Distributed Quantum Gaussian Processes for Multi-Agent Systems

Meet Gandhi
Colorado School of Mines
Golden, USA
meet_gandhi@mines.edu

George P. Kontoudis
Colorado School of Mines
Golden, USA
george.kontoudis@mines.edu

ABSTRACT

Gaussian Processes (GPs) are a powerful tool for probabilistic modeling, but their performance is often constrained in complex, large-scale real-world domains due to the limited expressivity of classical kernels. Quantum computing offers the potential to overcome this limitation by embedding data into exponentially large Hilbert spaces, capturing complex correlations that remain inaccessible to classical computing approaches. In this paper, we propose a Distributed Quantum Gaussian Process (DQGP) method in a multi-agent setting to enhance modeling capabilities and scalability. To address the challenging non-Euclidean optimization problem, we develop a Distributed consensus Riemannian Alternating Direction Method of Multipliers (DR-ADMM) algorithm that aggregates local agent models into a global model. We evaluate the efficacy of our method through numerical experiments conducted on a quantum simulator in classical hardware. We use real-world, non-stationary elevation datasets of NASA's Shuttle Radar Topography Mission and synthetic datasets generated by Quantum Gaussian Processes. Beyond modeling advantages, our framework highlights potential computational speedups that quantum hardware may provide, particularly in Gaussian processes and distributed optimization.

KEYWORDS

Gaussian Processes; Quantum Computing; Distributed Optimization; Riemannian ADMM; Multi-Agent Systems

ACM Reference Format:
Meet Gandhi and George P. Kontoudis. 2026. Distributed Quantum Gaussian Processes for Multi-Agent Systems. In *Proc. of the 25th Intern. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25–29, 2026*, IFAAMAS, 9 pages. https://doi.org/10.65109/ADPL7324

Code: github.com/mpala-lab/distributed-quantum-gaussian-processes

1 INTRODUCTION

Decision-making in autonomous systems relies on reliable uncertainty quantification. Gaussian processes (GPs), as an inherently probabilistic modeling technique, satisfy the need through accurate predictions and principled uncertainty estimation. To learn a GP model that characterizes the intrinsic dynamics of an unknown environment, an agent typically samples informative data points from the environment. However, training a GP model on N samples involves $O(N^3)$ computations and $O(N^2)$ memory. When a single agent is responsible for exploration and computation, the process becomes not only computationally demanding but also time-sensitive, as the agent must physically traverse all locations to gather samples. This limitation restricts the applicability of GPs to large-scale datasets and environments—conditions commonly encountered in autonomous systems—especially for single-agent systems. To this end, GP approximations have been introduced that can be broadly categorized into two main classes: exact aggregation methods and inducing point-based approximation methods [35]. We focus on the former class which serve as distributed GP (DGP) approaches [11, 23, 26]. These methods enable GP training on dataset sizes that would otherwise be infeasible for a single agent. In particular, they deploy multiple agents in local regions of the input space, allowing each to learn a local GP model that captures regional characteristics. The local GP models are then aggregated through multi-agent coordination to form a global GP model. FACT-GP [11] and its generalized version g-FACT-GP [34] enforce partitioning of sampled data, and the resulting local posteriors are subsequently aggregated. In addition, apx-GP [54] and gapx-GP [24] reach global consensus by using the multi-agent Alternating Direction Method of Multipliers (ADMM) [6]. By distributing both data storage and computational effort among agents, DGP methods effectively overcome the scalability limitations of standard GPs.

Gaussian Processes employ kernel functions [22] to model the correlations among the data points by projecting them into a high-dimensional feature space. This mapping enables GPs to capture complex relationships. However, the classical kernels possess limited expressivity due to the underlying mathematical formulation that is tractable on classical hardware. This shortcoming can be addressed through the emerging field of quantum computing. Our goal in this work is to leverage quantum computing to develop powerful and scalable GPs. Specifically, we aim to design a distributed framework for Quantum Gaussian Processes that exploits the expressive capability of quantum kernels while efficiently distributing the computational and memory load across multiple agents.

The current generation of quantum hardware, termed as the NISQ (Noisy Intermediate-Scale Quantum) era [43], lacks fault tolerance, making it challenging to realize a clear quantum advantage. This limitation has motivated the development of several hybrid quantum-classical techniques, termed as variational quantum algorithms (VQAs) [5]. VQAs employ parameterized quantum circuits [3], where each circuit parameter serves as an optimization variable adjusted to minimize a cost function. The system dynamics are modeled within the quantum domain, enabling faster gradient evaluation [50], while parameter updates are performed using classical optimizers. A notable example of VQA is the Quantum Approximate Optimization Algorithm (QAOA) [18], widely applied to combinatorial optimization problems. However, VQAs face several challenges, including the optimization landscape that often contains large, flat regions which cause gradients to vanish. This

Proc. of the 25th Intern. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25–29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/ADPL7324

phenomenon is called barren plateaus [30] and remains an active research topic [10, 38, 41, 42, 45]. Moreover, with the advancement of quantum hardware, exemplified by the milestone achievement of quantum supremacy [2], it becomes crucial to devise quantum algorithms that can be implemented in a parallelized and distributed fashion [39], where quantum circuit evaluations can be allocated to multiple quantum processors to enhance scalability and robustness.

A major advancement in quantum computing in recent years has been the introduction of quantum kernel functions [20], which have led to the development of Quantum Gaussian Processes (QGPs) [44]. Quantum kernels incorporate parameterized quantum circuits, called Quantum Encoding Circuits [21], to embed classical data into the quantum domain. Considerable research work has been conducted to designing kernels with the goal of achieving optimal alignment between quantum feature space and classical data labels. Beyond QGPs, these quantum kernels have proven beneficial in other domains of machine learning [1, 49]. In particular, [7, 40] employ quantum kernels with reinforcement learning in multi-agent settings. In addition to QGP training, significant efforts have been directed to leveraging quantum algorithms for GP inference. Numerous methods [8, 12, 14, 29, 56] have been formulated based on the Harrow-Hassidim-Lloyd (HHL) algorithm [36] to accelerate the computation of inverse kernel matrices required for prediction.

Contribution. The contribution of this paper is twofold. First, we introduce the distributed consensus Riemannian ADMM (DR-ADMM) optimization that can efficiently train parameterized quantum circuits across multiple agents. Next, we formulate the distributed quantum gaussian process (DQGP), which successfully scales the expressive power of quantum kernels utilizing quantum aspects of *entanglement*, *superposition* and *measurements* to handle complex, multi-agent scenarios (Fig. 1). Numerical experiments on non-stationary fields demonstrate the enhanced performance of the proposed method compared to other classical approaches.

2 PROBLEM FORMULATION

In this section, we discuss classical GPs, the function-space view of GPs that connects classical to quantum computing, quantum GP regression, distributed classical GP training, and state the problem.

2.1 Classical Gaussian Processes (GPs)

GP regression is a non-parametric Bayesian modeling approach [15] that provides a probability distribution over an infinite-dimensional space of functions. The system observations are modeled as $y(\boldsymbol{x}) = f(\boldsymbol{x}) + \epsilon$, where $\boldsymbol{x} \in \mathbb{R}^D$ is the input of a D dimensional space, $y \in \mathbb{R}$ is the corresponding label, $f(\boldsymbol{x}) \sim \mathcal{GP}(0, k(\boldsymbol{x}, \boldsymbol{x}'))$ is a zero-mean GP with covariance function $k : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$, and $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$ the i.i.d. zero mean Gaussian measurement noise with variance $\sigma_\epsilon^2 > 0$. The objective of GP regression then is to estimate the latent function f given dataset $\mathcal{D} = \{\boldsymbol{X}, \boldsymbol{y}\}$ with inputs $\boldsymbol{X} = \{\boldsymbol{x}_n\}_{n=1}^N$, outputs $\boldsymbol{y} = \{y_n\}_{n=1}^N$, where N is the number of observations.

We determine the hyperparameters $\boldsymbol{\theta}$ of the covariance function by using maximum likelihood estimation. The negative marginal log-likelihood function takes the form of,

$$\mathcal{L} = \log p(\boldsymbol{y} \,|\, \boldsymbol{X}) = \frac{1}{2}\Big(\boldsymbol{y}^\intercal \boldsymbol{C}_\theta^{-1} \boldsymbol{y} + \log |\boldsymbol{C}_\theta| + N \log(2\pi)\Big), \tag{1}$$

where $\boldsymbol{C}_\theta = \boldsymbol{K} + \sigma_\epsilon^2 \boldsymbol{I}_N$ is the positive definite covariance matrix and $\boldsymbol{K} = k(\boldsymbol{X}, \boldsymbol{X}) \succeq 0 \in \mathbb{R}^{N \times N}$ is the positive semi-definite correlation matrix between inputs of $\boldsymbol{X}$. The GP training problem yields,

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \boldsymbol{y}^\intercal \boldsymbol{C}_\theta^{-1} \boldsymbol{y} + \log |\boldsymbol{C}_\theta|, \tag{2}$$

which can be solved using gradient-based optimization methods with partial derivative of the objective,

$$\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \frac{1}{2} \mathrm{Tr}\left\{ \left(\boldsymbol{C}_\theta^{-1} - \boldsymbol{C}_\theta^{-1} \boldsymbol{y}\boldsymbol{y}^\intercal \boldsymbol{C}_\theta^{-1} \right) \frac{\partial \boldsymbol{C}_\theta}{\partial \boldsymbol{\theta}} \right\}. \tag{3}$$

2.2 Function-Space View of Gaussian Processes

The preceding discussion of GPs focuses on a formulation in a finite-dimensional space associated with a discrete dataset $\{\boldsymbol{X}, \boldsymbol{y}\}$. We can generalize to an infinite-dimensional function-space view, where a GP is defined as $f(\boldsymbol{x}) \sim \mathcal{GP}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}'))$ with $m(\boldsymbol{x})$ the mean function and $k(\boldsymbol{x}, \boldsymbol{x}')$ the covariance function [48]. In this way, a GP can be interpreted as a collection of random variables, any finite subset of which follows a joint Gaussian distribution. This allows the GP to assign probabilities over the space of possible functions, governed by the prior induced by the covariance. The function-space view provides a critical link to the kernel function κ through the associated feature map ϕ, expressed as,

$$\kappa(\boldsymbol{x}, \boldsymbol{x}') = \phi(\boldsymbol{x})^\intercal \phi(\boldsymbol{x}'). \tag{4}$$

In classical computing settings, the kernel function measures the correlation between two data points in a high-dimensional feature space without explicitly mapping the data into that space. In the quantum domain, the classical data are encoded into quantum states within a *quantum Hilbert space* using *Quantum Encoding Circuits* [46]. The Hilbert space is exponentially large, and thus a classical computer would struggle even to represent the quantum states, let alone compute their inner product (4). *Quantum kernel functions* [47] enable the estimation of correlations between these encoded quantum states, capturing complex and highly non-linear relationships in the original classical data.

