i-QLS: Quantum-supported Algorithm for Least Squares Optimization in Non-Linear Regression

Supreeth Mysore Venkatesh[1,2][0000−0002−9824−7399], Antonio Macaluso[2][0000−0002−1348−250X], Diego Arenas[3][0000−0001−7829−6102], Matthias Klusch[2][0009−0009−5431−8640], and Andreas Dengel[1,3][0000−0002−6100−8255]

[1] University of Kaiserslautern-Landau (RPTU), Kaiserslautern, Germany
{supreeth.mysore}@rptu.de
[2] German Research Center for Artificial Intelligence (DFKI), Saarbruecken, Germany
{supreeth.mysore, antonio.macaluso, matthias.klusch}@dfki.de
[3] German Research Center for Artificial Intelligence (DFKI), Kaiserslautern, Germany
{diego.arenas,andreas.dengel}@dfki.de

Abstract. We propose an iterative quantum-assisted least squares (i-QLS) optimization method that leverages quantum annealing to overcome the scalability and precision limitations of prior quantum least squares approaches. Unlike traditional QUBO-based formulations, which suffer from an exponential qubit overhead due to fixed discretization, our approach refines the solution space iteratively, enabling exponential convergence while maintaining a constant qubit requirement per iteration. This iterative refinement transforms the problem into an anytime algorithm, allowing for flexible computational trade-offs. Furthermore, we extend our framework beyond linear regression to non-linear function approximation via spline-based modeling, demonstrating its adaptability to complex regression tasks. We empirically validate i-QLS on the D-Wave quantum annealer, showing that our method efficiently scales to high-dimensional problems, achieving competitive accuracy with classical solvers while outperforming prior quantum approaches. Experiments confirm that i-QLS enables near-term quantum hardware to perform regression tasks with improved precision and scalability, paving the way for practical quantum-assisted machine learning applications.

Keywords: Quantum Annealing · Least Squares Optimization · Non-Linear Regression · Quantum Machine Learning

1 Background

A key distinction in machine learning methodologies lies in their optimization strategies, which are shaped by the assumptions that models make about the relationship between input features and the target variable. Models as Neural networks rely on non-convex optimization, often requiring extensive datasets and prolonged training due to the difficulty of escaping local minima [7]. In contrast, parametric models based on least squares (LS) optimization [10], such as splines

[19] and support vector machines [2], benefit from convex optimization, ensuring global optimality and theoretical robustness. However, the polynomial complexity of matrix operations in LS optimization introduces significant computational constraints as the number of features grows, limiting scalability.

Recently, several quantum machine learning models have been proposed for supervised learning tasks[14]. However, relatively few studies have explored the potential of leveraging near-term quantum computing exclusively for training and parameter estimation while maintaining a classical model for inference [20,12]. In particular, quantum annealing has been proposed as a method for reformulating LS optimization into a Quadratic Unconstrained Binary Optimization (QUBO) problem for execution on quantum hardware [3,1,4]. Despite these advancements, quantum annealing approaches face two fundamental bottlenecks. First, scalability remains a major challenge, as a tradeoff exists between the number of qubits and the precision of estimates of the optimized weights. Second, these methods are primarily to linear models, which fail to capture the complexities of real-world problems.

In this work, we introduce i-QLS, an iterative for quantum-assisted least squares optimization that overcomes the precision-scalability tradeoff inherent in prior quantum annealing-based methods. Instead of attempting to solve the problem in a single QUBO formulation with a fixed discretization, our approach iteratively refines the search space over multiple annealing cycles, exponentially converging to the optimal solution. This iterative approach is an anytime algorithm, meaning that at any point during execution, we can extract the best solution found, allowing for flexible computational tradeoffs. Furthermore, our method enables scalable adaptation to the constraints of existing quantum hardware by adjusting the granularity of discretization dynamically across iterations. In addition, we test i-QLS to non-linear regression via spline-based approximations, further demonstrating the versatility of our method beyond conventional linear regression. Our objective is to establish a practical and scalable framework for quantum-enhanced least squares optimization that leverages quantum hardware efficiently while remaining competitive with classical solvers. To summarize, this paper makes the following key contributions:

- **Iterative QUBO Refinement for Least Squares Optimization:** We introduce a novel iterative quantum-assisted least squares optimization strategy that mitigates the precision-qubit tradeoff inherent in prior single-shot QUBO formulations.
- **Scalability and Anytime Computation:** Our approach enhances scalability by enabling computation-limited devices to obtain progressively refined solutions throughout execution, supporting flexible computational tradeoffs.
- **Extension to Nonlinear Regression:** We extend our iterative framework beyond linear regression by incorporating spline-based function approximation, demonstrating its adaptability to complex modeling tasks.
- **Empirical Validation on Quantum Hardware:** We implement and evaluate our method on the D-Wave quantum annealer, showing that iterative refinement facilitates exponential convergence in optimization precision.

2 Related Works

Quantum annealing has been recently explored as an alternative paradigm for solving combinatorial optimization problems that naturally map onto a QUBO formulation. Although the least squares (LS) problem in regression does not inherently conform to this framework, few studies have attempted to adapt the LS formulation to harness quantum annealing for regression purposes.

For instance, [4] and [1] discretize the solution space by encoding each continuous weight as a series of binary variables, thereby formulating a corresponding QUBO problem that is amenable to solution via a quantum annealer. A significant limitation of these approaches is that increasing the precision of the weights necessitates an exponential increase in the number of qubits, rendering the method infeasible for large-scale problems on current quantum hardware. Consequently, classical LS solvers—such as direct matrix inversion [22], QR decomposition [6], or iterative gradient-based methods [13]—often outperform quantum approaches due to their capacity to achieve high precision without incurring the substantial overhead associated with quantum hardware.

