Variational Quantum Regression applied to Computer Graphics

Diogo Ramos, MEFIS
University of Minho, Portugal
Email: pg53771@uminho.pt

Gabriel Costa, MEFIS
university of Minho, Portugal
Email: pg53828@uminho.pt

Abstract

This paper explores the potential of quantum machine learning (QML) in computer graphics by exploring it's use to compute BRDF. The results demonstrate the viability of QML, showcasing its ability to achieve promising performance in machine learning tasks.

Keywords— QML, BRDF, VQP, Regression, Fitting, Lambert, Phong

1 Introduction

In computer graphics, creating realistic images involves a process called rendering. A key ingredient for achieving this realism is accurately representing the materials that make up the objects in a scene. Materials have specific properties that determine how light interacts with them. In other words, how much light bounces off the material in one direction depends on the direction the light hit it from.

This complex relationship between incoming and outgoing light directions is captured by a special function called the Bidirectional Reflectance Distribution Function (BRDF). By understanding and incorporating BRDFs into rendering, we can create images that look incredibly lifelike

The BRDF takes into account the incoming light direction ω_i and the outgoing direction ω_r, returning the ratio of reflected radiance exiting along ω_r to the irradiance incident on the surface from the incoming direction. Each direction ω is given by two angles, $\theta \in [0,\frac{\pi}{2}]$ and $\phi \in [0,2\pi]$, representing the elevation and orientation angles, respectively.

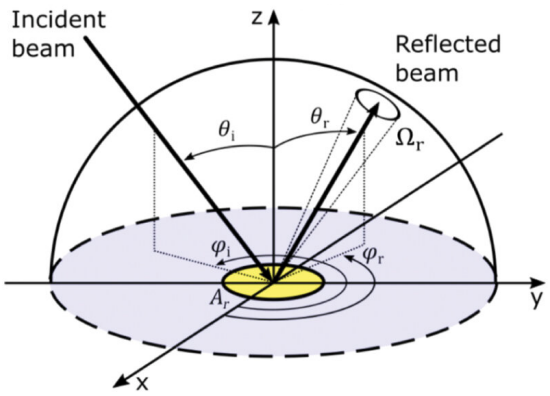


Figure 1: BRDF in an hemisphere

The simplest analytical models for BRDFs are the Lambert model and the Phong model, both shown in the following table:

BRDF	$f_r(\theta_i,\phi_i,\theta_r,\phi_r)$
Lambert	$k_d * \cos\theta_i$
Phong	$k_d * \cos\theta_i + k_s * (\cos(\omega_{\text{ref}},\omega_r))^n$ with $\cos(\omega_{\text{ref}},\omega_r) = \sin\theta_i \sin\theta_r \cos(\phi_i - \phi_r) - \cos\theta_i \cos\theta_r$

Table 1: Simple analytical models for BRDFs

where k_s, is a specular reflection constant (the ratio of reflection of the specular term of incoming light), k_d is a diffuse reflection constant (the ratio of reflection of the diffuse term of incoming light) and n is the shininess constant for the material.

In computer graphics, achieving photorealistic images relies heavily on accurately representing the materials that compose a scene. The BRDF plays a crucial role in this process.

However, simulating light transport through complex materials with traditional computers can be computationally expensive. That's where quantum computers can shine, offering a potential solution with their ability to perform calculations in parallel.

Quantum computation leverages the principles of quantum mechanics, such as superposition and entanglement, to offer computational capabilities that can surpass classical approaches. Quantum machine learning (QML) merges these quantum advantages with machine learning techniques, aiming to develop algorithms that potentially outperform classical counterparts in tasks like classification, regression, and optimization.

2 Theoretical Background

The generic process of Quantum Machine Learning (QML) algorithm is the following: encode the data, apply a variational quantum algorithm[1] (ansatz) with tunnable gates, measure the output and utilize this measurement to tune the parameters of the gates present in the ansatz.

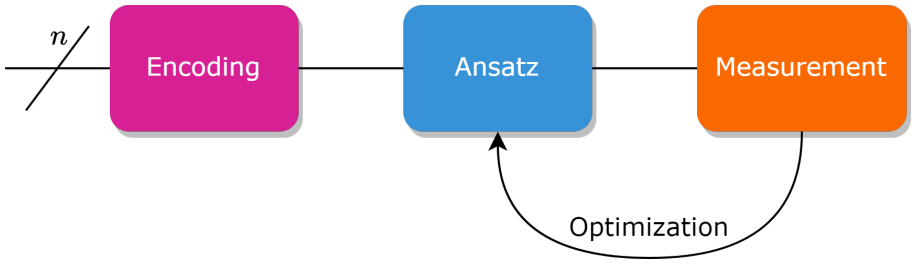


Figure 2: General QML algorithm

2.1 Encoding the data and creating the Ansatz

There are multiple ways of encoding data to a quantum circuit: angle encoding, amplitude encoding, IQP, etc.

