

WORKING PAPER · NO. 2026-40

Quantum Bayesian Inference: An Exploration

Jon Frost, Carlos Madeira, Yash Rastogi and Harald Uhlig

MARCH 2026

Quantum Bayesian Inference: An Exploration

Jon Frost, Carlos Madeira, Yash Rastogi and Harald Uhlig*

First draft: July 21, 2025

This revision: March 9, 2026

*Jon Frost, Head of Innovation and the Digital Economy, Bank for International Settlements (BIS) and research affiliate, Cambridge Centre for Alternative Finance. Carlos Madeira, Central Bank of Chile. Yash Rastogi, Ph.D. Student, School of Mathematics, Georgia Institute of Technology. Harald Uhlig, Bruce Allen and Barbara Ritzenthaler Professor of Economics, The Kenneth C. Griffin Department of Economics, University of Chicago, 1126 East 59th Street, Chicago, IL 60637, U.S.A. email: huhlig@uchicago.edu. Affiliations: National Bureau of Economic Research (NBER) and Centre for Economic Policy Research (CEPR). With thanks to Raphael Auer, Dave Campbell and Danica Marsden for comments, and Rudraksh Kansal for research assistance. The views expressed here are those of the authors and do not necessarily reflect those of the BIS or of the Central Bank of Chile or its board members. All errors are our own.

Abstract

This paper introduces a framework for performing Bayesian inference using quantum computation. It presents a proof-of-concept quantum algorithm that performs posterior sampling. We provide an accessible introduction to quantum computation for economists and a practical demonstration of quantum-based posterior sampling for Bayesian estimation. Our key contribution is the preparation of a quantum state whose measurement yields samples from a discretized posterior distribution. While the proposed approach does not yet offer computational speedups over classical techniques such as Markov Chain Monte Carlo, it highlights both the conceptual promise and practical challenges in integrating quantum computation into the econometrician's toolbox.

Keywords: Quantum computing; Bayesian estimator; Bayesian inference; Markov chain Monte Carlo (MCMC) algorithms; Gibbs sampling

JEL codes: C11; C20; C30; C50; C60

1 Introduction

This paper explores the potential of quantum computation as a framework for Bayesian inference. Quantum bits (qubits) are probabilistic objects, making quantum computation conceptually aligned with Bayesian inference, where belief updates are driven by probabilistic rules. This conceptual parallel motivates our key idea: encoding a discretized posterior distribution over 2^n possible parameter values into the amplitudes of an n -qubit state, and using quantum measurement to sample from this distribution. In effect, we propose using quantum computation as a means for sampling from the posterior distribution in Bayesian inference.

In practice, quantum computation is a field largely driven forward by engineers and computer scientists, while statisticians and econometricians study Bayesian inference. Bridging this disciplinary divide requires translating core ideas across fields. To that end, we provide an accessible introduction to quantum computation tailored to economists, alongside a concise review of Bayesian inference. We then present a simple quantum workflow for posterior sampling and implement it using the Qiskit package in the Python programming language. Although our method does not currently offer computational advantages over classical techniques, our primary aim is conceptual: to demonstrate the feasibility of using quantum computation to perform Bayesian inference and lay a foundation for future algorithmic innovations that may realize quantum speedups.

Despite the early excitement surrounding quantum computation, the field remains in its infancy when it comes to solving computationally intense problems at scale. Progress has been constrained by significant engineering challenges, including qubit decoherence, error correction, and hardware scalability. Many quantum algorithms offer exponential speedups over classical algorithms in theory but have not yet been demonstrated at scale in practice

due to the current limitations in quantum hardware.¹ For example, number factorization using Shor’s algorithm so far has been limited to small numbers, since quantum computing remains noisy and only operates at intermediate scale (Martín-López et al. 2012 [17]).² Only recently has a quantum computer been built that is able to run for longer than 2 hours (Chiu et al. 2025 [7]).

Some critics have questioned whether quantum advantage will ever be achieved at scale due to the computational requirements for noise-reduction as exercises scale up (Kalai 2020 [14]). Even optimistic assessments suggest that demonstrable speedups may remain elusive for applied problems of economic relevance (Hoeffler et al. 2023 [13]).

Despite these diverging views, however, some quantum computing developers believe that significant advances will be made for the study of molecular reactions, materials research and maybe for some managerial processes such as logistics optimization (Brooks 2023 [4]). Some teams are exploring newer quantum computing methods and hardware, including the combination of quantum and classical computing approaches. Financial applications are also gaining attention, with recent studies identifying promising use cases in risk management, investment and portfolio optimization, payments and settlement, rare disasters and dynamic asset pricing, and volatility modeling

¹A number of firms have developed quantum processors with varying numbers of qubits. However, there are a number of challenges: they generally have high error rates; require very low temperatures (near absolute zero), which is very energy-intensive; require frequent restarts (with operation times lower than 2 hours until a recent breakthrough by a team at Harvard); and they do not have enough qubits for commercially viable operations. This remains an area of active research and development.

²The Noisy Intermediate-Scale Quantum (NISQ) era refers to the current stage of quantum computing, characterized by devices with a moderate number of qubits (tens to a few hundred) that are prone to significant noise and errors, limiting their ability to achieve true quantum advantage.