2.3 Quantum Gaussian Processes (QGPs)

The first step in constructing a Quantum Gaussian process (QGP) is to encode classical data into quantum states. This is achieved using Quantum encoding circuits which implement a mapping from a classical data vector $\boldsymbol{x} \in \mathbb{R}^D$ to a quantum state $|\psi_{\boldsymbol{x}}\rangle$ in quantum Hilbert space $\mathcal{H}$, given by $\Phi : \boldsymbol{x} \to |\psi_{\boldsymbol{x}}\rangle = U(\boldsymbol{x}, \boldsymbol{\theta}) \, |0\rangle^{\otimes q}$, where Φ is the encoding map, $U(\boldsymbol{x}, \boldsymbol{\theta})$ a unitary operator representing the entire quantum encoding circuit with $\boldsymbol{\theta}$ the trainable hyperparameters, and $|0\rangle^{\otimes q}$ the initial quantum state of a system comprising q qubits. The circuit $U(\boldsymbol{x}, \boldsymbol{\theta})$ is composed of quantum unitary logic gates [53], primarily rotational gates $\boldsymbol{R}_{\boldsymbol{X}}(\boldsymbol{\sigma}_{\mathrm{X}}, \theta_x)$, $\boldsymbol{R}_{\boldsymbol{Y}}(\boldsymbol{\sigma}_{\mathrm{Y}}, \theta_y)$, $\boldsymbol{R}_{\boldsymbol{Z}}(\boldsymbol{\sigma}_{\mathrm{Z}}, \theta_z)$ with 2×2 Pauli matrices ($\boldsymbol{\sigma}_{\mathrm{X}}, \boldsymbol{\sigma}_{\mathrm{Y}}, \boldsymbol{\sigma}_{\mathrm{Z}}$) and their controlled versions. Additional gates include the Hadamard **H**, Phase **P**, **CNOT**, and the **SWAP** gate. Moreover, $U(\boldsymbol{x}, \boldsymbol{\theta})$ can be ι-layered with an identical gate structure in each layer, i.e., $U(\boldsymbol{x}, \boldsymbol{\theta}) = U_{\text{final}} U_\iota(\boldsymbol{x}, \boldsymbol{\theta}_\iota) \ldots U_1(\boldsymbol{x}, \boldsymbol{\theta}_1)$.

After encoding the classical data vectors $\boldsymbol{x}$ and $\boldsymbol{x}'$ into the quantum states $|\psi_{\boldsymbol{x}}\rangle$ and $|\psi_{\boldsymbol{x}'}\rangle$ respectively, the quantum kernel function κ computes the correlation between the states which can then be used to populate the GP covariance matrix $\boldsymbol{C}_\theta(\boldsymbol{x}, \boldsymbol{x}')$. A representative quantum kernel is the fidelity kernel $\kappa_{\mathcal{F}}$ [55] derived from

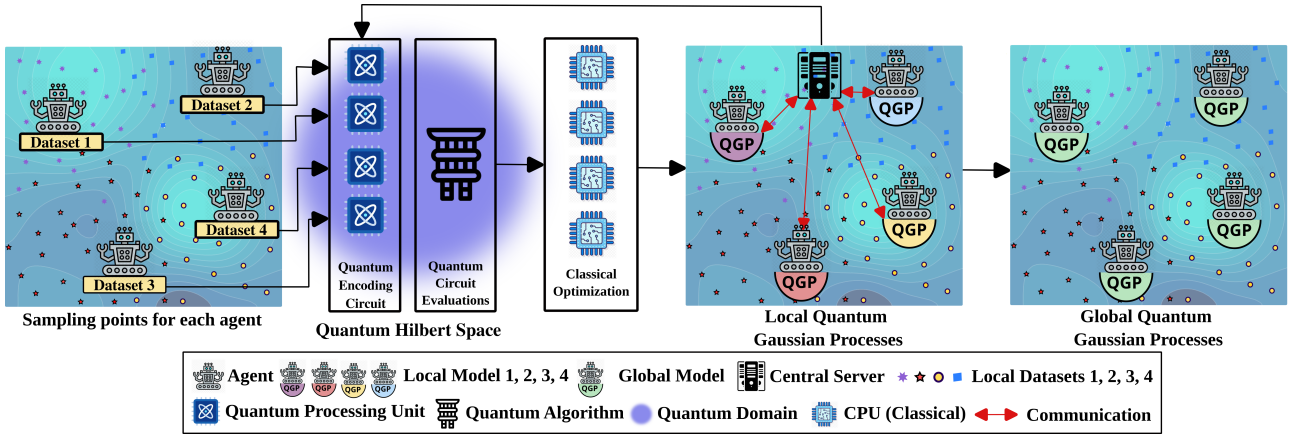

Figure 1: Distributed Quantum Gaussian Process (DQGP): A hybrid classical-quantum framework for multi-agent systems.

the fidelity measure $\mathcal{F}$,

$$\kappa_{\mathcal{F}} : \mathcal{H} \times \mathcal{H} \to [0,1] = |\langle \psi_{\boldsymbol{x}} | \psi_{\boldsymbol{x}'} \rangle|^2, \tag{5}$$

where $\kappa_{\mathcal{F}} = 1$ shows complete overlap between two quantum states and $\kappa_{\mathcal{F}} = 0$ indicates that the quantum states are orthogonal. There exists mathematical relevance between (4), (5), and the quantum kernels serving as covariance functions in QGP training.

2.4 Distributed Gaussian Processes (DGPs)

In classical GP training (2), each optimization round entails time complexity of $O(N^3)$, due to the computation of covariance matrix inverse $\boldsymbol{C}_{\boldsymbol{\theta}}^{-1}$. Additionally, storing $\boldsymbol{C}_{\boldsymbol{\theta}}^{-1}$ and N dataset size requires $O(N^2 + DN)$ space complexity. The high computational and memory demands make classical GPs impractical for large-scale, real-world applications. DGP [11] addresses the scalability bottleneck by distributing both computation and storage across multiple agents, under the assumption of local dataset independence.

Assumption 1. *All local datasets represent distinct areas with local models being statistically independent.*

Instead of computing the large covariance inverse $\boldsymbol{C}_{\boldsymbol{\theta}}^{-1}$, a DGP approximation yields, $\boldsymbol{C}_{\boldsymbol{\theta}}^{-1} \approx \text{diag}(\boldsymbol{C}_{\boldsymbol{\theta},1}^{-1}, \ldots, \boldsymbol{C}_{\boldsymbol{\theta},M}^{-1})$, where $\boldsymbol{C}_{\boldsymbol{\theta}}^{-1} \in \mathbb{R}^{N \times N}$ and $\boldsymbol{C}_{\boldsymbol{\theta},m}^{-1} \in \mathbb{R}^{N_m \times N_m}$ for all agents $m \in [1, M]$. Hence, by virtue of Assumption 1, DGPs can decompose the optimization problem over dataset $\mathcal{D}$ into a distributed optimization problem over local datasets $\mathcal{D}_m$. DGP training methods are also formulated using the multi-agent alternating direction method of multipliers (ADMM) [4, 6]. The analytical proximal GP (apx-GP) [54] training employs a first-order approximation on the local log-likelihood function $\mathcal{L}_m$ under the assumption of Lipschitz continuity,

Assumption 2. *The function $\mathcal{L}_m : \mathbb{R}^{\mathbb{N}} \to \mathbb{R}$ is Lipschitz continuous with a positive parameter $L > 0$ if,*

$$\|\nabla \mathcal{L}_m(\mathbf{x}) - \nabla \mathcal{L}_m(\mathbf{y})\|_2 \leq L \|\mathbf{x} - \mathbf{y}\|_2, \qquad \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^{\mathbb{N}}.$$

The main idea behind apx-GP and other ADMM-based GP training algorithms [25] is that every agent m is allowed to have an opinion on its hyperparameter vector $\boldsymbol{\theta}_m$; however, once the optimization is complete, they should agree on a global consensus vector $\boldsymbol{z}$. The optimization scheme of apx-GP yields,

$$\boldsymbol{z}^{(s+1)} = \frac{1}{M} \sum_{m=1}^{M} \left(\boldsymbol{\theta}_m^{(s)} + \frac{1}{\rho} \boldsymbol{\psi}_m^{(s)} \right) \tag{6a}$$

$$\boldsymbol{\theta}_m^{(s+1)} = \boldsymbol{z}^{(s+1)} - \frac{1}{\rho + L_m} \left(\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{z}^{(s+1)}) + \boldsymbol{\psi}_m^{(s)} \right) \tag{6b}$$

$$\boldsymbol{\psi}_m^{(s+1)} = \boldsymbol{\psi}_m^{(s)} + \rho \left(\boldsymbol{\theta}_m^{(s+1)} - \boldsymbol{z}^{(s+1)} \right) \tag{6c}$$

where $\rho > 0$ is the parameter promoting the consensus between all $\boldsymbol{\theta}_m$ and $\boldsymbol{z}$, L_m is a positive Lipschitz constant for each agent m, $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{z}^{(s+1)})$ is the gradient of marginal log-likelihood function with respect to $\boldsymbol{z}^{(s+1)}$, and $\boldsymbol{\psi}_m$ is the dual variable vector for each agent m. The reduced time and space complexity of apx-GP is $O(N_m) = O(N^3/M^3)$ and $O(N^2/M^2 + D(N/M))$ respectively.

2.5 Problem Statement

While DGPs effectively alleviate the scalability limitations of standard GPs, their predictive performance remains constrained by the limited expressivity of classical kernels. Quantum kernels, on the contrary, can capture subtle correlations that are inaccessible to classical kernels by leveraging the exponentially large Hilbert space for data mapping. Motivated by this, our work focuses on scaling QGPs to enable learning in multi-agent systems.