In angle encoding, a data point x with M features is encoded to the circuit by applying a $R_y(x_i)$ gate to each qubit q_i of the circuit. As such, each parameter will be encoded as a phase in a circuit with M qubits. Furthermore, for this encoding to be optimal, the data must initially be normalized, in order to only have values in the interval $]0,2\pi[$.

In amplitude encoding we utilize the amplitude of the basis states to represent our datapoints, in the case of a datapoint x of dimension N with M features we need $N \times M$ values in order to fully represent it which means that we need $\lceil \log_2(N \times M) \rceil$ qubits. One other important thing to note is that as we are essentially building quantum states there is a need for our datapoints to be normalized.

We can further expand this strategy by introducing data re-uploading. Data re-upload consists of encoding data to our circuit via an encoding technique, each we will call $S(\vec{x})$, followed by the application of tunnable gates $W(\vec{\theta})$, which represent the ansatz. This process is then repeated several times, with various tunnable parameters $\vec{\theta_i}$ on the different tunnable gates throughout the repetitions.

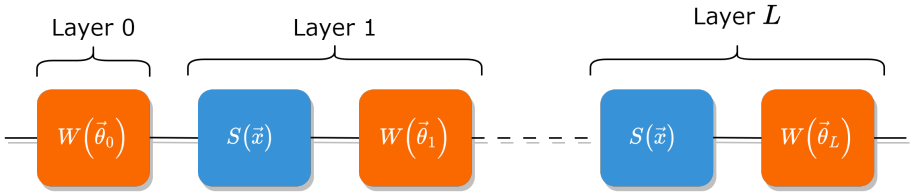


Figure 3: Data Re-uploading Ansatz

2.2 Data re-uploading

The goal of a QML fitting algorithm is to train tunnable parameters of a circuit ansatz in order to reproduce a target function $f(\vec{x})$.

The quantum model for data re-uploading can be written as

$$f_\theta(\vec{x}) = \langle 0|U^\dagger(\vec{x},\vec{\theta})OU(\vec{x},\vec{\theta})|0\rangle \tag{1}$$

where $U(\vec{x},\vec{\theta})$ represents the circuit previously described and O a given observable. Furthermore, we can also see from [3], that a quantum model can represent a function of the form:

$$f_\theta(\vec{x}) = \sum_{\omega\in\Omega} c_\omega(\vec{\theta})e^{i\omega x}, \tag{2}$$

which represents a Fourier series.

For L layers, the ansatz $U(\vec{x},\vec{\theta})$ can be written as:

$$U(\vec{x},\vec{\theta}) = W^{(L)}S(\vec{x})W^{(L-1)}\dots W^{(1)}S(\vec{x})W^{(0)} \tag{3}$$

where $S(\vec{x})$ represents the encoding gates and $W^{(i)}$ the gates $W^{(\vec{\theta})}$ containing the tunnable parameters of the i-th layer.

From ref. [3], we see that a circuit of this form, with L re-uploads, corresponds to a polynomial of, at maximum, degree L.

We can now see that data re-uploading has the potential to allow for more expressive quantum models, $f_\theta(\vec{x})$, since it allows a wider support. Expressivity, however, does not depend only on the support spectrum, but also on the values of the coefficients, which depend on the ansatz.

3 Encoding and Ansatz utilized

Both the models we are trying to fit can be represented by cosines and sums of cosines of different frequencies. As such, the approach followed was to use data re-uploading to fit both of the models (Lambert and Phong), adjusting the number of layers according to the number of frequencies required to represent such models.

Even though there are multiple ways of encoding data, the strategy we followed was to encode the data as rotations, since in both of our models (Lambert and Phong) the input values are always values between 0 and 2π. The only problem from using this encoding would appear if the value of n was greater than 6, which will not be the case for our models.

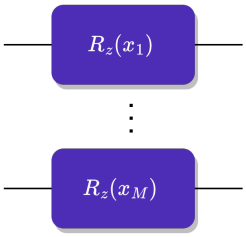


Figure 4: Encoding for a dataset with M features

For the ansatz we must take into account that a generic gate can be represented by 3 different angles:

$$G(\theta,\phi,\beta) = R_z(\theta)R_Y(\phi)R_z(\beta) \tag{4}$$

However, since adding gates introduces noise, which will lead to barren plateaus[4] and consequently to a non-trainable circuit, we will only use the angles θ and ϕ as trainable parameters, fixing the value of β at 0.