(Acharya et al. 2024 [1], Auer et al. 2024 [2], McMahon et al. 2024 [18], Ghysels et al. 2025 [10] and Ghysels and Morgan 2025 [11]). Recent work has also shown advances in terms of applying quantum annealers to dynamic programming problems, achieving order-of-magnitude speedups relative to benchmarks in solving real business cycle models (Fernandez and Hull 2023 [9]). Quantum annealers are also getting further on number factorization (Ding 2024 [8]). Some quantum Markov chain Monte Carlo (MCMC) algorithms seem to converge in fewer iterations than classical approaches (Layden 2023 [16]). Furthermore, there is a need to start to prepare the digital infrastructures of financial institutions for quantum-safe cryptography (Auer et al. 2025 [3]).

Nonetheless, now is an opportune moment to explore the theoretical capabilities of quantum computing, particularly in domains like Bayesian inference where probabilistic reasoning is central. Our aim is to contribute to that foundational understanding, in anticipation of future advances that may eventually unlock the practical potential of quantum computation. Our hope is that future research will build on this groundwork to develop practical, efficient quantum computing tools for econometric analysis.

2 A brief introduction to quantum computing

The purpose of this section is to provide a brief introduction to quantum computing, including widely used concepts, notations, and ideas. We focus on the mathematical and computer science description rather than the underlying physics.

The basis for all classical computation is a bit, which can take one of the two values 0 or 1. In quantum computation, the Dirac or bra-ket notation is

often used. This entails writing the value 0 as $|0\rangle$ and the value 1 as $|1\rangle$ or, alternatively, as the two-dimensional vectors $\begin{bmatrix} 1 & 0 \end{bmatrix}'$ and $\begin{bmatrix} 0 & 1 \end{bmatrix}'$. As a first pass, it might be helpful to think of a qubit as assigning a probability p to $|0\rangle$ and the probability $1-p$ to $|1\rangle$, with either outcome realized with these probabilities, once the qubit is observed or read out.³ Matters are slightly more complicated, however. The probabilities p and $1-p$ are actually parameterized by two complex-valued numbers α and β with $p = |\alpha|^2$ and $1-p = |\beta|^2$. This, in turn, leads to parameterizing a qubit with **Hopf coordinates** $[\nu, \phi, \delta]'$ with $\nu, \phi, \delta \in [0, 2\pi]$ and $\alpha = e^{i\delta} \cos(\nu/2)$, $\beta = e^{i(\delta+\phi)} \sin(\nu/2)$. The **global phase** $\delta \in [0, 2\pi]$ has no physically observable consequences and can therefore be normalized to equal 0. The parameterization then becomes $\alpha = \cos(\nu/2)$ and $\beta = e^{i\phi} \sin(\nu/2)$, with ϕ called or denoting the **relative phase**. This is written by introducing ψ as the state of the qubit,

$$\begin{aligned}
 |\psi\rangle &= \alpha|0\rangle + \beta|1\rangle \\
 &= e^{i\delta} \left(\cos\frac{\nu}{2} |0\rangle + e^{i\phi} \sin\frac{\nu}{2} |1\rangle \right) \\
 &= \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \\
 &= e^{i\delta} \begin{bmatrix} \cos\frac{\nu}{2} \\ e^{i\phi} \sin\frac{\nu}{2} \end{bmatrix}
 \end{aligned} \tag{1}$$

where the first line uses the ket notation and the second line the vector notation. Viewed from this perspective and depending on the notation, a qubit is $[\nu, \phi]'$, an element in $[0, 2\pi] \times [0, 2\pi] \subset \mathbb{R}^2$ (excluding the irrelevant

³Another way of thinking of this is that a classical bit is like a coin lying flat that is either heads or tails. By contrast, a quantum state is like the coin spinning in the air; there is a probability of both heads and tails, until the observers catches and looks at it (in which case measurement collapses it into either heads or tails). This metaphor helps to understand one major quantum phenomenon, namely superposition.

δ) and thus an element of the real two-dimensional space, or is the vector $[\alpha, \beta]' \in \mathbb{B}_2$. In this case, the unit sphere \mathbb{B}_2 is the set of all vectors $[\alpha, \beta]' \in \mathbb{C}^2$ with the property $\|[\alpha, \beta]'\| = |\alpha|^2 + |\beta|^2 = 1$. In short, a bit encodes one of the two numbers 0 or 1. A qubit encodes an element on a two-dimensional manifold. This is a remarkable difference.

Operators on a qubit are linear mappings $T : \mathbb{B}_2 \rightarrow \mathbb{B}_2$, called **quantum logic gates** or simply **quantum gates**. These mappings are **unitary operators** or unitary 2×2 complex-valued matrices satisfying $T^*T = TT^* = I_2$, where T^* is the adjoint (or complex conjugate transpose) of T and I_2 is the 2×2 identity matrix, since unitary operators preserve length in \mathbb{C}^2 . Various quantum gates have particular names and sometimes are analogues to operations on classic bits, including:

$$\begin{aligned}
 \text{Pauli-X: } & \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \\
 \text{Pauli-Y: } & \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \\
 \text{Pauli-Z: } & \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \\
 \text{Hadamard } H &: \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}
 \end{aligned} \tag{2}$$