Problem 1. *Develop a **Distributed consensus Riemannian ADMM (DR-ADMM)** approach for optimizing quantum kernel hyperparameters across multiple agents.*

Problem 2. *Formulate a **Distributed Quantum Gaussian Process (DQGP)** algorithm that simultaneously overcomes the expressivity limitations of DGPs and the scalability challenges of QGPs.*

3 PROPOSED METHODOLOGY

In this section, we present the proposed methodologies Distributed consensus Riemannian ADMM (DR-ADMM) and Distributed Quantum Gaussian Process (DQGP).

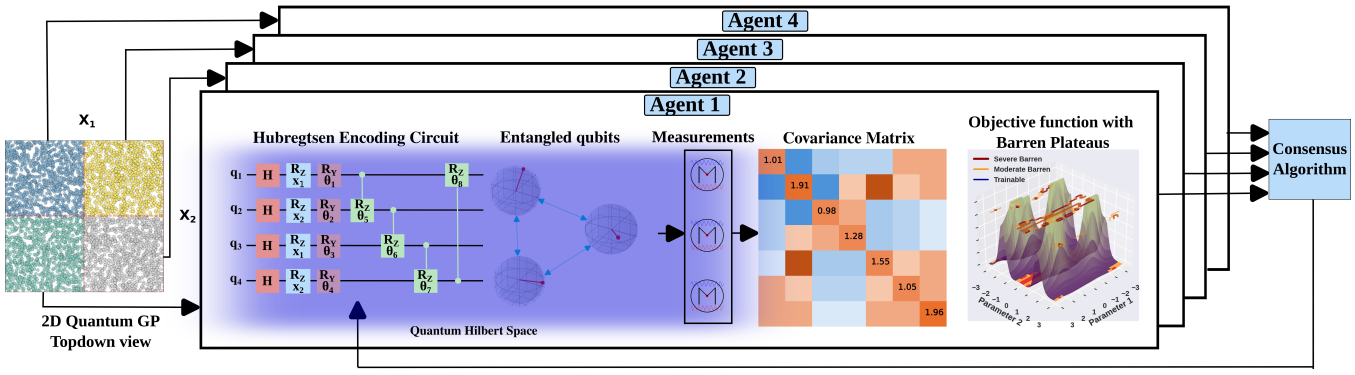

Figure 2: The structure of the proposed DQGP with 4 agents. The consensus algorithm is the proposed DR-ADMM optimization.

3.1 Distributed consensus Riemannian ADMM

The ADMM-based consensus problem (6) accounts for GP training with classical kernel hyperparameters that operate in a Euclidean parameter space. For quantum kernels, the hyperparameters are mostly rotational and thus lie in a more complex parameter space, a Riemannian manifold [31]. To address this, we first define the manifold in which the quantum hyperparameters exist and then describe the distributed consensus Riemannian ADMM algorithm. In [32], a Riemannian ADMM algorithm for solving nonconvex problems in centralized topologies is introduced.

3.1.1 Manifold Definition and Operations. The torus manifold for quantum circuit parameters is expressed as $\mathcal{T}^P = \mathcal{S}^1 \times \mathcal{S}^1 \times \cdots \times \mathcal{S}^1$, where $\mathcal{S}^1$ is the circle manifold for rotational parameters $\theta_p \in [0, \pi]$ for all $p = 1, 2, \ldots, P$. A combination of rotational quantum gates with parameters in $[0, \pi]$ ensures full-coverage of Bloch sphere [17]. Our choice of $[0, \pi]$ narrows the search space without sacrificing the expressiveness associated with the standard $[0, 2\pi]$ range. Subsequent operations can be constructed over the torus manifold,

$$\begin{aligned}
\textit{Manifold Projection}: &\ \Pi_{\mathcal{T}}(\theta) = \theta \bmod \pi, \\
\textit{Riemannian Distance}: &\ d_{\mathcal{T}}(\theta_p, \theta_r) = \|\mathcal{W}(\theta_p - \theta_r)\|_2, \\
\textit{Retraction}: &\ \mathcal{R}_{\theta_p}(\theta_r) = \Pi_{\mathcal{T}}(\theta_p + \theta_r), \\
\textit{Logarithmic Map}: &\ \mathrm{Log}_{\theta_p}(\theta_r) = \Pi_{\mathcal{T}}(\theta_r - \theta_p), \\
\textit{Vector Transport}: &\ \Gamma_{\theta_p \to \theta_r}(\boldsymbol{\nu}) = \boldsymbol{\nu}, \\
\textit{Inner Product}: &\ \langle \boldsymbol{\theta_p}, \boldsymbol{\theta_r} \rangle_{\mathcal{T}} = \langle \mathcal{W}(\boldsymbol{\theta_p}), \mathcal{W}(\boldsymbol{\theta_r}) \rangle,
\end{aligned}$$

where $\mathcal{W}(\theta) = [(\theta + \pi/2) \bmod \pi] - \pi/2$. Vector transport operation transports the tangent vector $\boldsymbol{\nu}$ from point θ_p to θ_r, which is an identity on a torus manifold. As the torus manifold is locally flat, Riemannian gradient is identical to the Euclidean gradient on a torus, i.e., $\mathrm{grad}_{\mathcal{T}} f(\theta) = \nabla f(\theta)$. Moreover, we assume,

Assumption 3. *All operations and gradients corresponding to the torus manifold are uniformly bounded.*

3.1.2 Quantum Gaussian Process Loss Function. Let us recast the negative marginal log-likelihood function (1) for QGPs as,

$$\mathcal{L}_{\mathrm{Q}}(\boldsymbol{\theta}) = \frac{1}{2}\Big(\boldsymbol{y}^{\intercal} \boldsymbol{\kappa}_{\mathrm{Q}}^{-1}(\boldsymbol{X}, \boldsymbol{X}|\boldsymbol{\theta})\boldsymbol{y} + \log|\boldsymbol{\kappa}_{\mathrm{Q}}(\boldsymbol{X}, \boldsymbol{X}|\boldsymbol{\theta})| + N\log(2\pi)\Big),$$

where $\boldsymbol{\kappa}_{\mathrm{Q}}(\boldsymbol{X}, \boldsymbol{X}|\boldsymbol{\theta})$ is the quantum kernel matrix with intrinsic measurement noise. The derivatives of the quantum kernel with respect to the individual hyperparameter are computed using the *parameter shift rule* [50], as the quantum gates are mostly rotational,

$$\frac{\partial[\boldsymbol{\kappa}_{\mathrm{Q}}]_{ij}}{\partial\theta_p} = \frac{[\boldsymbol{\kappa}_{\mathrm{Q}}]_{ij}(\boldsymbol{\theta} + \delta\boldsymbol{e}_p) - [\boldsymbol{\kappa}_{\mathrm{Q}}]_{ij}(\boldsymbol{\theta} - \delta\boldsymbol{e}_p)}{2\delta},$$

where δ is the shift value and $\boldsymbol{e}_p$ is the p-th unit vector. Then, from (3) the Quantum NLL loss gradient is described as,

$$\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathrm{Q}}(\boldsymbol{\theta}) = \frac{1}{2}\sum_{p=1}^{P}\mathrm{Tr}\left\{\left(\boldsymbol{\kappa}_{\mathrm{Q}}^{-1} - \boldsymbol{\kappa}_{\mathrm{Q}}^{-1}\boldsymbol{y}\boldsymbol{y}^{T}\boldsymbol{\kappa}_{\mathrm{Q}}^{-1}\right)\frac{\partial\boldsymbol{\kappa}_{\mathrm{Q}}}{\partial\theta_p}\right\}.$$

3.1.3 Distributed consensus Riemannian ADMM (DR-ADMM). The distributed optimization problem can be formulated as minimizing the sum of local QGP loss function across all agents, while enforcing a consensus constraint and assuming L_p-smooth cost functions, $\min_{\{\boldsymbol{\theta}_m\}} \sum_{m=1}^{M} \mathcal{L}_{\mathrm{Q},m}(\boldsymbol{\theta}_m)$ subject to $\boldsymbol{\theta}_m = \boldsymbol{z}$, $\forall m$.

Assumption 4. *$\mathcal{L}_{Q,m}$ is L_p-smooth, i.e., $\mathcal{L}_{Q,m}(\boldsymbol{z}^{(s+1)}) - \mathcal{L}_{Q,m}(\boldsymbol{z}^{(s)}) \le \langle \nabla\mathcal{L}_{Q,m}(\boldsymbol{z}^{(s)}), \boldsymbol{z}^{(s+1)} - \boldsymbol{z}^{(s)} \rangle_{\mathcal{T}} + \frac{L_p}{2}\mathrm{d}^2_{\mathcal{T}}(\boldsymbol{z}^{(s+1)}, \boldsymbol{z}^{(s)})$, $\forall m \in [1, M]$.*

To solve the distributed optimization problem, we construct an augmented Lagrangian function defined on a Riemannian manifold,

$$\mathbb{L}_{\rho}(\boldsymbol{\theta}_m, \boldsymbol{z}, \boldsymbol{\psi}_m) = \sum_{m=1}^{M}\left(\mathcal{L}_{\mathrm{Q},m}(\boldsymbol{\theta}_m) + \boldsymbol{\psi}_m^{\intercal}\mathrm{Log}_{\boldsymbol{z}}(\boldsymbol{\theta}_m) + \frac{\rho}{2}\|\mathrm{Log}_{\boldsymbol{z}}(\boldsymbol{\theta}_m)\|^2\right).$$

Subsequently, the optimization scheme of the DR-ADMM yields,

$$\boldsymbol{z}^{(s+1)} = \arg\min_{w \in \mathcal{T}^P} \sum_{m=1}^{M} d^2_{\mathcal{T}}\left(w, \boldsymbol{\theta}_m^{(s)} + \frac{\boldsymbol{\psi}_m^{(s)}}{\rho}\right) \tag{7a}$$

$$\boldsymbol{\theta}_m^{(s+1)} = \mathcal{R}_{\boldsymbol{z}^{(s+1)}}\left(-\frac{\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathrm{Q},m}\left(\boldsymbol{z}^{(s+1)}\right) + \boldsymbol{\psi}_m^{(s)}}{\rho + L_m}\right) \tag{7b}$$

$$\boldsymbol{\psi}_m^{(s+1)} = \boldsymbol{\psi}_m^{(s)} + \rho\mathrm{Log}_{\boldsymbol{z}^{(s+1)}}\left(\boldsymbol{\theta}_m^{(s+1)}\right), \tag{7c}$$

where $\boldsymbol{z}^{(s+1)}$ is the global consensus parameter, $\boldsymbol{\psi}_m^{(s+1)}$ the dual variable, $\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathrm{Q},m}(\boldsymbol{z}^{(s+1)})$ the Riemannian gradient of local loss at $\boldsymbol{z}^{(s+1)}$, L_m the Lipschitz constant for agent m, and $\mathcal{R}_{\boldsymbol{z}^{(s+1)}}$ the retraction operator from global parameter $\boldsymbol{z}^{(s+1)}$. (7a) uses the simplified