With these information, we will use as an ansatz the following circuit:

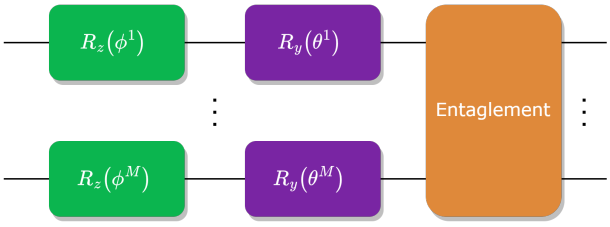


Figure 5: Ansatz utilized

followed by entanglement between every qubit using controlled-NOT gates, represented by the "Entanglement" gate.

With both the encoding and ansatz implemented, we can now build our circuit with data re-uploading, implementing each of these blocks throughout the successive layers.

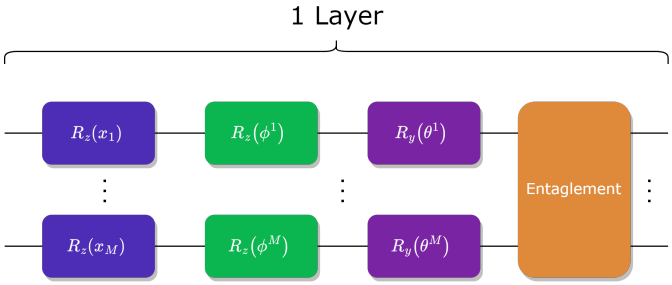


Figure 6: 1 repetition of the circuit

4 Fitting for Lambert's model

As shown in table 1, the Lambert model is the simplest way to represent a BRDF.

Since it's only given by a cosine, alongside a multiplicative factor (diffuse reflection constant), constant for a given material, we decided to implement the circuit with only 1 layer, since introducing more layers would stray the resulting function from a pure cosine.

Furthermore, we decided to train two different models. One with a singular qubit, learning to recreate a cosine from the data, and later multiplying the resulting function by the diffuse constant (k_d), and another with 2 qubits, one for the θ_i and another for k_d.

4.1 1-qubit circuit

For this first implementation, since the objective was to learn to represent a cosine, the following 1-qubit circuit was built:

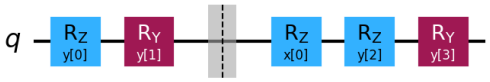


Figure 7: 1-qubit Circuit for Lambert model

As for the cost function, we decided to use one similar to a mean squared error function. However, since we are fitting with results obtained directly from the Lambert model, hence scaled by k_d, the value obtained from the circuit must also be scaled by k_d:

$$C(\vec{\theta}) = \sum_{i=1}^{N} (g(x_i) - k_d \times f_{\vec{\theta}}(x_i))^2 \tag{5}$$

where $g(\vec{x})$ represents the function we are trying to reproduce and N the number of points. Using the SPSA[2] optimizer with 200 training points and 200 iterations, the values obtained were the following.