This generalizes to higher dimensions. In two dimensions, two classical bits can take the values 00, 01, 10, or 11. A two-qubit is then

$$\begin{aligned}
 |\psi\rangle &= v_{00}|00\rangle + v_{01}|01\rangle + v_{10}|10\rangle + v_{11}|11\rangle \\
 &= \begin{bmatrix} v_{00} \\ v_{01} \\ v_{10} \\ v_{11} \end{bmatrix}
 \end{aligned} \tag{3}$$

Here, $[v_{00}, v_{01}, v_{10}, v_{11}]' \in \mathbb{B}_4$, where the unit sphere \mathbb{B}_4 is the set of all vectors $[v_{00}, v_{01}, v_{10}, v_{11}]' \in \mathbb{C}^4$ with the property $\sum_{i,j \in \{0,1\}} |v_{ij}|^2 = 1$.⁴ Operators on a two-qubit are linear mappings $T : \mathbb{B}_4 \rightarrow \mathbb{B}_4$, i.e. complex-valued unitary 4×4 matrices.

An n-qubit is an element of \mathbb{B}_{2^n} , where the unit sphere \mathbb{B}_{2^n} is the set of all vectors in \mathbb{C}^{2^n} with the property that the sum of the squared absolute values equals 1.⁵ Operators on an n-qubit are linear mappings $T : \mathbb{B}_{2^n} \rightarrow \mathbb{B}_{2^n}$, i.e. complex-valued unitary $2^n \times 2^n$ matrices. In short, while n bits encode one of 2^n numbers $j \in \{0, \dots, 2^n - 1\}$, an n-qubit encodes a 2^n -dimensional vector of complex numbers with a length restriction,⁶ and, possibly, the normalization of setting the global phase to zero, $\delta = 0$.⁷ Counting dimensions and recognizing \mathbb{C} as a two-dimensional real vector space, the manifold $\mathbb{B}_{2^n}^0$, i.e. all elements of \mathbb{B}_{2^n} with the first element real-valued, i.e. with $\delta = 0$, is of real dimension $2^{n+1} - 2$.⁸

Quantum computation on an n-qubit amounts to applying operators and,

⁴This is a standard representation of a two-qubit state. The basis states $(\begin{bmatrix} 0 & 0 \end{bmatrix})'$, $(\begin{bmatrix} 0 & 1 \end{bmatrix})'$, $(\begin{bmatrix} 1 & 0 \end{bmatrix})'$, and $(\begin{bmatrix} 1 & 1 \end{bmatrix})'$ form an orthonormal basis for the 4-dimensional Hilbert space describing the two-qubit system. The coefficients $(v_{00}), (v_{01}), (v_{10}),$ and (v_{11}) are complex numbers.

⁵The condition $(\sum_{i,j \in \{0,1\}} |v_{ij}|^2 = 1)$ is the normalization condition for a quantum state. It ensures that the state vector has unit length, which is a fundamental requirement in quantum mechanics. This normalization is what makes the vector lie on the unit sphere (\mathbb{B}_4).

⁶This implies the constraint that the sum of the squared magnitudes of its components equals 1.

⁷This implies fixing the phase of the first element, imposing the constraint that the first element is real-valued.

⁸Start with (2^{n+1}) real dimensions (from the complex vector space (\mathbb{C}^{2^n})). Subtract one for the normalization constraint $((\sum |v_i|^2 = 1))$. Subtract one for setting the phase of the first element to zero (making it real-valued). This results in a real dimension of $(2^{n+1} - 2)$.

eventually, reading out the result, where the n -qubit collapses to one of the 2^n values $j \in \{0, \dots, 2^n - 1\}$ represented by n classic bits with probability $|v_{ij}|^2$. In fact, the potential of quantum computing arises from this exponential growth of the state space. At the heart of our approach is encoding a probability distribution into a quantum state. Since measurement outcomes of a quantum system are the squares of amplitude magnitudes, we represent the posterior probabilities using the squared amplitudes of an n -qubit state. Our objective of encoding a classically computed posterior as a quantum state requires initializing a quantum register to a classical bitstring. We demonstrate this below (in listing 1) using the Qiskit package in the Python programming language.⁹

3 Bayesian inference

In this section, we review the fundamentals of Bayesian inference and establish notation for the remainder of the paper. In Bayesian inference, the goal is to learn about some unknown parameter $\theta \in \Theta$ where Θ is a parameter space equipped with a density measure $d\theta$. The researcher's prior beliefs about θ are encoded in a probability density $\pi(\theta)d\theta$ over Θ . An experiment yields data and provides a likelihood function $L(\theta)d\theta$, representing how likely the observed data are for different values of θ . Bayes' formula then provides the posterior density $\tilde{\pi}(\theta)d\theta$, where $\tilde{\pi}(\theta) \propto \pi(\theta)L(\theta)$, and the constant of proportionality ensures that $\tilde{\pi}$ integrates to one. Given the posterior (even if known only up to a constant), a central object of interest is the expectation of a function f under the posterior, $E_{\tilde{\pi}}[f] = \int f(\theta)\tilde{\pi}(\theta)d\theta$. For instance, choosing $f(\theta) = \theta$ yields the posterior mean $E_{\tilde{\pi}}[\theta]$.

⁹Qiskit code can be executed in various environments, including online platforms such as Google Colab (<https://colab.google/>), local Python environments on a personal computer, or on IBM Quantum hardware via the IBM Quantum Platform.