Moreover, existing quantum annealing methods for LS have predominantly focused on linear regression, thereby neglecting non-linear problems that are more representative of real-world applications. In contrast, gate-based quantum computation has addressed non-linear approximation through the development of quantum splines [15,11]. Splines, which are piecewise polynomial functions with smoothness constraints, provide an effective tool for modeling non-linear relationships within a structured regression framework. In the initial formulation of quantum splines [15], a specific LS formulation based on B-splines [5] was employed to exploit the computational advantages of the HHL algorithm [9] for solving sparse linear systems. Although this approach offers a theoretical computational advantage, its reliance on fault-tolerant quantum computation limits its near-term applicability. Consequently, a variational counterpart has been proposed to leverage near-term quantum devices [11], although it has not yet demonstrated a robust advantage over classical methods.

Our work builds on these methodologies but fundamentally differs in two key aspects. First, rather than formulating a fixed QUBO problem with a predetermined discretization, we introduce an iterative refinement process. In our approach, each QUBO solution serves as the input for the subsequent iteration, progressively narrowing the weight search space while maintaining a fixed qubit requirement. This iterative refinement accelerates convergence exponentially and overcomes the scalability limitations encountered in previous methods. Our method is inspired by classical iterative refinement techniques commonly used in numerical optimization—such as Newton's method[17], gradient descent[8], and expectation-maximization [16]. To the best of our knowledge, apart from combinatorial optimization problem like graph-clustering [18], previous QUBO-based quantum regression methods have not incorporated an iterative refinement mechanism, rendering our approach unique in this context.

Second, our approach explicitly extends quantum-assisted regression to non-linear function approximation by integrating quantum spline formulations. In

our framework, the advantages of splines—namely, their ability to model non-linear relationships through piecewise polynomial fitting with smoothness constraints—are harnessed within the iterative QUBO process to determine optimal spline coefficients. This integration enables the capture of non-linear dynamics while mitigating the fixed discretization issues of prior methods. By unifying iterative refinement with spline-based modeling, our method addresses the limitations of both conventional quantum annealing approaches and the earlier quantum spline formulations that rely on fault-tolerant computation.

3 Preliminaries

Least squares optimization is a technique in linear regression, where the goal is to identify the best-fitting linear relationship between input variables and a target by minimizing the sum of squared differences between observed and predicted values. This approach underpins many statistical models due to its simplicity and efficiency [10]. To extend these ideas to more complex, non-linear relationships, spline functions are often employed[2]. Spline functions are smoothing methods suitable for modelling the relationships between variables, typically adopted either as a visual aid in data exploration or for estimation purposes [10]. The underlying idea is to use linear models in which the input features are augmented with the so-called *basis expansions*. These consist of transformations of the original variables and serve to introduce non-linearity. Technically, splines are constructed by dividing the sample data into sub-intervals delimited by breakpoints, also referred to as knots. A fixed degree polynomial is then fitted in each of the segments, thus resulting in a piecewise polynomial regression. Formally, in the case of a 1-dimensional input vector $\boldsymbol{x}$, we can express its relationship with a target variable $\boldsymbol{y}$ in terms of an order-M spline with knots $\{\xi_k\}_{k=1,\dots,K}$:

$$\boldsymbol{y}_{n_{\text{obs}}\times 1} = \boldsymbol{N}_{n_{\text{obs}}\times(M+K)}\boldsymbol{\theta}_{(M+K)\times 1} + \boldsymbol{\epsilon}_{n_{\text{obs}}\times 1}, \tag{1}$$

where $\boldsymbol{\theta}$ is the vector of coefficients attached to the basis expansions, n_{obs} is the sample size, $\boldsymbol{\epsilon}$ is a random error term and the design matrix $\boldsymbol{N}$ contains $M+K$ basis functions defined as follows:

$$h_j(x) = x^{j-1}, \qquad j = 1, \cdots, M \tag{2}$$

$$h_{M+k} = (x-\xi_k)_+^{M-1}, \qquad k = 1, \cdots, K. \tag{3}$$

Notice that the formulation above includes M basis functions that determine the order-M polynomial to be fitted in each segment. The additional K basis introduce continuity constraints on the spline and its derivatives up to order $M-2$. The goal is then to find the optimal set of parameters $\boldsymbol{\theta}$ that minimises the residual sum of squares (RSS), with a *ridge regularisation* of the curvature acting as a roughness penalty:

$$\text{Score}\,(\boldsymbol{\theta}, \eta) = (\boldsymbol{y} - \boldsymbol{N}\boldsymbol{\theta})^T (\boldsymbol{y} - \boldsymbol{N}\boldsymbol{\theta}) + \eta\boldsymbol{\theta}^T \boldsymbol{\Omega}_{(M+K)\times(M+K)}\boldsymbol{\theta}, \tag{4}$$

where $\boldsymbol{\Omega}$ is a diagonal matrix containing the partial second derivatives. The solution to (4) can easily seen to be:

$$\hat{\boldsymbol{\theta}} = (\boldsymbol{N}^T\boldsymbol{N} + \eta\boldsymbol{\Omega})^{-1}\boldsymbol{N}^T\boldsymbol{y}. \tag{5}$$

4 Methods

In this section, we present the mathematical formulation and derivation of our proposed quantum-classical hybrid algorithm for performing iterative least squares optimization, named i-QLS. We begin by posing the standard linear regression model as a discretized optimization problem, then describe how to embed it into a QUBO form suitable for a quantum annealer. Subsequently, we extend the iterative scheme to refine the precision of the solution at each iteration without requiring a large number of qubits from the outset. Finally, we outline how the approach generalizes to non-linear regressions via spline representations.

