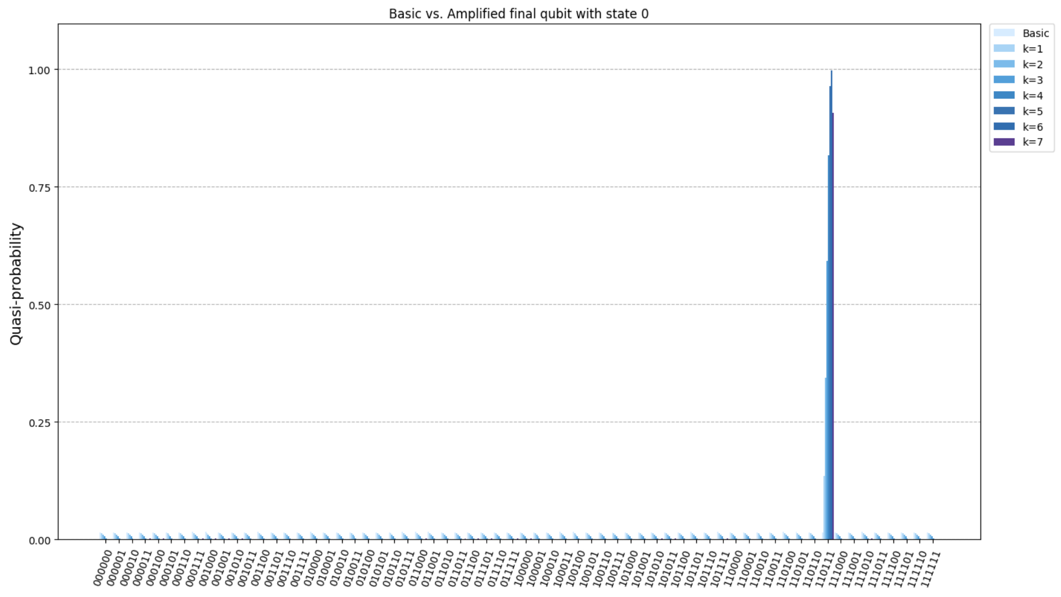
4.1.5 Results and Analysis

After k rounds of amplitude amplification (so one additional turn for overrotation after near perfect amplification), the probability of measuring a good state increases from P_0 to $P_k = \sin^2((2k+1)\alpha)$, where $\alpha = \arcsin(\sqrt{P_0})$. The canonical parameter of the exponential family updates as seen in eq 4.3.

The said changes in probabilities on the given logical formula can be seen in the following figures:



(a) Changes in the probability of measuring TRUE value on the logical connective

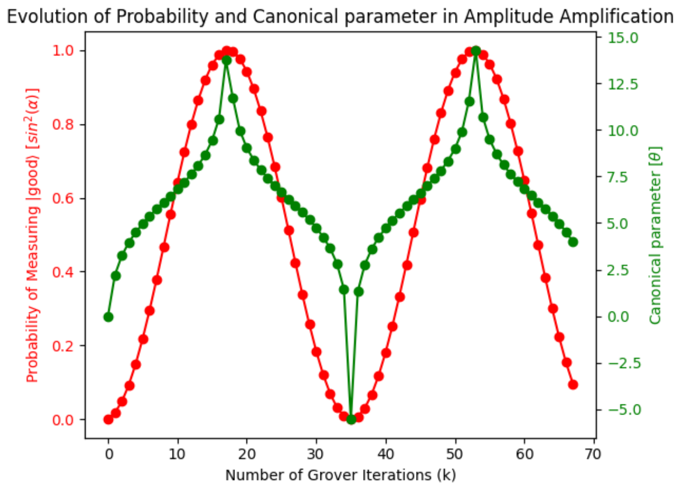


(b) Changes in the input values resulting in given $P_{\mathcal{G}}$

Figure 4.1: Series of Grover rotations on a given logical formula

We can see that we have achieved near perfect amplification after 6 steps, which is in line with our mathematical framework, given in 3.2.3, that $k = \lfloor \frac{\pi}{4}\sqrt{\frac{1}{P_g}} \rfloor = \lfloor \frac{\pi}{4}\sqrt{\frac{64}{1}} \rfloor = 6$. The fifth step, as expected, results in a lower probability, meaning that we have overrotated.