As a simple case, suppose the parameter space is finite: $\Theta = \{\theta_0, \dots, \theta_{k-1}\}$. The prior π assigns probabilities $\pi(\theta_j) = P(\theta = \theta_j)$, and the likelihood function is given by $L(\theta_j | X) = P(X = X_j | \theta = \theta_j)$, where $X_j \in \{X_1, \dots, X_n\}$ represents observed data. Bayes' formula then delivers the posterior probability weights

$$\begin{aligned} \tilde{\pi}(\theta_j) &= \tilde{P}(\theta = \theta_j) \\ &= \frac{L(\theta_j | X) \pi(\theta_j)}{\sum_{i=0}^{k-1} L(\theta_i | X) \pi(\theta_i)} \end{aligned} \tag{4}$$

For a function $f : \{\theta_0, \dots, \theta_{k-1}\} \rightarrow \mathbb{R}$, the posterior expectation becomes $E_{\tilde{\pi}}[f] = \sum_{j=0}^{k-1} f(\theta_j) \tilde{\pi}(\theta_j)$. A finite state space of size k can be encoded using $n \geq \log_2 k$ qubits, making this formulation amenable to quantum sampling.

Next, consider the continuous case where $\Theta = [0, 1]$ and π, L and $\tilde{\pi}$ are densities over $[0, 1]$. As before, the posterior density is defined (up to normalization) by $\tilde{\pi}(\theta) \propto \pi(\theta)L(\theta)$ with the proportionality constant chosen so that $\int_0^1 \tilde{\pi}(\theta) d\theta = 1$.¹⁰ From a computational perspective, the discrete case can be viewed as a histogram approximation of the continuous setting. Specifically, we can partition $[0, 1]$ into k equally sized bins $[(j-1)/k, j/k]$, $j = 1, \dots, k$ (ignoring the overlaps at the boundaries). We can then pursue inference for functions on $[0, 1]$ by approximating it with a grid and histogram and choosing an ever higher number of grid points k .¹¹

¹⁰This setting is quite general. As is well known, any Polish space \mathcal{P} equipped with its Borel σ algebra and a non-atomic probability measure μ admits a bijective, both-direction measurable and measure-preserving map $\Phi : \mathcal{P} \rightarrow [0, 1]$, where the Lebesgue measure is used on the Borel sets of $[0, 1]$. This result, often referred to as the Polish space isomorphism theorem or the measure isomorphism theorem for Polish spaces (Kechris 1995 [15]), implies that the unit interval with Lebesgue measure effectively represents any atomless probability space encountered in practice. The inclusion of finitely many atoms is straightforward, and in practice, countably infinite atoms can be handled by introducing a cutoff.

¹¹Since $[0, 1]$ with its uniform measure represents nearly all atomless measure spaces one

4 Quantum-computing Bayesian inference

Given a prior distribution π , a likelihood function L and a function f , we aim to calculate $E_{\tilde{\pi}}[f]$ using quantum computation. We assume the parameter space is $\Theta = \{\theta_0, \dots, \theta_{k-1}\}$, where $k = 2^n$ for some integer n . Note that each number $j \in \{0, \dots, k-1\}$ can be represented using n bits.

We implement Bayesian inference by computing the posterior $\tilde{\pi}$ classically and encoding it as a quantum state v over n qubits. We then repeatedly prepare and measure this state and sum the results. Although this is the simplest Bayesian Monte-Carlo method, it allows arbitrary posterior distributions to be encoded via classical preprocessing.

As a first step, use classical computation to calculate the posterior directly: $\tilde{\pi}(\theta_j) = L(\theta_j)\pi(\theta_j)/\kappa$, $j = 0, \dots, k-1$, where the normalization constant is $\kappa = \sum_{j=0}^{k-1} L(\theta_j)\pi(\theta_j)$. Usually, in Bayesian analysis, one avoids calculating κ if possible. For the method here, we cannot avoid it because it is essential in constructing the quantum state.

Let $e_0 \in \mathbb{C}^k$ denote the standard basis vector $e_0 = [1, 0, 0, \dots, 0]'$. We aim to construct a unitary operator $T \in \mathbb{C}^{k \times k}$ such that the state $x = Te_0$ satisfies $|x_j|^2 = \tilde{\pi}(\theta_j)$ for all j . One way to construct such a T is as follows. Let $x_j = \sqrt{\tilde{\pi}(\theta_j)}$ for all j . This defines a real, normalized vector $x \in \mathbb{R}^k$. Set x as the first column of T , and complete the remaining $k-1$ columns using the Gram-Schmidt orthonormalization procedure, starting from the standard basis of \mathbb{R}^n . The resulting matrix T is real-valued, orthonormal (hence unitary), and satisfies $Te_0 = x$ by construction.¹²

may encounter in practice, as we have argued in the previous footnote, the k -state case can therefore be understood as a finite approximation to any practical Bayesian inference problem.