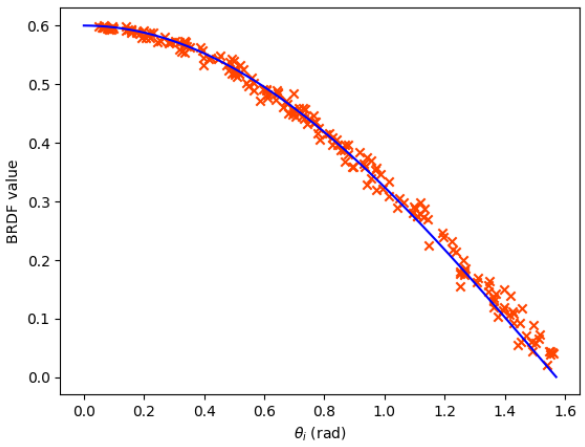


Figure 8: Results for the 1-qubit circuit

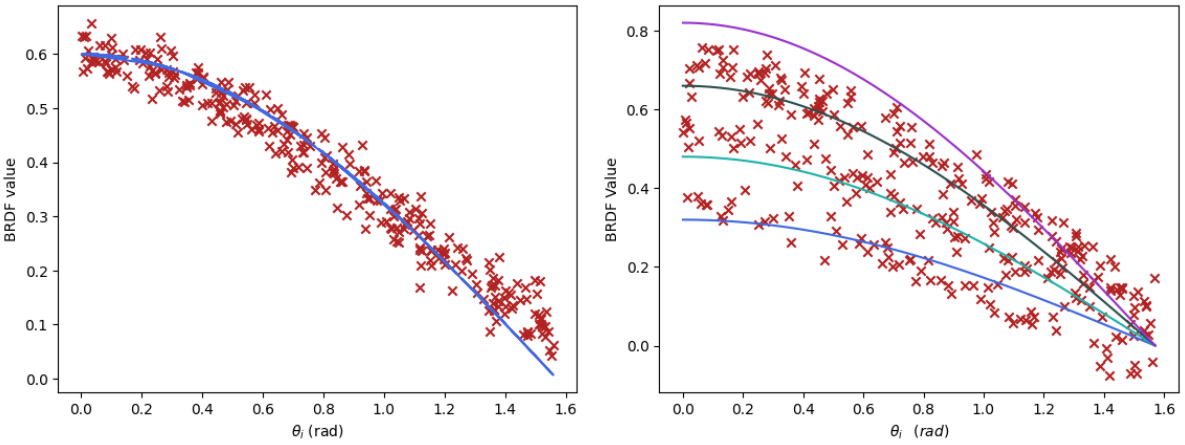
4.2 2-qubit circuit

For this second implementation, the circuit would be fed with both θ_i and k_d, so it could learn the values and weights of each one of them. For this reason the cost function doesn't need the value of k_d anymore, being given by a simple mean squared error function:

$$C(\vec{\theta}) = \sum_{i=1}^{N} (g(x_i) - f_{\vec{\theta}}(x_i))^2 \tag{6}$$

However, utilizing the same circuit with the same 1 re-upload wasn't enough to discriminate between values of k_d. For that reason, for the circuit fed with multiple values of k_d, 4 layers were utilized instead.

Running for the same number of training data, one with a fixed value of k_d (= 0.6) and another with multiple values of k_d, and utilizing the same optimizer with the same 200 iterations, the results were the following:



(a) Results using the circuit fed with a fixed k_d (b) Results for the circuit fed with multiple values of k_d

Figure 9: Results for training with fixed k_d versus variable k_d

5 Fitting for Phong's model

Phong's model can easily be separated into two different circuits: one fitting for the diffuse term and another for the specular term. Since the diffuse term is just the Lambert model, we make use of the previously trained model for Lambert as a drop-in addon for Phong's model.

On the other hand, the specular term is a lot different from the previous described model, and is described as:

$$f_r^{specular}(\theta_i,\theta_r,\phi_i,\phi_r) = k_s \times (\cos(\omega_{\text{ref}},\omega_r))^n \tag{7}$$

where

$$\cos(\omega_{\text{ref}},\omega_r) = \sin\theta_i \sin\theta_r \cos(\phi_i - \phi_r) - \cos\theta_i \cos\theta_r$$

For reasons we'll discuss later, the value of $(\omega_{\text{ref}},\omega_r)$ was calculated separately, and then fed to the circuit alongside the value of n. For the value of k_s, we followed the strategy applied in the fitting of Lambert's model, and decided to keep it as a multiplicative constant.

With this approach in mind, the loss function was, once again, a mean squared error function, but with a multiplicative constant (k_s), on the value returned by the measurement of the circuit.

$$C(\vec{\theta}) = \sum_{i=1}^{N} (g(x_i) - k_s * f_{\vec{\theta}}(x_i))^2 \tag{8}$$

5.1 Fitting for a fixed value of n

On a first approach we decided to train the circuit by feeding a fixed value of n (in our cases two circuits with $n = 3$ and $n = 4$), since if the model could not fit the data for this approach, it would hardly fit for a approach were multiple values of n would be given.