Algorithm 1 DR-ADMM

Input: $[\{\boldsymbol{X}_m, \boldsymbol{y}_m\}, \boldsymbol{\theta}_m^{(s)}, \boldsymbol{\psi}_m^{(s)}]_{m=1}^M$, κ_Q, δ, ρ, $\boldsymbol{L}$
Output: $\boldsymbol{z}^{(s+1)}$, $[\boldsymbol{\theta}_m^{(s+1)}]_{m=1}^M$, $[\boldsymbol{\psi}_m^{(s+1)}]_{m=1}^M$, $\|\boldsymbol{r}_{\text{pri}}^{(s)}\|_2$, $\|\boldsymbol{r}_{\text{dual}}^{(s)}\|_2$

1: $[\boldsymbol{\varphi}_m^{(s)}]_{m=1}^M = [\boldsymbol{\theta}_m^{(s)}]_{m=1}^M + \frac{[\boldsymbol{\psi}_m^{(s)}]_{m=1}^M}{\rho}$
2: $\boldsymbol{z}^{(s+1)} = \frac{1}{2}\Pi_{\mathcal{T}}\left\{\text{atan2}\left[\sum_{m=1}^M \sin\left(2\boldsymbol{\varphi}_m^{(s)}\right), \sum_{m=1}^M \cos\left(2\boldsymbol{\varphi}_m^{(s)}\right)\right]\right\}$
3: **for** $m = 1, \ldots, M$ **in PARALLEL do**
4: Compute quantum kernel matrix: $[\boldsymbol{\kappa}_Q]_m(\boldsymbol{X}_m, \boldsymbol{X}_m|\boldsymbol{z}^{(s+1)})$
5: **for** $p = 1, 2, \ldots, P$ **in PARALLEL do**
6: Perform kernel evaluations $[\boldsymbol{\kappa}_Q]_m(\boldsymbol{\theta}_m^{(s)} \pm \delta \boldsymbol{e}_p|\boldsymbol{z}^{(s+1)})$
7: **end for**
8: Compute derivatives via parameter shift: $\frac{\partial[\boldsymbol{\kappa}_Q]_m}{\partial\boldsymbol{\theta}}$
9: Compute local gradient: $\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\text{Q},m}(\boldsymbol{z}^{(s+1)})$
10: Update: $\boldsymbol{\theta}_m^{(s+1)} = \mathcal{R}_{\boldsymbol{z}^{(s+1)}}\left(-\frac{\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\text{Q},m}(\boldsymbol{z}^{(s+1)}) + \boldsymbol{\psi}_m^{(s)}}{\rho + L_m}\right)$
11: Update: $\boldsymbol{\psi}_m^{(s+1)} = \boldsymbol{\psi}_m^{(s)} + \rho \text{Log}_{\boldsymbol{z}^{(s+1)}}(\boldsymbol{\theta}_m^{(s+1)})$
12: **end for**
13: Compute residuals: $\|\boldsymbol{r}_{\text{pri}}^{(s)}\|_2$, $\|\boldsymbol{r}_{\text{dual}}^{(s)}\|_2$

Karcher mean [33] to compute $\boldsymbol{z}^{(s+1)}$ in non-Euclidean fashion. The optimization of DR-ADMM (7) iterates until both the primal and the dual residuals fall below the predefined tolerance thresholds $\|\boldsymbol{r}_{\text{pri}}^{(s)}\|_2 \leq \epsilon_{\text{pri}}$ and $\|\boldsymbol{r}_{\text{dual}}^{(s)}\|_2 \leq \epsilon_{\text{dual}}$, respectively. The primal and dual residuals are computed as $\boldsymbol{r}_{\text{pri}}^{(s)} = [d_{\mathcal{T}}(\boldsymbol{\theta}_1^{(s)}, \boldsymbol{z}^{(s)}) \ \ldots \ d_{\mathcal{T}}(\boldsymbol{\theta}_M^{(s)}, \boldsymbol{z}^{(s)})]^\intercal$ and $\boldsymbol{r}_{\text{dual}}^{(s)} = \rho d_{\mathcal{T}}(\boldsymbol{z}^{(s)}, \boldsymbol{z}^{(s-1)})$, respectively. The implementation details of DR-ADMM are presented in Algorithm 1. In $\mathcal{T}^P$, the Karcher mean for $\boldsymbol{z}^{(s+1)}$ can be reduced to the circular mean (Alg. 1-line 2).

Theorem 1 (Convergence of DR-ADMM). *Let the negative marginal log-likelihood functions $\mathcal{L}_{Q,m} : \mathcal{T}^p \to \mathbb{R}$ be L_p-smooth $\forall m \in [1, M]$ on the torus manifold $\mathcal{T}^p$ with bounded projections $\Pi_{\mathcal{T}}(\cdot)$, and assume the existence of a uniform bound $C < \infty$ such that $\|\nabla\mathcal{L}_{Q,m}(\cdot)\|_{\mathcal{T}} \leq C$. Then, for a sufficiently large penalty parameter $\rho > 0$, the sequence $\{\boldsymbol{\theta}_m^{(s)}, \boldsymbol{\psi}_m^{(s)}, \mathbf{z}^{(s)}\}$ generated by s iterations of the Distributed Riemannian ADMM algorithm converges to a stationary point $(\boldsymbol{\theta}_m^*, \boldsymbol{\psi}_m^*, \boldsymbol{z}^*)$ that satisfies the KKT conditions for the consensus problem, characterized by:*

- ***Primal Feasibility:*** *The primal residuals vanish, $\lim_{s\to\infty} \text{d}_{\mathcal{T}}(\boldsymbol{\theta}_m^{(s)}, \mathbf{z}^{(s)}) = 0$, yielding $\boldsymbol{\theta}_m^* = \mathbf{z}^* \ \forall m$.*
- ***Dual Feasibility and Stationarity:*** *The dual residuals vanish, $\lim_{s\to\infty} \rho\|\mathbf{z}^{(s)} - \mathbf{z}^{(s-1)}\|_{\mathcal{T}} = 0$, and the gradient of the negative marginal log-likelihood functions with respect to the consensus variable is zero, i.e., $\sum_{m=1}^M \nabla\mathcal{L}_{Q,m}(\mathbf{z}^*) = 0$.*
- ***Convergence Rate:*** *The algorithm achieves a sublinear convergence rate of $O(1/S)$, where S is the total number of iterations. To achieve ξ-accuracy in consensus residual, the required number of iterations is $S = O\left(\frac{(\rho + L_{max})(V^{(0)} - V^*)}{\xi}\right)$, where $L_{max} = \max_m L_m$, and $V^* = \inf_s V^{(s)}$ is the optimal value.*

Algorithm 2 DQGP

Input: $\mathcal{D} = \{\boldsymbol{X}, \boldsymbol{y}\}$, M, δ, $\kappa_Q(\Phi(q, \iota))$, ρ, $\boldsymbol{L}$, F, s_{max}, T, ϵ_{pri}, ϵ_{dual}
Output: $\boldsymbol{z}^*$, $\text{NLPD}_{\text{test}}$, $\text{NRMSE}_{\text{test}}$

1: **Initialize:** $\boldsymbol{\theta}_m^{(0)}, \boldsymbol{z}^{(0)}, \boldsymbol{\psi}_m^{(0)}, \kappa_Q(\boldsymbol{z}^{(0)}), NLPD_{CV}^* = \infty, t = 0, s = 0$
2: $\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{test}} \leftarrow$ `TrainTestSplit`$(\mathcal{D})$
3: $[\mathcal{D}_m]_{m=1}^M \leftarrow$ `Regionalk-dTreeSplit` $(\mathcal{D}_{\text{train}}, M)$
4: **while** $s < s_{max}$ and $t < T$ **do**
5: $\boldsymbol{z}^{(s+1)}, [\boldsymbol{\theta}_m^{(s+1)}]_{m=1}^M, [\boldsymbol{\psi}_m^{(s+1)}]_{m=1}^M, \|\boldsymbol{r}_{\text{pri}}^{(s)}\|_2, \|\boldsymbol{r}_{\text{dual}}^{(s)}\|_2$ $\leftarrow$ `DR-ADMM`$([\mathcal{D}_m, \boldsymbol{\theta}_m^{(s)}, \boldsymbol{\psi}_m^{(s)}]_{m=1}^M, \kappa_Q, \delta, \rho, \boldsymbol{L})$
6: $NLPD_{CV} \leftarrow$ `F-fold_Cross-Validation`$(\boldsymbol{z}^{(s+1)}, \bigcup_m \mathcal{D}_m)$
7: **if** $\text{NLPD}_{\text{CV}} < \text{NLPD}_{\text{CV}}^*$ **then**
8: $\text{NLPD}_{\text{CV}}^* = \text{NLPD}_{\text{CV}}$, $\boldsymbol{z}^* = \boldsymbol{z}^{(s+1)}$, $t = 0$
9: **else** $t = t + 1$
10: **end if**
11: **if** $\|\boldsymbol{r}_{\text{pri}}^{(s)}\|_2 \leq \epsilon_{\text{pri}}$ **and** $\|\boldsymbol{r}_{\text{dual}}^{(s)}\|_2 \leq \epsilon_{\text{dual}}$ **then break**
12: $s = s + 1$
13: **end while**
14: $\text{NLPD}_{\text{test}}, \text{NRMSE}_{\text{test}} \leftarrow$ `QGP-prediction`$(\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{test}}, \boldsymbol{z}^*)$