4.1 Problem Formulation

Let

$$\mathbf{X} \in \mathbb{R}^{N\times d}, \quad \mathbf{y} \in \mathbb{R}^{N},$$

where N is the number of data points and d is the number of features (or variables). The goal of linear regression is to find a weight vector

$$\mathbf{w}^* = (w_1^*, w_2^*, \ldots, w_d^*)^\top$$

that minimizes the sum of squared errors (SSE):

$$\mathbf{w}^* = \underset{\mathbf{w}\in\mathbb{R}^d}{\arg\min}\ S(\mathbf{w}) \ = \ \sum_{n=1}^{N}\big(y_n - \mathbf{w}^\top \mathbf{x}_n\big)^2, \tag{6}$$

where $\mathbf{x}_n$ is the n-th sample. The least squares solution can be found in closed form via $\mathbf{w} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}$ in the non-singular case. However, our method operates on a discretized search space to allow a quantum annealer to sample candidate $\mathbf{w}$ values.

4.2 Discretization and QUBO Construction

Representing each weight parameter $w_i \in \mathbb{R}$ by m binary variables and initializing a sufficiently large interval $\Delta_i^{(0)}$ with lower bound $\ell_i^{(0)}$ and the upper bound $u_i^{(0)}$, we find the discretization step size:

$$\delta_i^{(0)} \ = \ \frac{\Delta_i^{(0)}}{2^m - 1} \quad \text{for} \quad i = 1, \ldots, d, \qquad \text{where} \qquad \Delta_i^{(0)} \ = \ (u_i^{(0)} - \ell_i^{(0)}) \tag{7}$$

Initializing $w_i^{(0)} \leftarrow \frac{(u_i^{(0)} + \ell_i^{(0)})}{2}$, our goal is to find $w_i^{(1)}$ restricted to 2^m equally spaced values in the interval $[\ell_i^{(0)}, u_i^{(0)}]$ that is closest to the optimal weight w_i^*. Mathematically,

$$\begin{aligned} w_i^{(1)*} &= \underset{w_i^{(1)}}{\arg\min}\, |w_i^{(1)} - w_i^*|, \\ \text{where} \quad w_i^{(1)} &\in \{\ell_i^{(0)} + \delta_i^{(0)} p \mid p \in \{0, 1, \ldots, 2^m - 1\}\} \end{aligned} \tag{8}$$

$$\mathbf{b} \;=\; \big(b_{1,0}, b_{1,1}, \ldots, b_{1,m-1}, \ldots, b_{d,0}, \ldots, b_{d,m-1}\big)^\top \tag{9}$$

be the binary encoding vector representing the discretized weights. The weight vector $\mathbf{w}$ can now be expressed as a linear function of $\mathbf{b}$:

$$\mathbf{w^{(1)}}(\mathbf{b}) \;=\; \boldsymbol{\ell^{(0)}} \;+\; \mathbf{D^{(0)}}\mathbf{B}(\mathbf{b}), \tag{10}$$

$$\text{where} \quad \boldsymbol{\ell} = (\ell_1, \ldots, \ell_d)^\top,$$

$\mathbf{D}$ is a diagonal matrix containing the discretization step sizes δ_i,

$$\mathbf{D} = \mathrm{diag}(\delta_1, \ldots, \delta_d),$$

and $\mathbf{B}(\mathbf{b})$ is the vector whose entries are sums of powers of two weighted by the bits i.e.,

$$\begin{aligned} \mathbf{B}(\mathbf{b}) &= (B_1(\mathbf{b}_1), \ldots, B_d(\mathbf{b}_d))^\top, \\ B_i(\mathbf{b}_i) \;&=\; \sum_{p=0}^{m-1} 2^{m-1-p}\, b_{i,p}, \\ \mathbf{b}_i \;&=\; \big(b_{i,0}, b_{i,1}, \ldots, b_{i,m-1}\big)^\top. \end{aligned} \tag{11}$$

Quadratic Cost Function The sum of squared errors for the discretized weights is

$$S^{(1)}(\mathbf{b}) \;=\; \sum_{n=1}^{N} \Big(y_n - \mathbf{w^{(1)}}(\mathbf{b})^\top \mathbf{x}_n\Big)^2. \tag{12}$$

Expanding this, we get

$$S^{(1)}(\mathbf{b}) \;=\; \sum_{n=1}^{N} \Big(y_n - \boldsymbol{\ell^{(0)}}^\top \mathbf{x}_n - \mathbf{B}(\mathbf{b})^\top \mathbf{D^{(0)}}^\top \mathbf{x}_n\Big)^2. \tag{13}$$

Since each component of $\mathbf{B}(\mathbf{b})$ is linear in the binary variables, $S^{(1)}(\mathbf{b})$ becomes a polynomial (up to quadratic order) in those bits:

$$S^{(1)}(\mathbf{b}) \;=\; \alpha^{(1)} \;+\; \sum_{r} \gamma_r^{(1)}\, b_r \;+\; \sum_{r<s} \Gamma_{r,s}^{(1)}\, b_r\, b_s, \tag{14}$$

where the summations over r, s run over all binary variables in $\mathbf{b}$, and $\alpha^{(1)}, \gamma_r^{(1)}, \Gamma_{r,s}^{(1)}$ are real coefficients derived from $\mathbf{X}, \mathbf{y}, \boldsymbol{\ell^{(0)}}, \mathbf{D^{(0)}}$.