The circuits were built using 5 layers for $n = 3$ and 6 layers for $n = 4$. For the 6 layers, the circuit ended up having the following form:

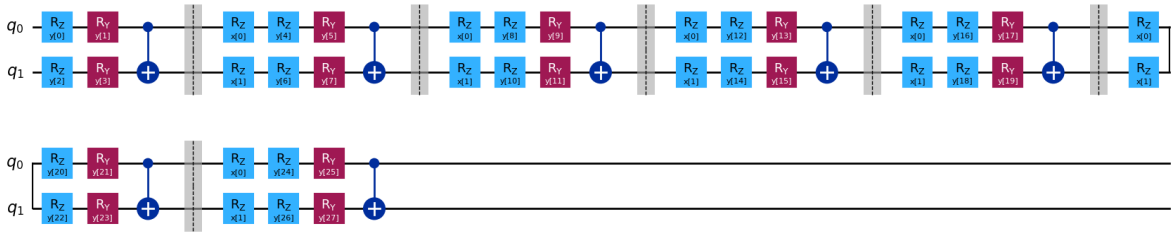
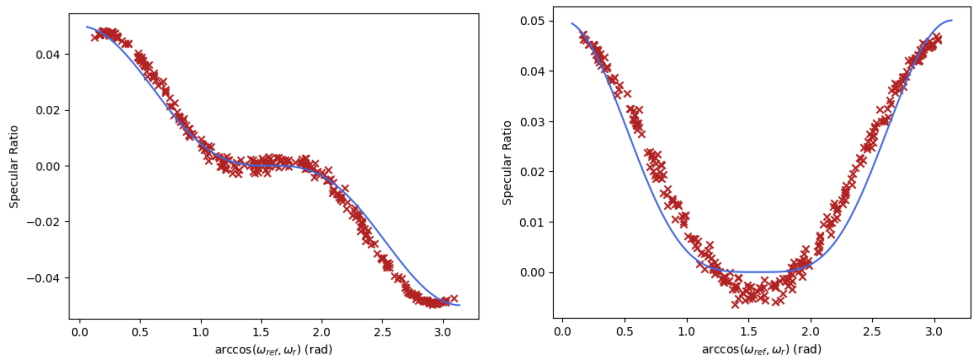


Figure 10: Circuit for the 2-qubit Phong's model

For these two circuits, using the SPSA optimizer, with 400 training points, the results obtained were the following:

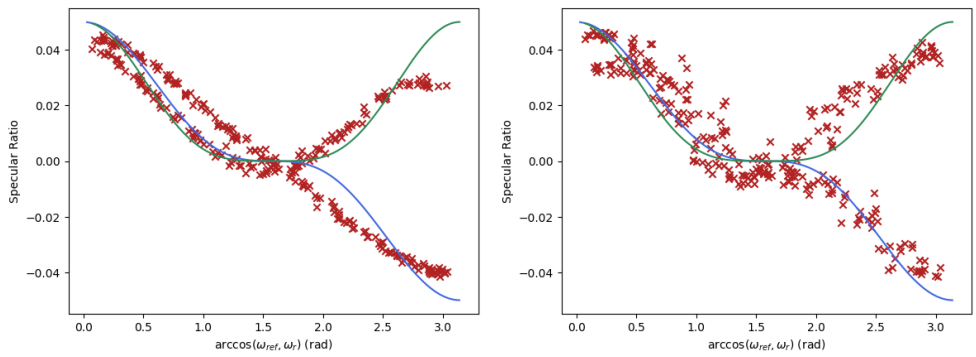


(a) Training results obtained for $n = 3$ (b) Training results obtained for $n = 4$

Figure 11: Results for $n = 3$ and $n = 4$

5.2 Fitting utilizing multiple values of n

With these great results we decided to try to fit and optimize the parameters by feeding some values with $n = 3$ and $n = 4$. However, since we are fitting for multiple values of n, we decided to increase the number of layers up to 9 layers, in order to increase the expressivity of the circuit. The results obtained, using the same number of iterations and the same number of elements on the training set, were the following:



(a) Results obtained when the input data had multiple values of n (3 and 4) with 7 layers (b) Results obtained when the input data had multiple values of n (3 and 4) with 9 layers

Figure 12: Impact of number of layers on training for multiple values of n

6 Discussion of the results

6.1 Lambert and Phong's diffuse term

As mentioned before, two different approaches were followed: one using a simple 1-qubit circuit, that would only train to fit a cosine, being the result later scaled by the diffuse constant (k_d), and another where the circuit would receive both an angle and a k_d value, and evaluate the diffuse term.