¹²While more efficient methods for quantum state preparation are known, we utilize the Gram-Schmidt procedure here because of its conceptual simplicity and suitability for NISQ-era implementation. For example, recursive methods for quantum state preparation

Now, to estimate the expectation $E_{\tilde{\pi}}[f]$ of a function f under the distribution $\tilde{\pi}$, proceed as follows:

Algorithm 1 Quantum Monte Carlo Estimation

Require: Discrete parameter set $\Theta = \{\theta_0, \dots, \theta_{k-1}\}$, where $k = 2^n$

Require: Function $f : \Theta \rightarrow \mathbb{R}$

Require: Unitary operator T acting on n qubits

Require: Number of samples N

- 1: Initialize running total: $S \leftarrow 0$
 - 2: **for** $i = 1, \dots, N$ **do**
 - 3: Prepare quantum register in the initial state: $\bar{v} \leftarrow e_0 = |0\rangle^{\otimes k}$
 - 4: Apply the unitary operator T : $v \leftarrow T\bar{v} = x$
 - 5: Measure v to obtain outcome $j \in \{0, \dots, k-1\}$, with probability $\tilde{\pi}(\theta_j)$
 - 6: Classically compute $f(\theta_j)$.
 - 7: Update the running total: $S \leftarrow S + f(\theta_j)$.
 - 8: **end for**
 - 9: **return** Estimated expectation: $\hat{\mu} = \frac{1}{N}S$
-

such as the one proposed by Grover and Rudolph for Monte Carlo integration (Grover and Rudolph 2002 [12]) construct the quantum state recursively by subdividing the domain of an efficiently integrable classical probability distribution, allowing for preparation of the state in logarithmic-depth circuits ($N = 2^n$). These methods scale to exponentially large state spaces and offer coherent amplitude encoding, which is valuable for achieving downstream quantum speedups. However, they involve substantial circuit overhead, including ancilla registers, controlled operations, and uncomputation steps, making them challenging to implement on current hardware. Similarly, efficient state preparation methods that utilize ancilla-based encoding of functions allow for expectation estimation but do not produce a coherent quantum state encoding of the posterior distribution (Vazquez and Woerner 2020 [5]). In contrast, the Gram-Schmidt-based methods provides a direct and flexible way to encode arbitrary posteriors with minimal circuit depth, making it well-suited for prototyping and near-term applications.

To see a review of applications of Markov chain Monte Carlo (MCMC) methods in economics such as Gibbs sampling in Bayesian estimation, please refer to Chib and Greenberg 1996 [6]. Furthermore, recent authors have proposed how to adapt traditional Bayesian simulation algorithms to quantum methods (Nikoloska and Simeone 2024 [19], Polson et. al. 2023 [20]). Our work improves on the basic formulation of the literature. Unlike Nikoloska 2024 [19], our approach directly encodes a discretized posterior distribution into the amplitudes of a quantum state.¹³ While Nikoloska’s work integrates quantum simulation into a broader Bayesian inference pipeline, our method seeks to replace classical sampling entirely by using quantum measurement as a novel posterior sampling mechanism. This leads us to address the technical challenge of explicit quantum state preparation for known posteriors, in contrast to using PQCs purely as likelihood simulators. Polson et al. 2023 [20] focuses on embedding quantum computation into classical machine learning workflows, such as feature extraction, regression, and stochastic gradient descent. In contrast, our paper serves as a foundational proof of concept for quantum-native Bayesian inference via direct amplitude encoding and measurement-based sampling.

Listing 2 gives our implementation of quantum-assisted Monte Carlo integration.

5 Simulated quantum computing results

We now simulate quantum sampling on a classical computer with the Qiskit code, choosing $\pi(\theta)$ to be uniform over $\theta \in [-5, 5]$, $L(\theta) = e^{-\frac{1}{2}(\theta-2)^2}$ and $f(j) = j^2$. The simulation uses $n = 6$ qubits and runs $N = 10000$ shots. Listing 3 gives the code. Figure 1 gives the output of the code—probability

¹³Nikoloska 2024 [19] uses Parameterized Quantum Circuits (PQCs) as generative models for data $X \sim p(X|\theta)$ and relies upon simulation-based inference.

densities for different values from the quantum sampling.

6 Discussion

This paper presents a proof-of-concept algorithm for performing Bayesian inference on a quantum computer. The core idea is straightforward: represent a discretized posterior distribution using the amplitudes of a quantum state, then generate samples via repeated measurements. In this framework, quantum computing becomes a novel tool for posterior simulation—analogous in goal but fundamentally distinct in mechanism from classical methods such as Markov Chain Monte Carlo (MCMC), importance sampling, and particle filtering.

The algorithm in this paper leverages the intrinsically probabilistic behavior of quantum systems to simulate draws from a posterior distribution. Conceptually, it treats qubits as random variables whose measurement outcomes yield posterior samples. The key technical challenge is the construction of a unitary operator that transforms the initial quantum state (typically $|0\rangle^{\otimes k}$) into a target state encoding the posterior distribution. In our current implementation, we address this by manually designing a quantum circuit that encodes a known posterior over a low-dimensional parameter space. While this works for simple settings, extending the method to continuous or high-dimensional parameter spaces presents significant challenges. General-purpose quantum algorithms for efficient and scalable state preparation remain an active research area. Advances in this direction would be especially valuable for Bayesian inference tasks involving analytically intractable posteriors, where classical normalization and sampling strategies often break down or become computationally prohibitive.