Proof: (Sketch) Under assumptions 3 and 4, define a Lyapunov function V for the s^{th} iteration with dual variable regularization as,

$$V^{(s)} = \sum_{m=1}^M \left[\mathcal{L}_{Q,m}(\boldsymbol{z}^{(s)}) + \langle\boldsymbol{\psi}_m^{(s)}, \log_{\boldsymbol{z}^{(s)}}(\boldsymbol{\theta}_m^{(s)})\rangle_{\mathcal{T}} + \frac{\rho}{2}\text{d}_{\mathcal{T}}^2(\boldsymbol{z}^{(s)}, \boldsymbol{\theta}_m^{(s)})\right] + \frac{1}{2\rho}\sum_{m=1}^M \|\boldsymbol{\psi}_m^{(s)}\|_{\mathcal{T}}^2.$$

Then prove that it is non-increasing,

$$V^{(s+1)} - V^{(s)} \leq -|\Lambda(\rho, L_m)| \times \sum_{m=1}^M \left[\|\nabla\mathcal{L}_{Q,m}(\boldsymbol{z}^{(s)}) + \boldsymbol{\psi}_m^{(s)}\|_{\mathcal{T}}^2 + \text{d}_{\mathcal{T}}^2(\boldsymbol{\theta}_m^{(s)}, \boldsymbol{z}^{(s)})\right]. \quad (8)$$

From (8), show that the optimization variables $\boldsymbol{\theta}_m$, $\boldsymbol{\psi}_m$, $\boldsymbol{z}$, the primal residual $(\sum_{s=0}^\infty \sum_{m=1}^M \text{d}_{\mathcal{T}}^2(\boldsymbol{\theta}_m^{(s)}, \boldsymbol{z}^{(s)}))$ and the dual residual $(\sum_{s=1}^\infty \rho^2\|\boldsymbol{z}^{(s)} - \boldsymbol{z}^{(s-1)}\|_{\mathcal{T}}^2)$ series remain uniformly bounded during DR-ADMM iterations. Ultimately, in the limit, these residuals vanish, and the optimization variables converge to $\boldsymbol{\theta}_m^*, \boldsymbol{\psi}_m^*, \boldsymbol{z}^*$. Applying (6b) at these limit points leads to $\sum_{m=1}^M \nabla\mathcal{L}_{Q,m}(\boldsymbol{z}^*) = 0$. The stated sublinear convergence rate can be proven using (8).

3.2 Distributed Quantum Gaussian Process

3.2.1 Quantum Encoding Circuits. We primarily employ two types of quantum encoding circuits. The first is the Chebyshev parameterized quantum circuit [27, 51] which consists of R_Y, R_X, and conditional-R_Z quantum rotational gates. The conditional quantum gates play a crucial role by inducing entanglement among the qubits, which is an essential property for representing complex quantum systems. The second is the Hubregtsen encoding circuit [21], which contains the Hadamard gate H, rotation gates R_Z, R_Y, and conditional-R_Z as shown in Fig. 2. The Hadamard gate introduces superposition among qubits. The overall configuration of these circuits depends on the number of qubits q, the number of layers ι, and the dimensionality D of the input space. Accordingly, the encoding circuits have P parameters $\{\theta_1, \theta_2, \ldots, \theta_P\} \in [0, \pi]^P$.

Table 1: $\text{NLPD}_{\text{test}}$ and $\text{NRMSE}_{\text{test}}$ for $N = 500$. Set 1 = $\{0.58, 2.45, 1.88, 1.40, 0.31, 1.44\}$ and Set 2 = $\{1.18, 2.99, 2.30, 1.88, 0.49, 0.49\}$.

Dataset	Subset	Single Agent Method		M	Distributed Methods					
		Full-GP [52]			DQGP-DR-ADMM		FACT-GP [11]		apxGP [54]	
		$NLPD_{\text{test}} \downarrow$	$NRMSE_{\text{test}} \downarrow$		$NLPD_{\text{test}} \downarrow$	$NRMSE_{\text{test}} \downarrow$	$NLPD_{\text{test}} \downarrow$	$NRMSE_{\text{test}} \downarrow$	$NLPD_{\text{test}} \downarrow$	$NRMSE_{\text{test}} \downarrow$
SRTM	N17E073	0.34 ± 0.09	0.10 ± 0.009	4	**0.75 ± 0.42**	**0.10 ± 0.01**	1.01 ± 0.04	0.18 ± 0.01	0.84 ± 0.15	0.21 ± 0.04
				8	**0.64 ± 0.38**	**0.03 ± 0.006**	1.10 ± 0.03	0.20 ± 0.01	0.97 ± 0.13	0.22 ± 0.03
				27	**0.50 ± 0.27**	**0.09 ± 0.02**	1.15 ± 0.03	0.21 ± 0.01	2.08 ± 1.96	0.27 ± 0.03
	N43W080	0.19 ± 0.09	0.06 ± 0.006	4	**0.21 ± 0.24**	**0.07 ± 0.02**	1.06 ± 0.03	0.16 ± 0.01	12.82 ± 3.40	0.29 ± 0.02
				8	**0.36 ± 0.56**	**0.07 ± 0.02**	1.14 ± 0.03	0.18 ± 0.01	30.82 ± 14.43	0.31 ± 0.02
				27	**0.34 ± 0.38**	**0.07 ± 0.02**	1.21 ± 0.03	0.19 ± 0.01	115.47 ± 62.43	0.31 ± 0.02
	N45W123	0.47 ± 0.10	0.10 ± 0.01	4	**0.79 ± 0.56**	**0.16 ± 0.03**	1.05 ± 0.03	0.17 ± 0.01	1.22 ± 0.06	0.23 ± 0.02
				8	**0.63 ± 0.31**	**0.12 ± 0.02**	1.1 ± 0.03	0.18 ± 0.01	1.42 ± 0.09	0.27 ± 0.04
				27	**0.78 ± 0.42**	**0.12 ± 0.02**	1.15 ± 0.04	0.19 ± 0.01	1.53 ± 0.18	0.29 ± 0.05
	N47W124	0.68 ± 0.12	0.13 ± 0.02	4	**1.24 ± 0.39**	**0.15 ± 0.02**	1.26 ± 0.05	0.23 ± 0.02	50.50 ± 18.56	0.38 ± 0.04
				8	1.41 ± 0.78	**0.15 ± 0.02**	**1.35 ± 0.04**	0.26 ± 0.02	40.90 ± 16.37	0.39 ± 0.04
				27	1.97 ± 0.66	**0.16 ± 0.01**	**1.39 ± 0.05**	0.27 ± 0.02	35.36 ± 30.90	0.38 ± 0.04
2D QGP prior	Set 1	−0.86 ± 0.05	0.03 ± 0.007	4	**-0.27 ± 0.21**	**0.04 ± 0.006**	0.58 ± 0.26	0.14 ± 0.03	1.49 ± 2.00	0.23 ± 0.08
				8	**-0.24 ± 0.25**	**0.03 ± 0.006**	0.78 ± 0.27	0.17 ± 0.03	2.73 ± 3.19	0.26 ± 0.07
				27	**-0.22 ± 0.15**	**0.04 ± 0.006**	0.93 ± 0.24	0.20 ± 0.04	4.08 ± 5.75	0.28 ± 0.07
	Set 2	−0.82 ± 0.07	0.03 ± 0.008	4	**-0.29 ± 0.19**	**0.03 ± 0.006**	0.54 ± 0.25	0.14 ± 0.02	0.94 ± 0.77	0.23 ± 0.06
				8	**-0.31 ± 0.15**	**0.03 ± 0.006**	0.72 ± 0.27	0.16 ± 0.02	2.04 ± 1.91	0.29 ± 0.08
				27	**-0.26 ± 0.18**	**0.03 ± 0.01**	0.88 ± 0.25	0.20 ± 0.02	3.34 ± 5.68	0.30 ± 0.09

3.2.2 Quantum Kernels. A standard choice for a quantum kernel based on the quantum fidelity measure $\mathcal{F}$ (5) is, $\kappa_{\mathcal{F}}(\boldsymbol{x}, \boldsymbol{x}'|\boldsymbol{\theta}) = |\langle\Phi(\boldsymbol{x}, \boldsymbol{\theta})|\Phi(\boldsymbol{x}', \boldsymbol{\theta})\rangle|^2 = \left|\langle 0^{\otimes q}|U^\dagger(\boldsymbol{x}, \boldsymbol{\theta})U(\boldsymbol{x}', \boldsymbol{\theta})|0^{\otimes q}\rangle\right|^2$, where U denotes the quantum encoding circuit with q qubits and $\boldsymbol{\theta}$ the hyperparameters. Since $\kappa_{\mathcal{F}}$ lacks observable-dependent operations, its expressivity and modularity are limited. To manage this limitation, we employ the *Projected Quantum Kernel* (PQK) [16]. By incorporating measurements [19] based on observables, PQK maps the quantum states into a classical feature vector space and then applies the classical outer kernel on that projected space,

$$\kappa_{PQK}(\boldsymbol{x}, \boldsymbol{x}'|\boldsymbol{\theta}) = \kappa_{\text{outer}}(\langle O\rangle_{\psi(\boldsymbol{x})}, \langle O\rangle_{\psi(\boldsymbol{x}')}) \tag{9}$$

where $\langle O\rangle_{\psi(\boldsymbol{x})} = \langle 0^{\otimes q}|U^\dagger(\boldsymbol{x}, \boldsymbol{\theta})OU(\boldsymbol{x}, \boldsymbol{\theta})|0^{\otimes q}\rangle$. The most common choices for the observable operator O include: (i) Pauli-Z measurement $O_Z = \bigotimes_{i=1}^{q}(\boldsymbol{\sigma}_Z)_i$, which captures global correlation among q qubits; (ii) Local Pauli measurement $O_{\text{local}} = \{(\boldsymbol{\sigma}_Z)_1, \ldots, (\boldsymbol{\sigma}_Z)_q\}$, which measure individual qubits; and (iii) Mixed Pauli measurement $O_{\text{mixed}} = \{(\boldsymbol{\sigma}_Z)_1(\boldsymbol{\sigma}_Z)_2, (\boldsymbol{\sigma}_X)_1(\boldsymbol{\sigma}_Y)_2, \ldots\}$, which measure pairwise qubit correlations. Any classical kernel can be used for the outer kernel κ_{outer} [37]. With PQK, the computational cost is significantly reduced as it only requires evaluating $O(\text{card}(O))$ expectations, while the fidelity kernel involves computation of $O(2^q)$ state overlap. In addition, the observable operator measurements provide a physical interpretability with respect to the feature vector space and a way to implement them on quantum hardware. The implementation details of the proposed *Distributed Quantum Gaussian Process* are presented in Algorithm 2.