This polynomial is in fact strictly quadratic, because each $b_r^2 = b_r$ as $b_r \in \{0, 1\}$. Thus we have effectively mapped the least squares cost onto a QUBO:

$$\mathbf{b^{(1)*}} \;=\; \min_{\mathbf{b}\in\{0,1\}^{d\times m}} \;\; \alpha^{(1)} \;+\; \sum_{r=1}^{dm} \gamma_r^{(1)}\, b_r \;+\; \sum_{1\,\le\, r<s\,\le\, dm} \Gamma_{r,s}^{(1)}\, b_r\, b_s. \tag{15}$$

Given Eq. 15, a quantum annealer can directly solve QUBO problems by mapping them onto its native hardware architecture, where the cost function is minimized adiabatically through quantum tunneling. The annealer attempts to find a low-energy configuration of the binary variables that corresponds to the optimal solution of the given problem. However, the total number of binary variables that can be encoded is constrained by the available qubits, making it challenging to achieve high precision (i.e., large m) in naive discretization, as each additional bit per weight significantly increases qubit requirements.

4.3 i-QLS Algorithm

To address the qubit-precision trade-off, we propose an iterative zoom-in approach that uses a small number of bits $m \geq 1$ per weight but refines the solution region over multiple iterations. Let $\ell_i^{(0)}$ and $u_i^{(0)}$ be the initial bounds for weight w_i. At iteration $k = 1, \ldots, K$, each weight w_i is represented in $[\ell_i^{(k-1)}, u_i^{(k-1)}]$ with m bits. With $\Delta_i^{(k-1)} = u_i^{(k-1)} - \ell_i^{(k-1)}$, the discretization step size is evaluated as

$$\delta_i^{(k-1)} \;=\; \frac{\Delta_i^{(k-1)}}{2^m - 1}.$$

Thus,

$$\mathbf{w^{(k)}}(\mathbf{b}) \;=\; \boldsymbol{\ell^{(k-1)}} \;+\; \mathbf{D^{(k-1)}}\mathbf{B}(\mathbf{b})$$

Construct the QUBO from

$$S^{(k)}(\mathbf{b}) \;=\; \sum_{i=1}^{N} \big(y_i - \mathbf{w^{(k)}}(\mathbf{b})^\top \mathbf{x}_i\big)^2$$

with the updated bounds. Simplify the expression so that powers of binary variables are reduced via $b_{j,r}^2 = b_{j,r}$. Solve the QUBO using a quantum annealer. Retrieve the optimal binary solution $\mathbf{b}^{(k)*}$. Compute

$$w_i^{(k)*} \;=\; w_i(\mathbf{b}_i^{(k)*}), \quad i = 1, \ldots, d.$$

For each i:

$$\ell_i^{(k)} \;=\; w_i^{(k)*} \;-\; (\frac{\delta_i^{(k-1)}}{2f(m)}). \quad u_i^{(k)} \;=\; w_i^{(k)*} \;+\; (\frac{\delta_i^{(k-1)}}{2f(m)}),$$
$$\text{where} \quad f(x) = \begin{cases} 2, & \text{if } x = 1, \\ 1, & \text{otherwise.} \end{cases} \tag{16}$$

This ensures the next iteration's search space is an interval of width $\Delta_i^{(k)}$ centered at the best estimate $w_i^{(k)*}$. Observe that, for $m = 1$, $\delta_i^k = \Delta_i^k \implies w_i^{(k)} \in \{\ell_i^{(k-1)}, u_i^{(k-1)}\}$, thus the step function $f(x)$ helps in shrinking the search space exponentially. Repeat until the maximum number of iterations K is reached, or until a convergence criterion is met (e.g., changes in the loss below a threshold).

Algorithm 1: i-QLS: Iterative Quantum-Assisted Least Squares

Input: $\mathbf{X} \in \mathbb{R}^{N \times d}$ (feature matrix), $\mathbf{y} \in \mathbb{R}^N$ (target values), bits per weight m, max iterations K, initial bounds $\ell_i^{(0)}, u_i^{(0)}$ for each $i = 1, \ldots, d$.

for $k = 1$ **to** K **do**

- Compute step size: $\delta_i^{(k-1)} \leftarrow \frac{u_i^{(k-1)} - \ell_i^{(k-1)}}{2^m - 1},\ i = 1, \ldots, d.$
- Construct weight representation $w_i^{(k)}(\mathbf{b})$ (Eq. 10)
- Formulate QUBO cost function $S^{(k)}(\mathbf{b})$.
- Solve $\min S^{(k)}(\mathbf{b})$ using a quantum annealer.
- Extract optimal solution: $\mathbf{b}^{(k)*} \leftarrow \arg\min_{\mathbf{b}} S^{(k)}(\mathbf{b})$.
- Compute updated weight estimates: $w_i^* \leftarrow w_i(\mathbf{b}^{(k)*})$.
- Update bounds: $\ell_i^{(k)} \leftarrow w_i^{(k)*} - \frac{1}{2f(m)}\delta_i^{(k)}, \quad u_i^{(k)} \leftarrow w_i^{(k)*} + \frac{1}{2f(m)}\delta_i^{(k)}.$

Output: $\mathbf{w}^{(K)*} = (w_1^{(K)*}, \ldots, w_d^{(K)*})$ (optimal weights).