We stress that our approach does not currently offer a computational speedup over classical algorithms. In fact, it is likely less efficient due to the

overhead of quantum state preparation and the limitations of near-term quantum hardware. However, we view this work as a stepping stone toward more practical quantum-computation-based methods for Bayesian inference. Future work could focus on approximate state preparation techniques—particularly those using hybrid classical-quantum methods such as variational quantum algorithms, which optimize a parametrized quantum circuit to approximate the target distribution. Alternative approaches might include efficient amplitude encoding, quantum rejection sampling, or Grover-inspired algorithms tailored for posterior sampling. In all of this, rigorous statistical benchmarking against classical Bayesian inference algorithms will be crucial for identifying quantum advantage.

7 Conclusions

This paper offers a first step toward rethinking Bayesian inference through the lens of quantum computation. While quantum computers are unlikely to replace classical tools in Bayesian inference in the near term, they offer a fundamentally different way to represent and sample from probability distributions. Our proof-of-concept advance, involving encoding a posterior into quantum amplitudes and generating samples via measurement, serves as an early conceptual step in the direction of exploiting the potential of quantum computing in Bayesian inference. As quantum hardware matures, we expect approaches like ours to inform the development of quantum-native inference techniques that could eventually outperform or complement classical Bayesian methods.

Given the novelty of this field, there are a number of fruitful avenues for future research. For instance, research could focus on developing quantum algorithms for high-dimensional optimization problems commonly found in econometrics, such as those used in generalized method of moments (GMM)

estimation. A promising direction is to evaluate whether quantum speedups in matrix inversion or sampling could offer practical advantages in solving large-scale economic models. Given the fundamentally different architecture of quantum systems, new algorithmic paradigms may be needed to adapt standard techniques like regression or time series modeling. Open-source implementations on quantum simulators (e.g., Qiskit) could help researchers test these ideas before scalable quantum hardware becomes widely available.

References

- [1] Acharya, A., Yalovetzky, R., Minssen, P., Chakrabarti, S., Shaydulin, R., Raymond, R., Sun, Y., Herman, D., Andrist, R. S., Salton, G., Schuetz, M. J. A., Katzgraber, H. G., & Pistoia, M. (2024). Decomposition pipeline for large-scale portfolio optimization with applications to near-term quantum computing [Version 2]. *arXiv preprint*. doi:10.48550/arXiv.2409.10301.
- [2] Auer, R., Dupont, A., Gambacorta, L., Park, J., Takahashi, K., & Valko, A. (2024). *Quantum computing and the financial system: Opportunities and risks* (BIS Papers No. 149). Basel, Switzerland: Bank for International Settlements.
- [3] Auer, R., Dodson, D., Dupont, A., Haghighi, M., Margaine, N., McCarthy, S., Valko, A., & Marsden, D. (2025). *Quantum-readiness for the financial system: A roadmap* (BIS Papers No. 158). Basel, Switzerland: Bank for International Settlements.
- [4] Brooks, M. (2023). Quantum computers: What are they good for? *Nature*, 617. (Spotlight, May 24, 2023).
- [5] Carrera Vazquez, A., & Woerner, S. (2020). Efficient state preparation for quantum amplitude estimation. *Physical Review Applied*, 15(3). doi:10.1103/PhysRevApplied.15.034027.
- [6] Chib, S., & Greenberg, E. (1996). Markov chain Monte Carlo simulation methods in econometrics. *Econometric Theory*, 12, 409–431.
- [7] Chiu, N.C., Trapp, E.C., Guo, J.N., Abobeih, M.H., Stewart, L.M., Hollerith, S., Stroganov, P.L., Kalinowski, M., Geim, A.A., Evered, S.J., Li, S.H., Lyu, X.J., Peters, L.M., Bluvstein, D., Wang, T.T.,

- Greiner, M., Vuletić, V., & Lukin, M.D. (2025). Continuous operation of a coherent 3,000-qubit system. *Nature*. Advance online publication. doi:10.1038/s41586-025-09596-6.
- [8] Ding, J., Spallitta, G., & Sebastiani, R. (2024). Effective prime factorization via quantum annealing by modular locally-structured embedding. *Scientific Reports*. Advance online publication. doi:10.1038/s41598-024-53708-7.
- [9] Fernandez-Villaverde, J., & Hull, I. (2023). Dynamic programming on a quantum annealer: Solving the RBC model. *CEPR Discussion Paper* No. 18190. London, UK: Centre for Economic Policy Research.
- [10] Ghysels, E., Morgan, J., & Mohammadbagherpoor, H. (2025). On quantum and quantum-inspired maximum likelihood estimation and filtering of stochastic volatility models. *SSRN Working Paper*. doi:10.2139/ssrn.5274549.
- [11] Ghysels, E., & Morgan, J. (2025). On quantum ambiguity and potential exponential computational speed-ups to solving dynamic asset pricing models. *International Economic Review*. Advance online publication. doi:10.1111/iere.70019.
- [12] Grover, L., & Rudolph, T. (2002). Creating superpositions that correspond to efficiently integrable probability distributions. *arXiv preprint* arXiv:quant-ph/0208112.
- [13] Hoefler, T., Häner, T., & Troyer, M. (2023). Disentangling hype from practicality: On realistically achieving quantum advantage: What are the promising applications to realize quantum advantage? *Communications of the ACM*, 66(5), 82–87. doi:10.1145/3571725.