The changes in angle and canonical parameters are:



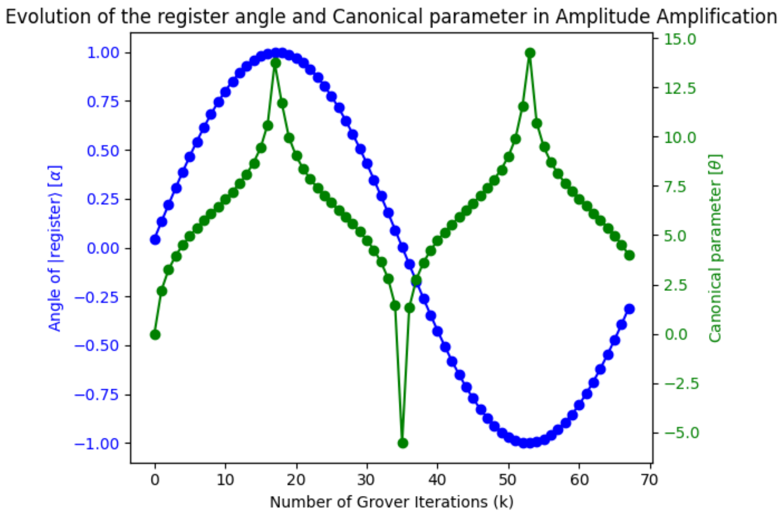


Figure 4.2: Series of Grover rotations on a given logical formula

Where we can see the expected shootout in the canonical parameters of the exponential family. The log-odds function, with which we define θ_k, has vertical asymptotes at $P = 0$ and $P = 1$, with the added change in behaviour coming from the non-monotonic, but sinusoidal change of P_k. This understates those explosive "jumps", as the function reacts extremely to values close to 0 or 1, which shows us that with near perfect amplification, when $P_{\mathcal{G}} \to 1$, we have a local maxima in the canonical parameter. Furthermore, this shows that the sampling procedure remains within the same exponential family, but with an updated parameter θ_k reflecting the amplified probability. The updated θ_k can be enumerated as:

Iteration (k)	P_k	θ_k
0	0.016	0
1	0.134	1.543
2	0.344	2.763
3	0.592	3.781
4	0.817	4.905
5	0.963	6.668
6	0.997	9.215
7	0.908	5.698

Table 4.1: Amplification of probability and exponential family parameter over iterations.

Interpretation: The amplitude amplification procedure efficiently concentrates probability mass on the set of satisfying assignments, achieving quadratic speedup over classical rejection sampling. Importantly, the process preserves the exponential family structure of the distribution, with the canonical parameter θ evolving according to the amplification dynamics. This demonstrates the compatibility of quantum amplitude amplification with classical probabilistic modeling frameworks.

4.2 Generalization and Scalability

Put the more dimensional problem here

Chapter 5

Benchmarking and Comparative Analysis

5.1 Classical Sampling Methods

Sampling from complex probability distributions is a foundational task in statistics and machine learning, enabling probabilistic inference, uncertainty quantification, and model training 1. Classical sampling methods primarily include the brute-force approach, rejection sampling and Markov Chain Monte Carlo (MCMC) methods, each with distinct strengths and limitations [21].

5.1.1 Brute-Force Enumeration

Brute force sampling, also known as exhaustive enumeration, is the most direct approach to sampling from a probability distribution: it systematically generates and evaluates every possible state in the sample space. For a discrete distribution over N states, this means explicitly listing all N configurations, computing their probabilities, and either selecting samples according to their weights or simply iterating through all possibilities.