In Fig. 1, we depict the conceptual architecture of DQGP, where each agent is a computational node assigned to a local subset of the spatial dataset and trains a local QGP model. After each local step, agents push a learned $\boldsymbol{\theta_m}$ to a central server, which aggregates them into a consensus model $\boldsymbol{z}$. Next, each agent pulls $\boldsymbol{z}$ from the central server to form a consensus on their local models with the globally aggregated model. This push-pull mechanism produces model-level consensus. In Algorithm 2, we use the negative log predictive density (NLPD) for validation,

$$\text{NLPD}_{\text{test}} = \frac{1}{\text{card}(\mathcal{D}_{\text{test}})} \sum_{j \in \mathcal{D}_{\text{test}}} \left[\frac{1}{2}\log(2\pi\sigma_*^2(j)) + \frac{(y_j - \mu_*(j))^2}{2\sigma_*^2(j)}\right]. \tag{10}$$

To evaluate the global parameter $\boldsymbol{z}$ across training iterations, we perform `F-fold_Cross-Validation` using combined datasets $\bigcup_m \mathcal{D}_m$. For each fold $f = 1, 2, \ldots, F$, the combined dataset is randomly shuffled and partitioned into training $\mathcal{D}^f_{\text{train}}$ and validation $\mathcal{D}^f_{\text{val}}$ sets to compute $\text{NLPD}^f_{\text{CV}}$. The mean of all these folds, NLPD_{CV}, is monitored over iterations to obtain the optimal global $\boldsymbol{z}^*$. The `QGP-prediction` equations are mathematically analogous to those of classical GPs, with the only difference being the substitution of the classical kernel κ by the quantum kernel κ_Q,

$$\mu_*(\boldsymbol{x}_*) = \boldsymbol{\kappa}_Q(\boldsymbol{x}_*, \boldsymbol{X})[\boldsymbol{\kappa}_Q(\boldsymbol{X}, \boldsymbol{X}) + \sigma_\epsilon^2 \boldsymbol{I}_N]^{-1}\boldsymbol{y},$$
$$\sigma_*^2(\boldsymbol{x}_*) = \boldsymbol{\kappa}_Q(\boldsymbol{x}_*, \boldsymbol{x}_*) - \boldsymbol{\kappa}_Q(\boldsymbol{x}_*, \boldsymbol{X})[\boldsymbol{\kappa}_Q(\boldsymbol{X}, \boldsymbol{X}) + \sigma_\epsilon^2 \boldsymbol{I}_N]^{-1}\boldsymbol{\kappa}_Q(\boldsymbol{X}, \boldsymbol{x}_*),$$

where $\boldsymbol{x}_*$ is the unknown input. Finally, we employ the normalized root mean squared error (NRMSE) to assess the prediction,

$$\text{NRMSE}_{\text{test}} = \frac{\sqrt{\frac{1}{\text{card}(\mathcal{D}_{\text{test}})}\sum_{j \in \mathcal{D}_{\text{test}}}(y_j - \mu_*(j))^2}}{|y_{max} - y_{min}|}. \tag{11}$$

Table 2: $\text{NLPD}_{\text{test}}$ and $\text{NRMSE}_{\text{test}}$ for $N = 5,000$. Set 1 = $\{0.58, 2.45, 1.88, 1.40, 0.31, 1.44\}$ and Set 2 = $\{1.18, 2.99, 2.30, 1.88, 0.49, 0.49\}$.

Dataset	Subset	Single Agent Method		M	Distributed Methods					
		Full-GP [52]			DQGP-DR-ADMM		FACT-GP [11]		apxGP-pxADMM [54]	
		$NLPD_{\text{test}} \downarrow$	$NRMSE_{\text{test}} \downarrow$		$NLPD_{\text{test}} \downarrow$	$NRMSE_{\text{test}} \downarrow$	$NLPD_{\text{test}} \downarrow$	$NRMSE_{\text{test}} \downarrow$	$NLPD_{\text{test}} \downarrow$	$NRMSE_{\text{test}} \downarrow$
SRTM	N17E073	-0.01 ± 0.02	0.06 ± 0.001	4	1.56 ± 0.79	$\mathbf{0.06 \pm 0.005}$	$\mathbf{1.11 \pm 0.005}$	0.19 ± 0.004	1.28 ± 0.04	0.24 ± 0.006
				8	1.62 ± 0.65	$\mathbf{0.06 \pm 0.004}$	$\mathbf{1.20 \pm 0.009}$	0.21 ± 0.005	1.38 ± 0.06	0.25 ± 0.006
				27	1.59 ± 0.56	$\mathbf{0.06 \pm 0.004}$	$\mathbf{1.23 \pm 0.01}$	0.22 ± 0.005	2.13 ± 0.25	0.27 ± 0.01
	N43W080	-0.66 ± 0.05	0.03 ± 0.001	4	$\mathbf{-0.36 \pm 0.23}$	$\mathbf{0.03 \pm 0.004}$	1.09 ± 0.009	0.16 ± 0.003	17.56 ± 1.50	0.27 ± 0.005
				8	$\mathbf{-0.34 \pm 0.26}$	$\mathbf{0.02 \pm 0.004}$	1.17 ± 0.01	0.17 ± 0.003	34.08 ± 2.65	0.28 ± 0.005
				27	$\mathbf{-0.19 \pm 0.37}$	$\mathbf{0.02 \pm 0.003}$	1.21 ± 0.01	0.18 ± 0.003	37.91 ± 10.16	0.28 ± 0.005
	N45W123	-0.06 ± 0.03	0.05 ± 0.002	4	$\mathbf{0.75 \pm 0.49}$	$\mathbf{0.05 \pm 0.006}$	1.09 ± 0.008	0.17 ± 0.006	1.19 ± 0.02	0.19 ± 0.008
				8	$\mathbf{0.51 \pm 0.51}$	$\mathbf{0.05 \pm 0.005}$	1.18 ± 0.01	0.18 ± 0.006	1.36 ± 0.02	0.23 ± 0.01
				27	$\mathbf{1.03 \pm 0.67}$	$\mathbf{0.05 \pm 0.003}$	1.24 ± 0.02	0.19 ± 0.007	1.58 ± 0.05	0.27 ± 0.02
	N47W124	0.29 ± 0.01	0.08 ± 0.003	4	5.21 ± 0.86	$\mathbf{0.07 \pm 0.004}$	$\mathbf{1.29 \pm 0.009}$	0.22 ± 0.009	60.95 ± 3.95	0.33 ± 0.01
				8	4.52 ± 1.31	$\mathbf{0.07 \pm 0.006}$	$\mathbf{1.34 \pm 0.009}$	0.23 ± 0.01	38.22 ± 3.31	0.33 ± 0.01
				27	4.38 ± 0.89	$\mathbf{0.07 \pm 0.006}$	$\mathbf{1.37 \pm 0.01}$	0.24 ± 0.01	13.65 ± 1.71	0.33 ± 0.01
2D QGP prior	Set 1	-0.88 ± 0.01	0.02 ± 0.003	4	0.96 ± 0.19	$\mathbf{0.03 \pm 0.004}$	$\mathbf{0.42 \pm 0.33}$	0.11 ± 0.02	2.05 ± 3.12	0.23 ± 0.08
				8	1.03 ± 0.19	$\mathbf{0.03 \pm 0.003}$	$\mathbf{0.61 \pm 0.34}$	0.13 ± 0.03	3.12 ± 3.75	0.25 ± 0.09
				27	1.35 ± 0.19	$\mathbf{0.03 \pm 0.004}$	$\mathbf{0.77 \pm 0.34}$	0.15 ± 0.03	3.62 ± 3.36	0.28 ± 0.09
	Set 2	-0.86 ± 0.03	0.03 ± 0.008	4	0.99 ± 0.24	$\mathbf{0.03 \pm 0.004}$	$\mathbf{0.49 \pm 0.21}$	0.11 ± 0.03	2.81 ± 2.91	0.24 ± 0.06
				8	0.96 ± 0.16	$\mathbf{0.03 \pm 0.003}$	$\mathbf{0.71 \pm 0.20}$	0.14 ± 0.03	4.11 ± 4.33	0.27 ± 0.07
				27	0.99 ± 0.19	$\mathbf{0.03 \pm 0.004}$	$\mathbf{0.88 \pm 0.22}$	0.16 ± 0.03	5.33 ± 5.67	0.28 ± 0.07

4 NUMERICAL EXPERIMENTS & RESULTS

We evaluate the predictive capabilities of our approach through experiments on both real-world and synthetic datasets. For the real-world datasets, we incorporate four tiles, N17E073, N43W080, N45W123, and N47W124 from NASA's Shuttle Radar Topography Mission (SRTM) [13]. The datasets have two-dimensional inputs representing latitude and longitude, with elevation as the output. As reported in [9], these datasets exhibit non-stationarity, i.e., different regions have varying degrees of variability. This is important for assessing whether the learned model can fully adapt to the local features, thus demonstrating its suitability for solving complex problems. For the synthetic dataset, we generate a quantum Gaussian process (QGP) prior using quantum kernels, sample points from it to act as a training dataset, and learn the original QGP.

We use two metrics, the NLPD (10) and NRMSE (11), which test complementary aspects of performance. Lower values for both metrics indicate better model performance. Since GPs are probabilistic, NLPD evaluates the quality of the predictive distribution, while NRMSE measures the accuracy of the mean predictions. All numerical experiments are conducted using classical quantum state vector simulators in Qiskit and PennyLane, employing quantum encoding circuits and kernels in sQUlearn [28]. The computational complexity analysis on a quantum simulator run on classical hardware does not represent the true complexity on current NISQ quantum hardware, hence this work does not involve complexity analysis.