- [14] Kalai, G. (2020). The argument against quantum computers, the quantum laws of nature, and Google’s supremacy claims laws. *arXiv preprint arXiv:2008.05188*.
- [15] Kechris, A. S. (1995). *Classical descriptive set theory*. New York, NY: Springer.
- [16] Layden, D., Mazzola, G., Mishmash, R. V., Motta, M., Wocjan, P., Kim, J.S. & Sheldon, S. (2023). Quantum-enhanced Markov chain Monte Carlo. *Nature*, *619*, 282–287. doi:10.1038/s41586-023-06095-4.
- [17] Martín-López, E., Laing, A., Lawson, T., Alvarez, R., Zhou, X., & O’Brien, J. (2012). Experimental realization of Shor’s quantum factoring algorithm using qubit recycling. *Nature Photonics*, *6*, 773–776. doi:10.1038/nphoton.2012.259.
- [18] McMahon, C., McGillivray, D., Desai, A., Rivadeneyra, F., Lam, J.-P., Lo, T., ... Skavysh, V. (2024). Improving the efficiency of payments systems using quantum computing. *Management Science*, *70*(10), 7325–7341. doi:10.1287/mnsc.2023.00314.
- [19] Nikoloska, I., & Simeone, O. (2024). An introduction to Bayesian simulation-based inference for quantum machine learning with examples. *Frontiers in Quantum Science and Technology*, *3*, 1394533. doi:10.3389/frqst.2024.1394533.
- [20] Polson, N., Sokolov, S., & Xu, J. (2023). Quantum Bayesian computation. *Applied Stochastic Models in Business and Industry*, *9*(6), 869–883. doi:10.1002/asmb.2807.

Figures

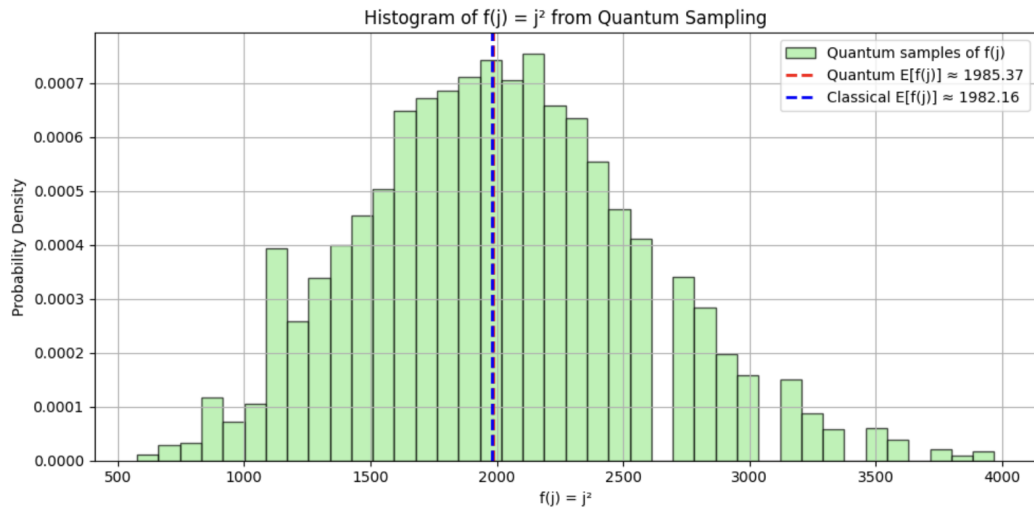


Figure 1: Histogram from quantum sampling. This is the output of the code in Listing 3.

Listings

Listing 1: Qiskit code to initialize an n -qubit state to a given classical bitstring

```
1 from qiskit_aer import AerSimulator # simulator backend for
   running quantum circuits on a classical hardware
2 from qiskit.circuit import QuantumCircuit # module for
   building quantum circuits
3 from qiskit.visualization import plot_histogram
4
5 def create_n_qubit_state(n, bitstring):
6     """
7     Create a quantum circuit that initializes an n-qubit
8     system
9     to a given classical bitstring state using X gates.
10
11     Parameters:
12         n (int): Number of qubits
13         bitstring (str): A binary string of length n
14
15     Returns:
16         QuantumCircuit: Circuit initialized to the given
17         bitstring
18     """
19     # Create a quantum circuit with n qubits and n classical
20     # bits (for measurement)
21     qc = QuantumCircuit(n, n)
22
23     # Apply X gates to qubits corresponding to '1' in the
24     # bitstring
25     for i in range(n):
26         if bitstring[i] == '1':
27             qc.x(i) # Flip qubit i from 0 to 1
```

```
25     qc.measure(range(n), range(n)) # Measure all qubits and
      store results in corresponding classical bits
26     return qc
```