While conceptually simple and guaranteed to be exact, brute force methods are only feasible for very small systems. The time and memory complexity scale as $\mathcal{O}(N)$, where N is the total number of possible states. For problems involving n binary variables, $N = 2^n$, so the cost grows exponentially with n—a classic example of the "curse of dimensionality" or combinatorial explosion, resulting in our case $\mathcal{O}(2^n)$ [22].

Brute force enumeration is sometimes used as a baseline for validating other sampling algorithms or for very small models. However, for most real-world applications, the exponential scaling renders brute force approaches intractable. Even modest increases in n quickly make the approach impractical, motivating the use of more sophisticated methods such as rejection sampling, MCMC, or quantum algorithms.

5.1.2 Rejection Sampling

Rejection sampling is a fundamental technique for generating samples from a target distribution $f(x)$ when direct sampling is impractical. The method uses a proposal distribution $g(x)$, from which sampling is easy, and accepts or rejects each candidate based on the ratio $f(x)/(Mg(x))$, where M is a constant such that $M \geq \sup_x p(x)/q(x)$ for all x. The efficiency of rejection sampling heavily depends on the choice of the proposal distribution; if $g(x)$ poorly approximates $f(x)$, the acceptance rate drops and the method becomes computationally inefficient [23]. The coice of said proposal distribution will define the M value, from which a suitable acceptance rate can be deduced. Despite its simplicity, rejection sampling can be wasteful for high-dimensional or rare-event scenarios, as the expected number of trials to obtain one valid sample scales as $\mathcal{O}(M)$. In rare events, where P_g – the probability of a "good" state – is low, matched with a proposal distribution resulting in a high M value $\approx \frac{1}{P_g}$, resulting in a computational complexity of $\mathcal{O}(\frac{1}{P_g})$, which just like with brute-force algorithms, is becoming prohibitive for rare events ($P_g \ll 1$). With an easy-to-sample distribution, or a suitable proposal one, the value for M will be much lower, and the acceptance rate higher. This means in the end a well built rejection sampling can achieve over the previously defined complexity, albeit only by a constant multiplier.

5.1.3 Markov Chain Monte Carlo (MCMC)

MCMC methods, such as the Metropolis-Hastings algorithm and Gibbs sampling, construct a Markov chain whose stationary distribution is the target distribution [24]. By sequentially generating correlated samples, MCMC can explore complex, high-dimensional distributions even when direct sampling or rejection sampling is infeasible. The key advantage of MCMC is its flexibility: it only requires the ability to compute the (unnormalized) density of the target distribution. However, MCMC methods can suffer from slow mixing, especially in multimodal or high-dimensional spaces, and require careful tuning to ensure convergence and independence of samples.
MCMC constructs a Markov chain converging to $p(x)$. For Gibbs sampling, with N being the number of possible states:

- Mixing time scales as $\mathcal{O}(e^N)$ for multimodal distributions
- Per-iteration cost: $\mathcal{O}(N)$ for local updates
- Suffers from slow mixing in high dimensions due to metastability

So while classical sampling methods such as brute-force methods, rejection sampling and MCMC are widely used and well-understood, neither method achieves better than linear scaling in $1/P_g$ or exponential in N, thus they face significant challenges in the context of high-dimensional, structured, or rare-event distributions.