Across the four environments from the SRTM real-world dataset, we use $N = \{500, 5,000\}$ samples, of which 10% is reserved for testing and the rest 90% is distributed among the agents as their respective training datasets. The features and target are z-score normalized to $[-3, 3]$ as the quantum parameters—mostly rotational—are naturally bounded. For quantum encoding, we employ a Chebyshev parameterized quantum circuit [27] with $q = 4$ qubits and $\iota = 3$ variational layers, resulting in $P = 24$ quantum hyperparameters. We utilize the projected quantum kernel κ_{PQK} (9) with the Pauli $\boldsymbol{\sigma}_X\boldsymbol{\sigma}_Y\boldsymbol{\sigma}_Z$ measurement operator and the Matérn outer kernel, whose parameters are set to $\ell = 1.0$, $\nu = 1.5$. Quantum hyperparameters are optimized using DR-ADMM (Algorithm 1) with shift value $\delta = (\pi/8)$ and ADMM parameters: penalty $\rho = 100$ and Lipschitz constant $\boldsymbol{L} = \{100\}_{m=1}^{M}$. For the synthetic dataset, we use $N = \{500, 5,000\}$ samples. The quantum encoding circuit is the Hubregtsen Encoding Circuit (Fig. 2), with $q = 3$ qubits and $\iota = 1$ layer resulting in $P = 6$ quantum hyperparameters. The quantum kernel is the projected κ_{PQK} (9) with the Pauli $\boldsymbol{\sigma}_X\boldsymbol{\sigma}_Y\boldsymbol{\sigma}_Z$ measurement operator. The outer kernel is Gaussian with parameter $\gamma = 1.0$. ADMM parameters remain identical to those of the SRTM dataset.

In Table 1, 2, we present the performance of our method across datasets of size $N = 500$ and $N = 5,000$, respectively. We report the mean and standard deviation of $\text{NLPD}_{\text{test}}$ and $\text{NRMSE}_{\text{test}}$ evaluated over 20 replications to reduce the effect of randomly assigned data. We compare our method with other single- and multi-agent GP methods: Full-GP [52], FACT-GP [11], and apxGP [54]. In addition to NLPD and NRMSE, the comparison allows for scalability assessment of network sizes in Table 1, 2. In Fig. 3, 4, we present the performance in the SRTM and 2D QGP datasets, respectively.

Aggregating improvements across the four SRTM environments for $N = 500$ and $N = 5,000$ datasets, DQGP achieves $51.1\% \pm 17.8\%$ lower $\text{NRMSE}_{\text{test}}$ than FACT-GP, and $65.2\% \pm 16.1\%$ lower

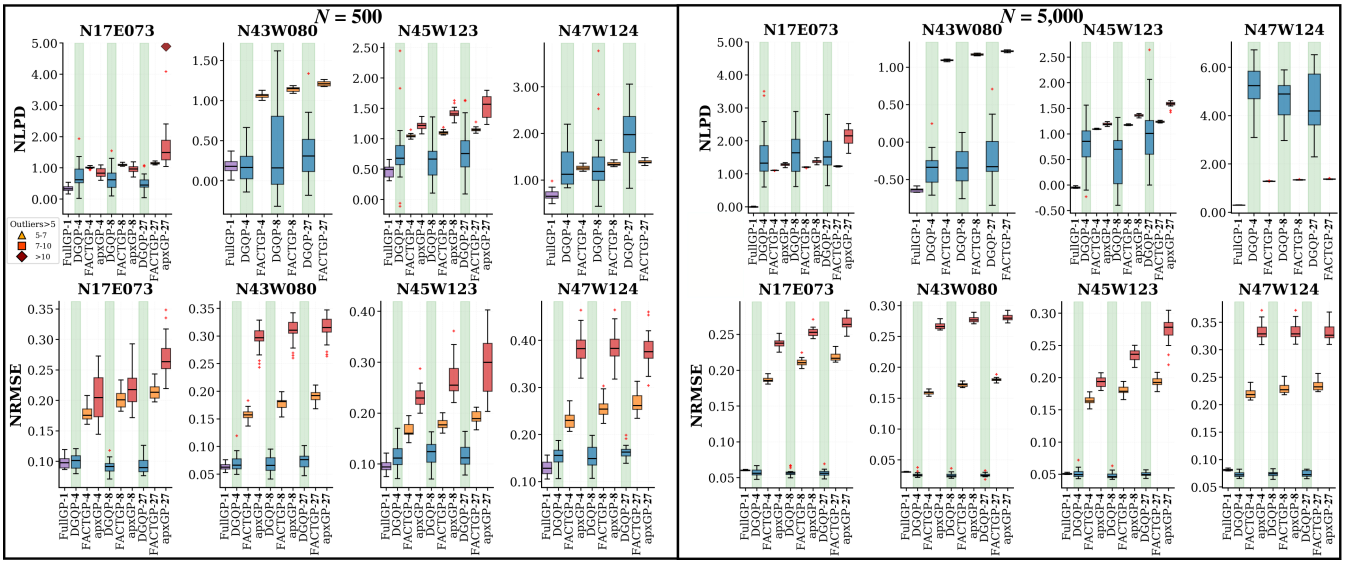

Figure 3: Performance of DQGP (green) with the SRTM dataset, compared to Full-GP [52], FACT-GP [11], and apx-GP [54]. For N43W080 and N47W124, we have excluded visualizing the apxGP results (worst performance) in NLPD for better readability.

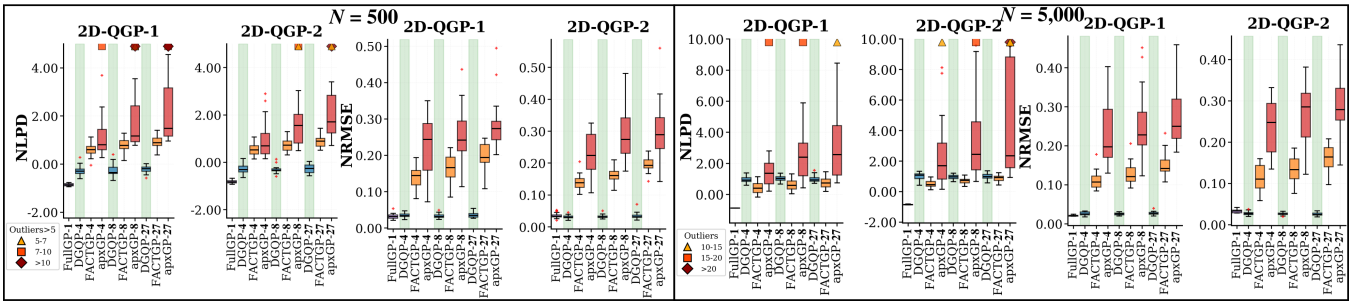

Figure 4: Performance of DQGP (green) on 2D QGP prior dataset, compared to Full-GP [52], FACT-GP [11], and apx-GP [54].

$\text{NRMSE}_{\text{test}}$ than apxGP. Moreover, our approach exhibits a competitive performance compared to the single-agent Full-GP method, showing a slight improvement of 1.4% on average for $\text{NRMSE}_{\text{test}}$. This trend suggests that DQGP scales effectively with network size for prediction accuracy. A trade-off is observed for $\text{NLPD}_{\text{test}}$ as the network size increases. In particular, DQGP achieves $8.9\% \pm 120.4\%$ improvement in $\text{NLPD}_{\text{test}}$ relative to FACT-GP, and $91.7\% \pm 11.2\%$ improvement in $\text{NLPD}_{\text{test}}$ relative to apxGP.

For the synthetic dataset, DQGP achieves on average $73.6\% \pm 10.2\%$ lower $\text{NRMSE}_{\text{test}}$ than FACT-GP, and $87.0\% \pm 4.0\%$ $\text{NRMSE}_{\text{test}}$ improvement over apxGP. Compared to the single-agent Full-GP, DQGP shows a 16.7% improvement in aggregated $\text{NRMSE}_{\text{test}}$. These results indicate that our method scales effectively as the network size increases. For uncertainty quantification with synthetic datasets, DQGP achieves $37.2\% \pm 108.6\%$ improvement in $\text{NLPD}_{\text{test}}$ relative to FACT-GP, and $67.8\% \pm 17.9\%$ improvement over apxGP.

Overall, the proposed DQGP demonstrates a substantial advantage in prediction over classical distributed counterparts, as is evident from the lower $\text{NRMSE}_{\text{test}}$ values for both $N = \{500, 5,000\}$. It also successfully quantifies the uncertainty for $N = 500$, as reflected by lower $\text{NLPD}_{\text{test}}$ values. However, for some datasets with $N = 5,000$, FACT-GP yields lower $\text{NLPD}_{\text{test}}$ due to its block-diagonal posterior covariance approximation, which produces less conservative and more stable uncertainty estimates. In contrast, DQGP with DR-ADMM prioritizes on aligning the global objective across the agents, resulting in more accurate mean predictions and only occasionally resulting in conservative uncertainty estimates.

5 CONCLUSION

This paper introduces a distributed quantum gaussian process (DQGP) method that scales the expressive power of quantum kernels to real-world, non-stationary datasets. To address the non-Euclidean quantum hyperparameter optimization, we propose a Distributed consensus Riemannian ADMM (DR-ADMM) approach. Experiments on real-world and synthetic datasets demonstrate enhanced performance compared to classical distributed methods. The results highlight the potential of DQGP for scalable probabilistic modeling on hybrid classical-quantum systems.

REFERENCES

[1] Luca Arceci, Viacheslav Kuzmin, and Rick Van Bijnen. 2024. Gaussian process model kernels for noisy optimization in variational quantum algorithms. *arXiv preprint arXiv:2412.13271* (2024).

[2] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. 2019. Quantum supremacy using a programmable superconducting processor. *Nature* 574, 7779 (2019), 505–510.

[3] Marcello Benedetti, Erika Lloyd, Stefan Sack, and Mattia Fiorentini. 2019. Parameterized quantum circuits as machine learning models. *Quantum science and technology* 4, 4 (2019), 043001.

[4] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. 2011. *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*. Vol. 3.

[5] Marco Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, et al. 2021. Variational quantum algorithms. *Nature Reviews Physics* 3, 9 (2021), 625–644.

[6] Tsung-Hui Chang, Mingyi Hong, and Xiangfeng Wang. 2014. Multi-agent distributed optimization via inexact consensus ADMM. *IEEE Transactions on Signal Processing* 63, 2 (2014), 482–497.

[7] Kuan-Cheng Chen, Samuel Yen-Chi Chen, Chen-Yu Liu, and Kin K Leung. 2025. Quantum-train-based distributed multi-agent reinforcement learning. In *2025 IEEE Symposium for Multidisciplinary Computational Intelligence Incubators (MCII Companion)*. IEEE, 1–5.