This procedure scales linearly with d in qubit usage (only $d \times m$ qubits are needed), yet allows an effective exponential zoom-in on the candidate solution region over multiple iterations, thereby improving precision without globally increasing the qubit count. The pseudocode of the algorithm is given in 1.

Lemma 1. *If the underlying least squares problem is well-posed (i.e., there exists a unique optimal weight vector w^* such that $y = Xw^*$) and $w_i^* \in [l_i^{(0)}, u_i^{(0)}] \forall i$, then the mean squared error evaluated from the weights estimated at each iteration exponentially converges to 0 as $k \to \infty$.*

Proof. At iteration $k = 0$, the search interval for weight w_i is given by $[l_i^{(0)}, u_i^{(0)}]$ with width $\Delta_i^{(0)}$. In the first iteration, the algorithm discretizes this interval into 2^m equally spaced values. Let the discretization step size be

$$\delta_i^{(0)} = \frac{\Delta_i^{(0)}}{2^m - 1}.$$

The QUBO formulation selects the discrete value closest to the true optimum, denoted by $w_i^{(1)}$. The algorithm then updates the search interval to be centered at $w_i^{(1)}$ with new bounds

$$l_i^{(1)} = w_i^{(1)} - \frac{\delta_i^{(0)}}{2f(m)}, \quad u_i^{(1)} = w_i^{(1)} + \frac{\delta_i^{(0)}}{2f(m)},$$

so that the new interval width is

$$\Delta_i^{(1)} = \frac{\delta_i^{(0)}}{f(m)} = \frac{\Delta_i^{(0)}}{f(m)(2^m - 1)}.$$

By repeating this process iteratively, the width of the interval after k iterations is

$$\Delta_i^{(k)} = \frac{\Delta_i^{(k-1)}}{f(m)(2^m - 1)} = \frac{\Delta_i^{(0)}}{(f(m)(2^m - 1))^k}. \tag{17}$$

Since $f(m)(2^m - 1) > 1$ for any $m \geq 1$, it follows that

$$\lim_{k\to\infty} \Delta_i^{(k)} = 0.$$

Because the search interval shrinks to a single point, the weight estimate $w_i^{(k)}$ converges to a unique value, which must coincide with the optimal weight w_i^* provided the model is correctly specified.

Furthermore, since the mean squared error is a continuous function of the weights, it follows that

$$\lim_{k\to\infty} E^{(k)} = \lim_{k\to\infty} \|Xw^{(k)} - y\|^2 = \|Xw^* - y\|^2.$$

Under the assumption that the least squares problem is consistent (or that w^* minimizes the error), we have $\|Xw^* - y\|^2 = 0$, which implies

$$\lim_{k\to\infty} E^{(k)} = 0.$$

5 Experiments

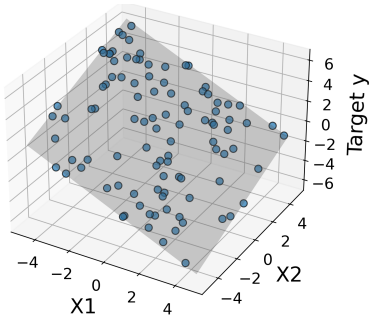
In this section, we present a series of experiments to evaluate various aspects of i-QLS. Specifically, the first set of experiments examines the convergence rate and the accuracy of thee algorithm, analyzing how the number of bits used for weight representation and the number of iterations influence these two factors.

The second set of experiments evaluates the performance of i-QLS in terms of scalability on a multivariate linear regression problem. We demonstrate that i-QLS overcomes the scalability limitations of existing quantum approaches by successfully handling datasets with up to 175 features. Our results indicate that the performance of i-QLS is comparable to classical methods while outperforming all previously proposed quantum annealing-based techniques.

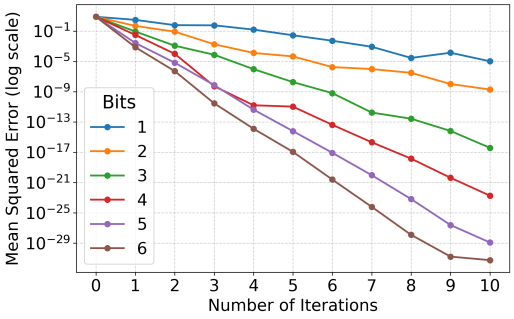
Finally, we extend our analysis beyond linear regression by applying i-QLS to nonlinear function estimation using the least squares formulation of spline functions. In this case, the algorithm achieves significantly higher precision in weight estimation compared to existing gate-based quantum spline methods, even with a relatively low number of iterations.

5.1 Convergence Rate and Accuracy

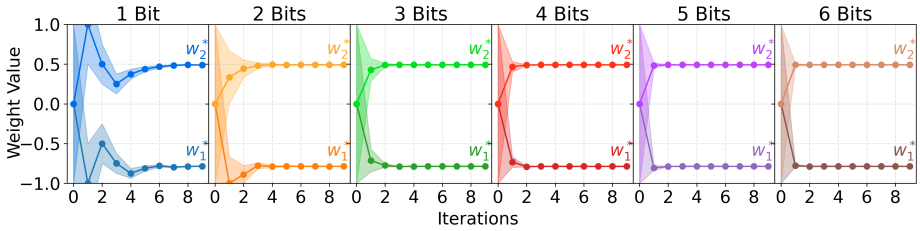
In this section, we demonstrate our iterative quantum-assisted least squares algorithm on a synthetic linear regression problem with two features. The goal is to illustrate the exponential rate of convergence of the mean squared error (MSE) across iterations, along with the exponential refinement of the weights' search space. We generate synthetic data (Figure 1a) with two input features, sampling each feature value x_i uniformly from $[-5, 5]$. The corresponding target values are computed using a linear model ($y_i = w_1^* x_{i1} + w_2^* x_{i2}$) where the true weights w_1^* and w_2^* were set to fixed values within $(-1, 1)$.