5.2 Quantum vs. Classical: Asymptotic Speedup

Quantum amplitude amplification achieves a quadratic improvement, requiring only $\mathcal{O}(1/\sqrt{p_g})$ queries. This follows from Grover's theorem and its generalization to amplitude amplification:

Theorem 5.2.1 (Brassard et al. 2000). *Given an initial state $A|0\rangle$ with good-state probability $p_g = \sin^2\alpha$, after $k = \lfloor \pi/(4\alpha) \rfloor$ iterations of $Q = -AS_0A^\dagger S_\chi$, the probability of measuring a good state satisfies:*

$$P_{good} = \sin^2((2k+1)\alpha) \geq 1 - p_g \tag{5.1}$$

Proof. The state evolution under Q performs a rotation in a 2D subspace spanned by $|\psi_g\rangle$ and $|\psi_b\rangle$ 3.2.3. After k iterations:

$$Q^k A|0\rangle = \sin((2k+1)\alpha)|\psi_g\rangle + \cos((2k+1)\alpha)|\psi_b\rangle$$

Choosing $k = \lfloor \pi/(4\alpha) \rfloor$ yields $(2k+1)\alpha \in [\pi/2-\alpha, \pi/2+\alpha]$, giving $P_{\text{good}} \geq \cos^2\alpha = 1 - p_g$. □

This demonstrates that $\mathcal{O}(1/\sqrt{p_g})$ iterations suffice to amplify the success probability near 1, quadratically faster than classical rejection sampling.

Full Circuit Depth Analysis Although we see that the amplitude amplification achieves a quadratic improvement over classical algorithms, it is only the 'search' part of our procedure, and the quantum advantage must account for full circuit depth, not just oracle queries. Thus we need to include the state preparation unitaries as well:

- D_A: Depth of state preparation circuit
- D_O: Depth of oracle implementation
- D_D: Depth of diffusion operator

The total depth for k iterations is:

$$D_{\text{total}} = D_A + k(D_O + D_D) \tag{5.2}$$

Which in my implementation sums up to:

- Uniform preparation: $D_A = \mathcal{O}(n)$ (Hadamard gates + NOT/Toffoli gates for logical formula mapping)
- Logical oracle: $D_O = \mathcal{O}(1)$ (utilizing the symmetry between D_A and D_D)
- Diffusion: $D_D = \mathcal{O}(n)$ (essentially overlapping with the state preparation unitary)

Thus $D_{\text{total}} = \mathcal{O}(n/\sqrt{p_g})$, still linear in variables, preserving quadratic speedup.

Bibliography

[1] Maximilian Balthasar Mansky et al. *Sampling problems on a Quantum Computer.* Sept. 2023. DOI: 10.1109/qce57702.2023.00062. URL: http://dx.doi.org/10.1109/QCE57702.2023.00062.

[2] Kevin Mallinger Sebastian Raubitzek. *On the Applicability of Quantum Machine Learning.* June 2023. DOI: 10.3390/e25070992. URL: https://pubmed.ncbi.nlm.nih.gov/37509939/.

[3] Masahiro Suzuki and Yutaka Matsuo. »A survey of multimodal deep generative models«. In: *Advanced Robotics* 36.5–6 (Feb. 2022), pp. 261–278. ISSN: 1568-5535. DOI: 10.1080/01691864.2022.2035253. URL: http://dx.doi.org/10.1080/01691864.2022.2035253.

[4] Ryan LaRose. *A brief history of quantum vs classical computational advantage.* 2024. arXiv: 2412.14703 [quant-ph]. URL: https://arxiv.org/abs/2412.14703.

[5] Blake A. Wilson, Zhaxylyk A. Kudyshev and Kildishev. »Machine learning framework for quantum sampling of highly constrained, continuous optimization problems«. In: *Applied Physics Reviews* 8.4 (Dec. 2021). ISSN: 1931-9401. DOI: 10.1063/5.0060481. URL: http://dx.doi.org/10.1063/5.0060481.

[6] Lov K. Grover. *A fast quantum mechanical algorithm for database search.* 1996. arXiv: quant-ph/9605043 [quant-ph]. URL: https://arxiv.org/abs/quant-ph/9605043.

[7] IQM. »Demonstration of Improved Qubit Quality«. In: *Quantum Computing Report* (2024). URL: https://quantumcomputingreport.com/iqm-demonstrates-continued-improvement-in-qubit-quality/.