[8] Meng-Han Chen, Chao-Hua Yu, Jian-Liang Gao, Kai Yu, Song Lin, Gong-De Guo, and Jing Li. 2022. Quantum algorithm for Gaussian process regression. *Physical Review A* 106, 1 (2022), 012406.

[9] Weizhe Chen, Roni Khardon, and Lantao Liu. 2024. Adaptive robotic information gathering via non-stationary Gaussian processes. *The International Journal of Robotics Research* 43, 4 (2024), 405–436.

[10] Jack Cunningham and Jun Zhuang. 2025. Investigating and mitigating barren plateaus in variational quantum circuits: a survey. *Quantum Information Processing* 24, 2 (2025), 48.

[11] Marc Deisenroth and Jun Wei Ng. 2015. Distributed gaussian processes. In *International conference on machine learning*. PMLR, 1481–1490.

[12] Ahmad Farooq, Cristian A Galvis-Florez, and Simo Särkkä. 2024. Quantum-assisted Hilbert-space Gaussian process regression. *Physical Review A* 109, 5 (2024), 052410.

[13] Tom G Farr, Paul A Rosen, Edward Caro, Robert Crippen, Riley Duren, Scott Hensley, Michael Kobrick, Mimi Paller, Ernesto Rodriguez, Ladislav Roth, et al. 2007. The shuttle radar topography mission. *Reviews of geophysics* 45, 2 (2007).

[14] Cristian A Galvis-Florez, Ahmad Farooq, and Simo Särkkä. 2025. Provable Quantum Algorithm Advantage for Gaussian Process Quadrature. *arXiv preprint arXiv:2502.14467* (2025).

[15] Zoubin Ghahramani. 2013. Bayesian non-parametrics and the probabilistic approach to modelling. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 371, 1984 (2013), 20110553.

[16] Elies Gil-Fuster, Jens Eisert, and Vedran Dunjko. 2024. On the expressivity of embedding quantum kernels. *Machine Learning: Science and Technology* 5, 2 (2024), 025003.

[17] Ian Glendinning. 2005. The bloch sphere. In *QIA meeting. Vienna*.

[18] Gian Giacomo Guerreschi and Anne Y Matsuura. 2019. QAOA for Max-Cut requires hundreds of qubits for quantum speed-up. *Scientific reports* 9, 1 (2019), 6903.

[19] Teiko Heinosaari, Daniel Reitzner, and Peter Stano. 2008. Notes on joint measurability of quantum observables. *Foundations of Physics* 38, 12 (2008), 1133–1147.

[20] Hsin-Yuan Huang, Michael Broughton, Masoud Mohseni, Ryan Babbush, Sergio Boixo, Hartmut Neven, and Jarrod R McClean. 2021. Power of data in quantum machine learning. *Nature communications* 12, 1 (2021), 2631.

[21] Thomas Hubregtsen, David Wierichs, Elies Gil-Fuster, Peter-Jan HS Derks, Paul K Faehrmann, and Johannes Jakob Meyer. 2022. Training quantum embedding kernels on near-term quantum computers. *Physical Review A* 106, 4 (2022), 042431.

[22] Motonobu Kanagawa, Philipp Hennig, Dino Sejdinovic, and Bharath K Sriperumbudur. 2018. Gaussian processes and kernel methods: A review on connections and equivalences. *arXiv preprint arXiv:1807.02582* (2018).

[23] George P Kontoudis and Daniel J Stilwell. 2021. Decentralized nested Gaussian processes for multi-robot systems. In *IEEE International Conference on Robotics and Automation*. 8881–8887.

[24] George P Kontoudis and Daniel J Stilwell. 2023. Decentralized federated learning using Gaussian processes. In *IEEE International Symposium on Multi-Robot and Multi-Agent Systems*. 1–7.

[25] George P Kontoudis and Daniel J Stilwell. 2024. Scalable, federated Gaussian process training for decentralized multi-agent systems. *IEEE Access* 12 (2024), 77800–77815.

[26] George P Kontoudis and Daniel J Stilwell. 2025. Multi-Agent Federated Learning Using Covariance-Based Nearest Neighbor Gaussian Processes. *IEEE Transactions on Machine Learning in Communications and Networking* 4 (2025), 115–138.

[27] David A Kreplin and Marco Roth. 2024. Reduction of finite sampling noise in quantum neural networks. *Quantum* 8 (2024), 1385.

[28] David A Kreplin, Moritz Willmann, Jan Schnabel, Frederic Rapp, Manuel Hagelüken, and Marco Roth. 2025. sQUlearn: a Python library for quantum machine learning. *IEEE Software* (2025).

[29] Gaweł I Kuś, Sybrand van der Zwaag, and Miguel A Bessa. 2021. Sparse quantum Gaussian processes to counter the curse of dimensionality. *Quantum Machine Intelligence* 3, 1 (2021), 6.

[30] Martin Larocca, Supanut Thanasilp, Samson Wang, Kunal Sharma, Jacob Biamonte, Patrick J Coles, Lukasz Cincio, Jarrod R McClean, Zoë Holmes, and Marco Cerezo. 2025. Barren plateaus in variational quantum computing. *Nature Reviews Physics* (2025), 1–16.

[31] John M Lee. 2018. *Introduction to Riemannian manifolds*. Vol. 2. Springer.

[32] Jiaxiang Li, Shiqian Ma, and Tejes Srivastava. 2022. A riemannian admm. *arXiv preprint arXiv:2211.02163* (2022).

[33] Yongdo Lim and Miklós Pálfia. 2012. Matrix power means and the Karcher mean. *Journal of Functional Analysis* 262, 4 (2012), 1498–1514.

[34] Haitao Liu, Jianfei Cai, Yi Wang, and Yew Soon Ong. 2018. Generalized Robust Bayesian Committee Machine for Large-scale Gaussian Process Regression. In *International Conference on Machine Learning*. 3131–3140.

[35] Haitao Liu, Yew-Soon Ong, Xiaobo Shen, and Jianfei Cai. 2020. When Gaussian process meets big data: A review of scalable GPs. *IEEE Transactions on Neural Networks and Learning Systems* 31, 11 (2020), 4405–4423.

[36] Seth Lloyd. 2010. Quantum algorithm for solving linear systems of equations. In *APS March Meeting Abstracts*, Vol. 2010. D4–002.

[37] Sergei Manzhos and Manabu Ihara. 2024. Degeneration of kernel regression with Matern kernels into low-order polynomial regression in high dimension. *The Journal of Chemical Physics* 160, 2 (2024).

[38] Jarrod R McClean, Sergio Boixo, Vadim N Smelyanskiy, Ryan Babbush, and Hartmut Neven. 2018. Barren plateaus in quantum neural network training landscapes. *Nature communications* 9, 1 (2018), 4812.

[39] Rhea Parekh, Andrea Ricciardi, Ahmed Darwish, and Stephen DiAdamo. 2021. Quantum algorithms and simulation for parallel and distributed quantum computing. In *IEEE/ACM Intern. Workshop on Quantum Computing Software*. 9–19.

[40] Soohyun Park, Jae Pyoung Kim, Chanyoung Park, Soyi Jung, and Joongheon Kim. 2023. Quantum multi-agent reinforcement learning for autonomous mobility cooperation. *IEEE Communications Magazine* 62, 6 (2023), 106–112.

[41] Taylor L Patti, Khadijeh Najafi, Xun Gao, and Susanne F Yelin. 2021. Entanglement devised barren plateau mitigation. *Physical Review Research* 3, 3 (2021), 033090.

[42] Yifeng Peng, Xinyi Li, Zhemin Zhang, Samuel Yen-Chi Chen, Zhiding Liang, and Ying Wang. 2025. Breaking Through Barren Plateaus: Reinforcement Learning Initializations for Deep Variational Quantum Circuits. *arXiv preprint arXiv:2508.18514* (2025).

[43] John Preskill. 2018. Quantum computing in the NISQ era and beyond. *Quantum* 2 (2018), 79.

[44] Frederic Rapp and Marco Roth. 2024. Quantum gaussian process regression for bayesian optimization. *Quantum Machine Intelligence* 6, 1 (2024), 5.

[45] Stefan H Sack, Raimel A Medina, Alexios A Michailidis, Richard Kueng, and Maksym Serbyn. 2022. Avoiding barren plateaus using classical shadows. *PRX Quantum* 3, 2 (2022), 020365.

[46] Maria Schuld and Nathan Killoran. 2019. Quantum machine learning in feature Hilbert spaces. *Physical review letters* 122, 4 (2019), 040504.

[47] Maria Schuld and Francesco Petruccione. 2021. Quantum models as kernel methods. In *Machine Learning with Quantum Computers*. Springer, 217–245.

[48] Jian Qing Shi and Taeryon Choi. 2011. *Gaussian process regression analysis for functional data*. CRC press.

[49] Alistair WR Smith, AJ Paige, and MS Kim. 2023. Faster variational quantum algorithms with quantum kernel-based surrogate models. *Quantum Science and Technology* 8, 4 (2023), 045016.

[50] David Wierichs, Josh Izaac, Cody Wang, and Cedric Yen-Yu Lin. 2022. General parameter-shift rules for Quantum gradients. *Quantum* 6 (2022), 677.

[51] Chelsea A Williams, Annie E Paine, Hsin-Yu Wu, Vincent E Elfving, and Oleksandr Kyriienko. 2023. Quantum chebyshev transform: Mapping, embedding, learning and sampling distributions. *arXiv preprint arXiv:2306.17026* (2023).

[52] Christopher KI Williams and Carl Edward Rasmussen. 2006. *Gaussian processes for machine learning*. Vol. 2. MIT press Cambridge, MA.

[53] Colin P. Williams. 2011. *Quantum Gates*. Springer London, London, 51–122.

[54] Ang Xie, Feng Yin, Yue Xu, Bo Ai, Tianshi Chen, and Shuguang Cui. 2019. Distributed Gaussian processes hyperparameter optimization for big data using proximal ADMM. *IEEE Signal Processing Letters* 26, 8 (2019), 1197–1201.

[55] Paolo Zanardi and Nikola Paunković. 2006. Ground state overlap and quantum phase transitions. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics* 74, 3 (2006), 031123.

[56] Zhikuan Zhao, Jack K Fitzsimons, and Joseph F Fitzsimons. 2019. Quantum-assisted Gaussian process regression. *Physical Review A* 99, 5 (2019), 052331.