Listing 2: Qiskit implementation of quantum-assisted Monte Carlo integration

```
1 from qiskit_aer import AerSimulator # simulator backend for
   running quantum circuits on a classical hardware
2 from qiskit.circuit import QuantumCircuit # module for
   building quantum circuits
3 import numpy as np
4
5 def quantum_monte_carlo(n, T, f, N):
6     """
7     Perform quantum-assisted Monte Carlo integration using a
8     quantum circuit.
9
10    Parameters:
11        n (int): Number of qubits
12        T (ndarray): Unitary matrix representing the state
13        preparation operator
14        f (Callable[[int], float]): Function to evaluate at
15        each sampled point
16        N (int): Number of measurement shots (samples)
17
18    Returns:
19        float: Estimated expected value of the function f
20        using quantum sampling
21    """
22
23    simulator = AerSimulator()
24    qc = QuantumCircuit(n, n)
25    qc.unitary(T, list(range(n)), label="T")
26    qc.measure(range(n), range(n))
27
28    result = simulator.run(qc, shots=N).result()
29    counts = result.get_counts()
30
31    # Compute expected value
32    total = 0
```

```
28     for bitstring, count in counts.items():
29         j = int(bitstring, 2) # Reverse for correct bit
           order
30         total += count * f(j)
31     return total / N
```

Listing 3: Full Quantum Monte Carlo Implementation and Comparison to Classically Computed Exact Value

```

1 from qiskit_aer import AerSimulator # simulator backend for
   running quantum circuits on a classical hardware
2 from qiskit.circuit import QuantumCircuit # module for
   building quantum circuits
3 import numpy as np
4 import matplotlib.pyplot as plt
5
6 # -----
7 # Step 1: Build T matrix
8 # -----
9 def find_T(theta_min, theta_max, n, prior, likelihood):
10     k = 2 ** n
11     theta_values = np.linspace(theta_min, theta_max, k)
12     prior_vals = np.vectorize(prior)(theta_values)
13     likelihood_vals = np.vectorize(likelihood)(theta_values)
14     posterior_values = likelihood_vals * prior_vals
15     posterior_values /= np.sum(posterior_values) # Normalize
16
17     x = np.sqrt(posterior_values)
18     # Create a full-rank matrix with x as the first column and
       random others
19     T = np.random.randn(k, k)
20     T[:, 0] = x
21
22     # Orthonormalize using QR (better numerical stability,
       faster speed, and lower complexity than manual Gram-
       Schmidt)
23     Q, _ = np.linalg.qr(T)
24     return Q, posterior_values
25
26 # -----
27 # Step 2: Quantum sampling
28 # -----

```

```

29 def quantum_monte_carlo(n, T, f, N):
30     simulator = AerSimulator()
31     qc = QuantumCircuit(n, n)
32     qc.unitary(T, list(range(n)), label="T")
33     qc.measure(range(n), range(n))
34
35     result = simulator.run(qc, shots=N).result()
36     counts = result.get_counts()
37
38     # Compute expected value
39     total = 0
40     for bitstring, count in counts.items():
41         j = int(bitstring, 2) # Reverse for correct bit
42             order
43         total += count * f(j)
44     expected = total / N
45     return expected, counts
46
47 # -----
48 # Step 3: Setup of example usage
49 # -----
50 theta_min = -5
51 theta_max = 5
52 n = 6
53 k = 2 ** n
54 N = 10000
55
56 def prior(theta):
57     return 1.0 # Uniform prior over the interval
58
59 def likelihood(theta):
60     return np.exp(-0.5 * (theta - 2) ** 2)
61
62 T = find_T(theta_min, theta_max, n, prior, likelihood)
63 f = lambda j: j**2

```

```

63
64 # -----
65 # Step 4: Run
66 # -----
67 T, posterior_values_normalized = find_T(theta_min, theta_max,
    n, prior, likelihood)
68 estimate, counts = quantum_monte_carlo(n, T, f, N)
69 print(f"Estimated E[f(j)] from quantum sampling = {estimate
    :.4f}")
70
71 expected_classical = np.sum([posterior_values_normalized[j] *
    f(j) for j in range(k)])
72 print(f"Analytical E[f(j)] from classical posterior = {
    expected_classical:.4f}")
73
74 # -----
75 # Step 5: Histogram of f(j)
76 # -----
77 dict_f = {f(int(bitstring, 2)): count for bitstring, count in
    counts.items()} # Convert each measured bitstring into an
    integer index j and then compute f(j) for each. Since
    different bitstrings can produce the same f(j), we
    aggregate the counts for each unique f(j) value. This
    gives the number of times each f(j) was observed across
    all quantum measurement shots.
78 x_vals = list(dict_f.keys()) # Extract the list of unique f(j
    ) values (to use as x-axis values in the histogram)
79 y_vals = list(dict_f.values()) # Extract the corresponding
    aggregated counts to use as y-axis values in the histogram
    )
80
81 plt.figure(figsize=(10, 5))
82 plt.hist(x_vals, weights=y_vals, bins=len(set(x_vals)),
    density=True,
83         color='lightgreen', edgecolor='black', alpha=0.7,

```

```

      label="Quantum_samples_of_f(j)")
84
85 # Overlay expectations
86 plt.axvline(estimate, color='red', linestyle='--', linewidth
      =2, label=f"Quantum_E[f(j)]={estimate:.2f}")
87 plt.axvline(expected_classical, color='blue', linestyle='--',
      linewidth=2, label=f"Classical_E[f(j)]={
      expected_classical:.2f}")
88
89 # Plot styling
90 plt.xlabel("f(j)=j^2")
91 plt.ylabel("Probability_Density")
92 plt.title("Histogram_of_f(j)=j^2_from_Quantum_Sampling")
93 plt.legend()
94 plt.grid(True)
95 plt.tight_layout()
96 plt.show()

```