[8] Moody's. »Quantum Computing's Six Most Important Trends for 2025«. In: *Moody's Analytics* (2025). URL: https://www.moodys.com/web/en/us/insights/quantum/quantum-computings-six-most-important-trends-for-2025.html.

[9] SpinQ. »Quantum Hardware Challenges«. In: *SpinQ News* (2025). URL: https://www.spinquanta.com/news-detail/quantum-hardware-explained-a-complete-guide.

[10] Number Analytics. *Real-World Applications of Probabilistic Graphical Models.* 2025. URL: https://www.numberanalytics.com/blog/real-world-applications-probabilistic-graphical-models-uncovered-today.

[11] UC Berkeley. *Propositional Logic I.* 2024. URL: https://inst.eecs.berkeley.edu/~cs188/sp24/assets/lectures/cs188-sp24-lec07.pdf.

[12] Md Kamruzzaman Sarker et al. »Neuro-symbolic artificial intelligence: Current trends«. In: *AI Communications* 34.3 (2021), pp. 197–209. DOI: 10.3233/AIC-210084. URL: https://journals.sagepub.com/doi/abs/10.3233/AIC-210084.

[13] Martin J. Wainwright and Michael I. Jordan. »Graphical Models, Exponential Families, and Variational Inference«. In: *Foundations and Trends® in Machine Learning* 1.1–2 (2008), pp. 1–305. ISSN: 1935-8237. DOI: 10.1561/2200000001. URL: http://dx.doi.org/10.1561/2200000001.

[14] Dan Geiger et al. »Stratified exponential families: Graphical models and model selection«. In: *The Annals of Statistics* 29.2 (2001), pp. 505–529. DOI: 10.1214/aos/1009210550. URL: https://doi.org/10.1214/aos/1009210550.

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

[16] Douglas Youvan. *Beyond the Limit: Implications and Consequences of Violating the No-Communication Theorem.* Dec. 2023. DOI: 10.13140/RG.2.2.16756.32646.

[17] Gilles Brassard et al. *Quantum amplitude amplification and estimation.* 2002. DOI: 10.1090/conm/305/05215. URL: http://dx.doi.org/10.1090/conm/305/05215.

[18] Minzhao L. Aleksandr B. and Atithi A. *Tensor networks for quantum computing.* 2025. arXiv: 2503.08626 [quant-ph]. URL: https://arxiv.org/abs/2503.08626.

[19] Shi-Ju Ran. »Encoding of matrix product states into quantum circuits of one- and two-qubit gates«. In: *Physical Review A* 101.3 (Mar. 2020). ISSN: 2469-9934. DOI: 10.1103/physreva.101.032310. URL: http://dx.doi.org/10.1103/PhysRevA.101.032310.

[20] Frank L. Hitchcock. »The Expression of a Tensor or a Polyadic as a Sum of Products«. In: *Journal of Mathematics and Physics* 6.1-4 (1927), pp. 164–189. DOI: https://doi.org/10.1002/sapm192761164. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/sapm192761164.

[21] Benyamin Ghojogh et al. »Sampling Algorithms, from Survey Sampling to Monte Carlo Methods: Tutorial and Literature Review«. In: *arXiv preprint arXiv:2011.00901* (2020). URL: https://arxiv.org/abs/2011.00901.

[22] GFG. »Brute Force Approach and its pros and cons«. In: (2024). URL: `https://www.geeksforgeeks.org/dsa/brute-force-approach-and-its-pros-and-cons/`.

[23] Sarah Lee. »Rejection Sampling: A Deep Dive«. In: (2025). URL: `https://www.numberanalytics.com/blog/rejection-sampling-deep-dive`.

[24] Dani Gamerman and Hedibert F. Lopes. »A simple introduction to Markov Chain Monte–Carlo sampling«. In: *PLoS Computational Biology* (2016). URL: `https://pmc.ncbi.nlm.nih.gov/articles/PMC5862921/`.