(a) **Two-Feature Linear Data.** 100 data points (x_i, y_i) where $x_i \in [-5, 5]^2$ sampled from a linear model.



(b) **MSE Convergence Over Iterations.** Exponential decay of MSE with iterations, improving with increasing bits per weight.



(c) **Exponential Shrinking of the Search Space.** The weight search space for w_1 and w_2 narrows exponentially with iterations. Higher bit precision enables finer updates, accelerating convergence.

Fig. 1: Assessment of convergence rate and accuracy of i-QLS

We use i-QLS to find the optimal set of weights w_1^* and w_2^*. Initially, each weight is discretized into 2^m equally spaced values in a sufficiently broad range (we initialized $[-1, 1]$). At each iteration k, we solve the corresponding QUBO using a D-Wave Advantage annealer, extract the best candidate $(w_1^{(k)*}, w_2^{(k)*})$, and then refine the search interval around it (Eq. 16). This process ensures that, the search space shrinks exponentially by a factor of $f(m)(2^m - 1)$ (Eq. 17).

Figure 1b plots the MSE per iteration for different bit-precisions b. The logarithmic scale on the vertical axis reveals a linear decline, which corresponds to exponential convergence in the original scale. Increasing m (i.e., using more qubits per weight) accelerates convergence. For example, at iteration $k = 9$, the 6-bit case achieves MSE ≈ 0 (within machine precision using `float64`), whereas the 2-bit case remains around 10^{-9}. This follows our theoretical prediction that finer discretization improves solution precision but at the cost of additional quantum resources.

Figure 1c shows how the search space bounds for w_1 and w_2 evolve over iterations. The shaded regions indicate the possible value ranges, which shrink exponentially with each iteration. Higher bit precision results in faster contrac-

tion of the search space, leading to earlier convergence to w_1^* and w_2^*. This aligns with our theoretical framework (Lemma 1), which predicts that the bounds contract by a factor of $f(m)(2^m - 1)$ per iteration.

5.2 Scalability Analysis

In this experiment, we investigate how i-QLS scales with the number of features in a linear regression setting. We fix the number of bits per weight to 1 and the maximum number of iterations to 10. We generate synthetic data in the same manner as Section 5.1, but now the number of features d ranges from 1 to 175. Specifically, for each d, we draw each feature value x_{ij} uniformly from $[-5, 5]$ and compute the target values via

$$y_i \;=\; \sum_{j=1}^{d} w_i^*\, x_{ij},$$

where the true weights w_i^* are sampled from $(-1, 1)$. At each iteration of i-QLS, a QUBO problem of size proportional to the number of features d is defined, discretizing each weight into $2^1 = 2$ equally spaced values in a broad interval (here we chose $[-1, 1]$). This QUBO is solved on a D-Wave Advantage quantum annealer accessed remotely, employing a default embedding strategy, which is heuristic and may produce different embeddings for repeated runs on identical problems. For a classical QUBO solver, we use Gurobi, a state-of-the-art branch-and-bound mixed-integer optimizer. Additionally, to compare with classical linear regression, we employ scikit-learn's `LinearRegression`, which by default uses an efficient singular value decomposition (SVD)-based method. This makes it highly competitive for moderate to large d.

The results in Figure 1 highlight the trade-off between quantum hardware constraints and classical solver limitations. D-Wave Advantage (Quantum Annealer) efficiently handles problems up to $d = 175$, yielding MSE values near 10^{-1} for larger d. This is, to our knowledge, the first demonstration of linear regression on a real quantum annealer scaling to 175 features while maintaining high accuracy. The inconsistencies of minor embedding and remote access on the performance of the quantum annealer for certain instances can be seen as spikes in Figure 2. Gurobi's branch-and-bound algorithm struggles beyond $d = 50$ due to the fully connected QUBO (the number of pairwise interactions is equal to $\frac{d(d-1)}{2}$). Consequently, Gurobi's runtime becomes prohibitively large, and the solver occasionally terminates with suboptimal solutions or runs out of memory. Scikit-learn remains highly accurate (MSE $\approx 10^{-28}$) and efficiently solves these linear regressions for d beyond 175.

5.3 Non-Linear Function Approximation Using Splines

In this section, we use i-QLS to non-linear function approximation by leveraging the LS optimization of splines provided in Eq. (4). We perform similar

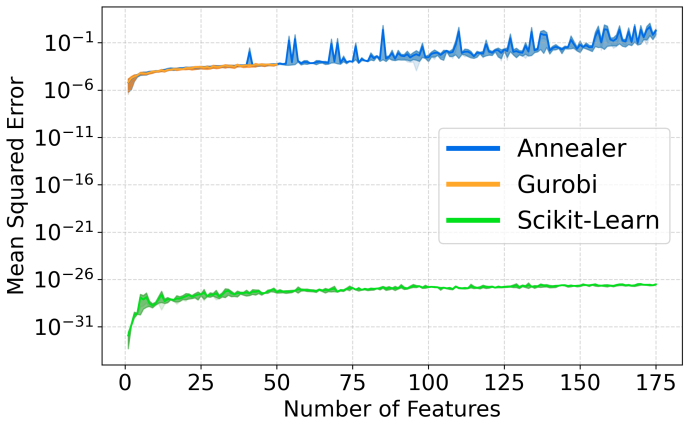


Fig. 2: MSE at iteration $k = 10$ for i-QLS executed on D-Wave Advantage, Gurobi, and scikit-learn's `LinearRegression`. Shaded regions indicate variance (darker) and min-max range (lighter) over three random seeds.

d	$\mathbf{R^2}$
1	0.999943
25	0.999988
50	0.999990
75	0.999987
100	0.999987
125	0.999923
150	0.999627
175	0.998592

Table 1: R^2 values for the fit generated by our quantum-assisted approach across different numbers of features.

experiments to existing quantum splines [15,11] generating synthetic datasets from five widely used non-linear functions in deep learning: (1) the sine function $\sin(\pi x)$, which is periodic and frequently appears in Fourier analysis and wave modeling; (2) the sigmoid function $\sigma(\pi x) = (1 + e^{-\pi x})^{-1}$, commonly used in neural networks for binary classification; (3) the hyperbolic tangent $\tanh(\pi x)$, which serves as a popular activation function for hidden layers; (4) the ReLU (Rectified Linear Unit) $\max(0, \pi x)$, a piecewise linear function that underpins many modern deep networks; and (5) the ELU (Exponential Linear Unit), an extension of ReLU that smooths the transition for negative inputs, thereby improving optimization stability. For each function, we generated 100 data points uniformly in the range $x \in [-1, 1]$. Given these target highly non-linear functions, we define a LS optimization problem using spline functions (Section 3 with 20 knots. Thus, we estimate the optimal parameters of the non-linear regression using i-QLS. The optimization is performed by discretizing the solution space, assigning one qubit per parameter, and refining the weight estimates iteratively for up to 10 iterations.

Figure 3 presents the results of our spline-based approximations. The blue scatter points indicate the ground-truth function values, while the green curves represent the fits obtained by estimating the parameters of the splines using i-QLS. Due to the choice of a linear function in each interval of the splines, the approximation is piecewise linear, generating a recurrent angular pattern in the function representation. This effect arises from the limited number of knots (20), which results in a segmented approximation of the underlying function.

While we demonstrated the capability of i-QLS to model non-linear functions, more expressive spline models—such as *quadratic* or *cubic splines*—could further improve smoothness and accuracy. However, these higher-degree splines

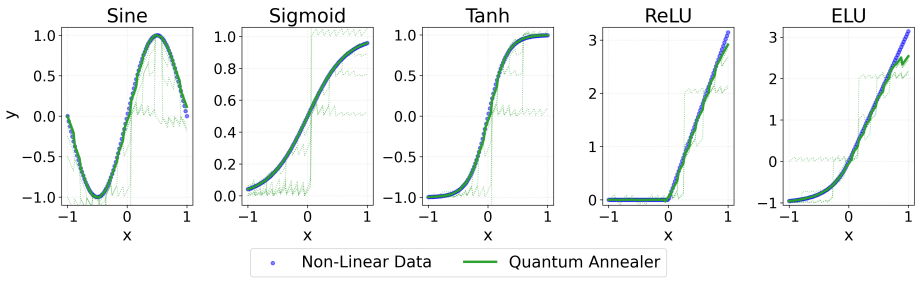


Fig. 3: **Spline-Based Approximation of Non-Linear Functions.** The green curves illustrate the iterative refinement of the regression fit obtained using our quantum-assisted least squares approach with *linear splines* (20 knots), one qubit per parameter, and up to 10 iterations. The lighter green lines correspond to intermediate fits across iterations, while the darker green curve represents the final approximation after 10 iterations.

introduce additional computational overhead, requiring more qubits, which must be accounted for in practical quantum implementations.

By integrating splines with our iterative refinement strategy, we demonstrated how quantum-assisted optimization can be extended beyond traditional linear models, offering a scalable path towards non-linear function learning on quantum hardware.

6 Discussion and Conclusion

In this work, we proposed i-QLS, a novel iterative quantum-assisted least squares optimization algorithm that exhibits favorable scaling compared to prior single-shot QUBO formulations. The key advantage stems from the iterative refinement mechanism, which systematically narrows the search space around the best-found weight values at each step. Given that the search space contracts by a factor of $f(m)(2^m - 1)$ per iteration, the number of iterations K required to achieve a given precision ϵ scales as:

$$K = \mathcal{O}\left(\log_{f(m)(2^m-1)} \frac{1}{\epsilon}\right),$$

where m is the number of bits allocated per weight. This implies that, with higher bit precision, convergence is achieved in fewer iterations. However, as observed in Figure 1b, increasing m requires additional quantum resources, introducing a trade-off between iteration count and hardware feasibility. Nevertheless, the approach theoretically guarantees to find the optimal solution even with $m = 1$ as $k \to \infty$, making it inherently more scalable.

As a well-rounded analysis of the approach, considering practical hardware constraints, we acknowledge that if an iteration selects a weight far from the true optimal, restricting the subsequent shrunk search space from containing the true optimal weight, successive iterations may struggle to recover. This issue

is particularly relevant when practical quantum hardware noise or solver limitations prevent finding the best candidate in an iteration. Also, our method demonstrates strong scalability in terms of problem size, the practical runtime depends heavily on hardware access constraints. As the annealer is remotely accessed and shared among many users, the time to solution for each QUBO problem is primarily dominated by network latency and queue wait times rather than raw annealing time [21]. Consequently, having the annealer on-site could significantly reduce overhead and make i-QLS a highly competitive method for real-time applications.

The weight chosen at step k might be nearer to the optimal weight than the weight chosen at step $k+1$, but the search space would have shrunk by a factor of $f(m)(2^m - 1)$. This effect is exacerbated when fewer qubits per weight are used, as demonstrated in Figure 1b for the 1-bit case. From iteration 8 to 9, the MSE temporarily increases instead of decreasing. This occurs because the selected weight at iteration 8 was closer to the optimal than any available discretized value at iteration 9. However, the MSE resumes decreasing at iteration 10, illustrating that while local fluctuations may arise, the overall convergence trend remains intact. Thus, one can eventually mitigate this issue by running a sufficiently large number of iterations.

Empirical validation on the D-Wave quantum annealer demonstrated the effectiveness of i-QLS, showing that it scales efficiently to problems with up to 175 features while maintaining high accuracy. Additionally, we extended our framework to non-linear regression using spline-based modeling, demonstrating its adaptability beyond linear problems.

While our approach significantly improves existing quantum-assisted regression, practical implementations against classical solvers must carefully balance bit precision, iteration count, and hardware limitations to ensure robust performance. Future work will explore adaptive bit precision per iteration and weight to further enhance efficiency and accuracy.

Code Availability. The code associated with this paper is available at: https://github.com/supreethmv/i-QLS

Acknowledgments. This work has been partially funded by the German Ministry for Education and Research (BMB+F) in the project QAIAC-QAI2C under grant 13N17167.

References

1. Borle, A., Lomonaco, S.J.: Analyzing the quantum annealing approach for solving linear least squares problems. In: Das, G.K., Mandal, P.S., Mukhopadhyaya, K., Nakano, S.i. (eds.) WALCOM: Algorithms and Computation. pp. 289–301. Springer International Publishing, Cham (2019)
2. Cortes, C., Vapnik, V.: Support-vector networks. Machine learning **20**, 273–297 (1995)
3. Cruz-Santos, W., Venegas-Andraca, S.E., Lanzagorta, M.: A qubo formulation of minimum multicut problem instances in trees for d-wave quantum annealers. Scientific Reports **9**(1), 17216 (Nov 2019). https://doi.org/10.1038/s41598-019-53585-5

4. Date, P., Potok, T.: Adiabatic quantum linear regression. Scientific Reports **11**(1), 21905 (Nov 2021). https://doi.org/10.1038/s41598-021-01445-6
5. De Boor, C., De Boor, C.: A practical guide to splines, vol. 27. springer New York (1978)
6. Francis, J.G.F.: The qr transformation a unitary analogue to the lr transformation—part 1. The Computer Journal **4**(3), 265–271 (01 1961). https://doi.org/10.1093/comjnl/4.3.265
7. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016), http://www.deeplearningbook.org
8. Haji, S.H., Abdulazeez, A.M.: Comparison of optimization techniques based on gradient descent algorithm: A review. PalArch's Journal of Archaeology of Egypt/Egyptology **18**(4), 2715–2743 (2021)
9. Harrow, A.W., Hassidim, A., Lloyd, S.: Quantum algorithm for linear systems of equations. Physical review letters **103**(15), 150502 (2009)
10. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer Series in Statistics, Springer New York Inc., New York, NY, USA (2001)
11. Inajetovic, M.A., Orazi, F., Macaluso, A., Lodi, S., Sartori, C.: Enabling non-linear quantum operations through variational quantum splines. In: International Conference on Computational Science. pp. 177–192. Springer (2023)
12. Jerbi, S., Gyurik, C., Marshall, S.C., Molteni, R., Dunjko, V.: Shadows of quantum machine learning. Nature Communications **15**(1), 5676 (2024)
13. Lanczos, C.: Solution of systems of linear equations by minimized iterations. J. Res. Nat. Bur. Standards **49**(1), 33–53 (1952)
14. Macaluso, A.: Quantum supervised learning. KI-Künstliche Intelligenz pp. 1–15 (2024)
15. Macaluso, A., Clissa, L., Lodi, S., Sartori, C.: Quantum splines for non-linear approximations. In: Proceedings of the 17th ACM International Conference on Computing Frontiers. pp. 249–252 (2020)
16. Moon, T.K.: The expectation-maximization algorithm. IEEE Signal processing magazine **13**(6), 47–60 (1996)
17. Moré, J.J., Sorensen, D.C.: Newton's method. Tech. rep., Argonne National Lab.(ANL), Argonne, IL (United States) (1982)
18. Mysore Venkatesh, S., Macaluso, A., Klusch, M.: Gcs-q: Quantum graph coalition structure generation. In: Mikyška, J., de Mulatier, C., Paszynski, M., Krzhizhanovskaya, V.V., Dongarra, J.J., Sloot, P.M. (eds.) Computational Science – ICCS 2023. pp. 138–152. Springer Nature Switzerland, Cham (2023)
19. Reinsch, C.H.: Smoothing by spline functions. Numerische mathematik **10**(3), 177–183 (1967)
20. Sinha, A., Macaluso, A., Klusch, M.: Nav-q: quantum deep reinforcement learning for collision-free navigation of self-driving cars. Quantum Machine Intelligence **7**(1), 1–20 (2025)
21. Venkatesh, S.M., Macaluso, A., Nuske, M., Klusch, M., Dengel, A.: Q-seg: Quantum annealing-based unsupervised image segmentation. IEEE Computer Graphics and Applications **44**(5), 27–39 (2024). https://doi.org/10.1109/MCG.2024.3455012
22. Woodbury, M.A.: Inverting modified matrices. Department of Statistics, Princeton University (1950)