While the second approach might seem appealing, it has some hard to avoid problems. Firstly, if we train the circuit with only a fixed k_d value, even thought the circuit will be able to fit that specific value of k_d very well, it won't generalise for different values, and the result won't be what was expected, as we can see from the following picture:

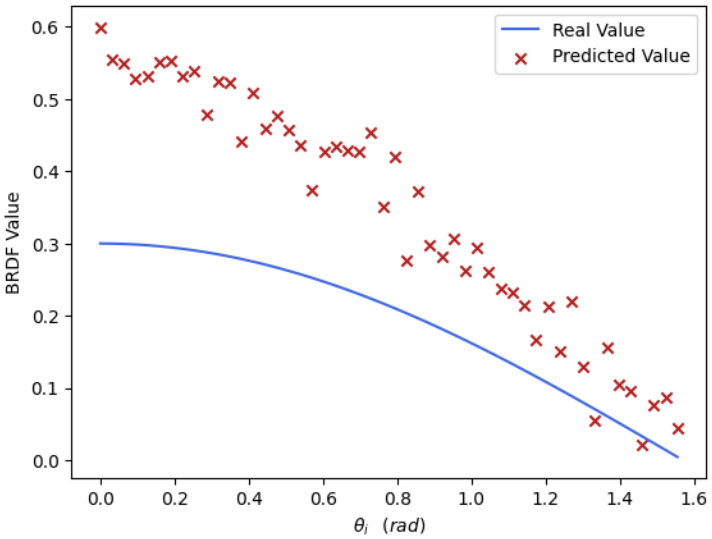


Figure 13: Predicted Values for $k_d = 0.3$ using the circuit trained with $k_d = 0.6$

This happens because the circuit, was only trained with a fixed value of k_d, and thus cannot generalize all the possible values. Considering now the circuit trained with multiple values of k_d, while it can better generalise, the predicted values are not that close to the true values. However, if we use the circuit that only trained to learn the cosine, and then multiply the results by the fixed constant k_d, the final result is always a really good approximation of the real value, independently from the value of k_d. In the next table we show a comparison of scores between the 3 circuits:

Values of k_d	Scores (MSE) 1-qubit circuit	Scores (MSE) 2-qubit with fixed k_d	Scores (MSE) 2-qubit with multiple k_d
0.1	8.64E-6	6.98E-2	1.51E-3
0.3	4.54E-5	2.60E-2	3.13E-3
0.6	2.62E-4	1.78E-3	1.41E-2
0.95	4.04E-4	3.57E-2	2.19E-2

Table 2: Values of the Mean Squared Error (MSE) for multiple values of k_d

As expected, the best scores were obtained with the 1-qubit circuit, and then scaling the output by the value of k_d, since the circuit was simpler, less noisy, and only dependent on a single variable (θ_i).

For the 2-qubit circuits, as we have seen in figure 9, we can obtain relatively good results by fixing the value of k_d, however when using different values it becomes evident that the circuit didn't generalise the impact of k_d and obtains it's best result for $k_d = 0.6$ which was the value it was trained on. On the other hand, training with a variable value of k_d allows the circuit to better generalise the role of k_d leading to overall better results, comparing to training with a fixed value.

6.2 Phong's specular term

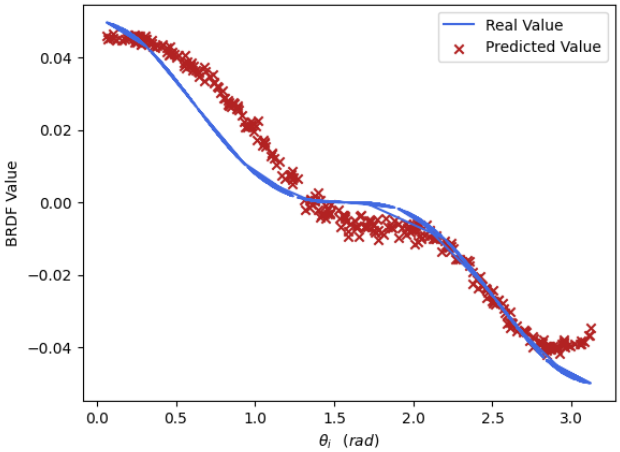
For the second term of the Phong's model, the specular term, the results obtained from training the model with a single value of n were very promising. Both the circuits, for $n = 3$ and $n = 4$, managed to do a good fit of the functions.

Since both models managed to get good results, we then built two circuits that would be trained with both values of $n = 3$ and $n = 4$, one with 7 layers and another with 9 layers. The results of those are shown in figures 12a and 12b. By only looking at the figures, we can see that the circuit with 9 layers performed slightly better than the one with 7 layers, what can be confirmed by the following MSE values:

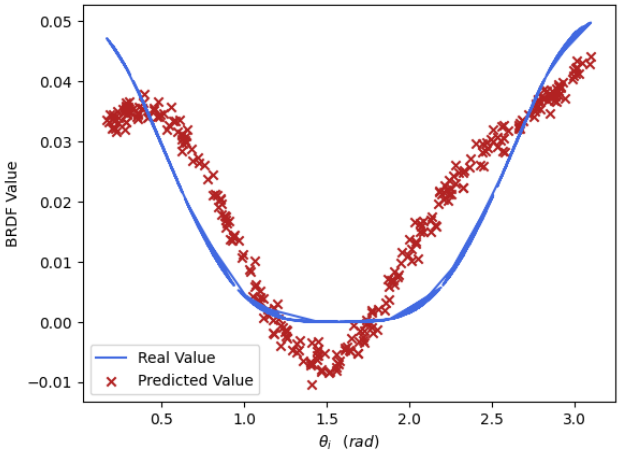
Values of n	Scores (MSE) trained with n=3	Scores (MSE) trained with n=4	Scores (MSE) both n's and 7 layers	Scores (MSE) both n's and 9 layers
3	1.54E-5	1.4E-2	6.00E-5	4.28E-5
4	7.30E-3	1.99E-5	4.94E-5	6.56 E-5

Table 3: Values of the Mean Squared Error (MSE) for multiple values of n

As we can see from these results, the models trained with a single value of n, make very good prediction for that specific value, and nothing else, while the one trained with multiple values can make decent predictions of data with those values of n. However, these prediction might not be as good compared to those made with the circuit trained with a specific n, as we can see from the following graphs:



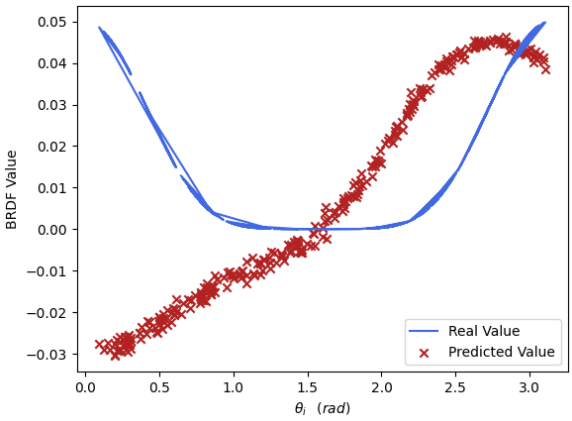
(a) Results obtained by using the circuit trained with both values of n, for $n = 3$



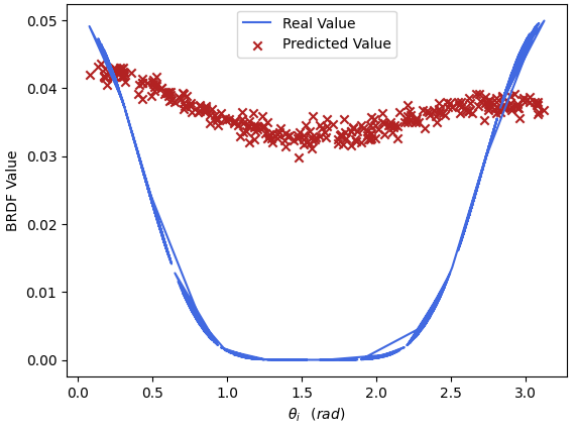
(b) Results obtained by using the circuit trained with both values of n, for $n = 4$

Figure 14: Results obtained using the circuit trained with multiple values of n

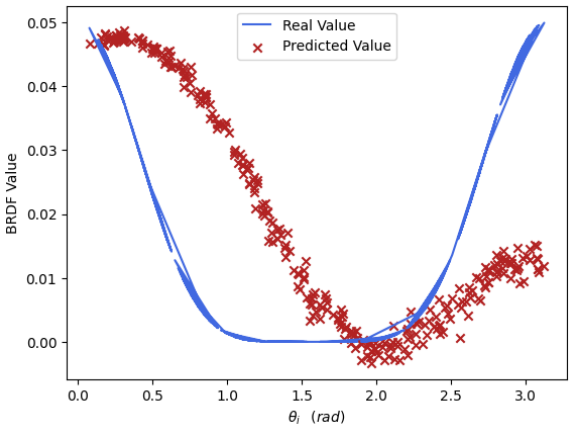
To further explore this dependence of the n values used for training, we tried to use the previous models to fit a data set where $n = 6$. The results are the following:



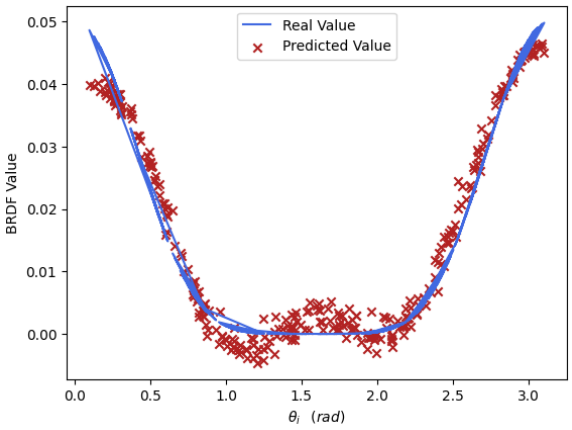
(a) Results obtained by using the circuit trained with $n = 3$, for $n = 6$



(b) Results obtained by using the circuit trained with $n = 4$, for $n = 6$



(c) Results obtained by using the circuit trained with both values of n, for $n = 6$



(d) Results obtained by using a circuit trained with $n = 6$

Figure 15: Results obtained using the circuits to predict values for $n = 6$

While circuit-based approaches show promise in fitting the Phong's model specular term, creating a generalized solution for various angles and n values remains challenging. The complex relationship between these variables necessitates extensive training data and multilayer circuits, which can lead to optimization difficulties like barren plateaus, preventing the parameters to be further optimized. This last point is also one of the reasons that we chose to precompute the value of ω_r as the 4 entries needed, as ω_r is dependent on 4 angles, massively increase the number of gates and depth of the circuit leading to much more noticeable barren plateaus and making it almost impossible to train.

However, if the values of n are trained separately, it's possible to create and optimize circuits to a given specific value of n, which can still be a great approach when dealing with this kind of models.

6.3 Joining the Phong's diffuse and specular terms

Having both the circuits for the specular and diffuse terms, we can simply sum up the results obtained from both models, to fully calculate the value of the BRDF using Phong's model.

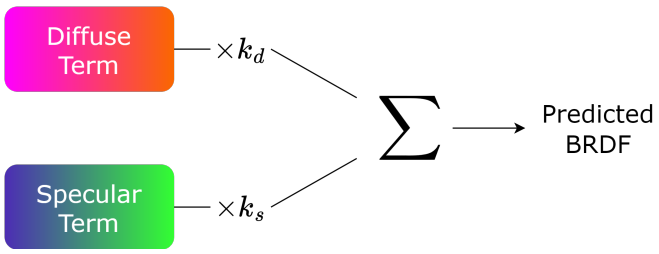
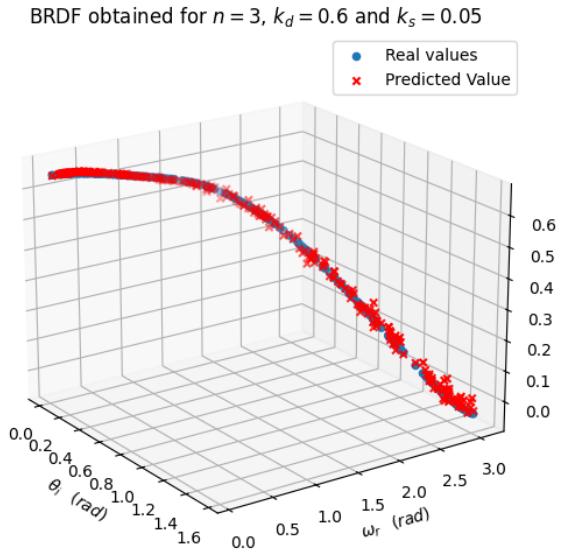
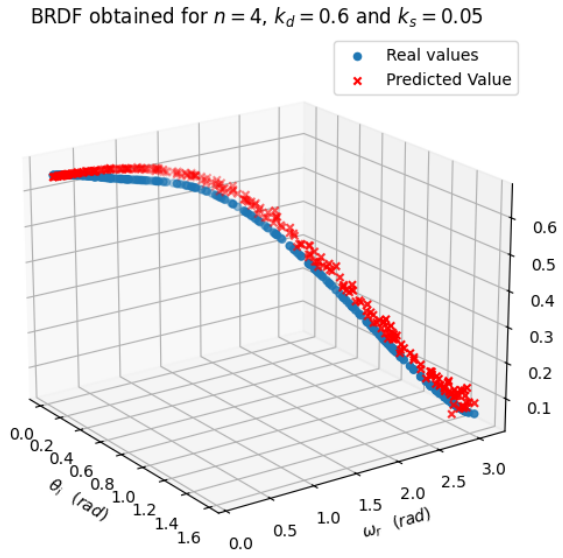


Figure 16: Diagram illustrating the full fit of a Phong's BRDF

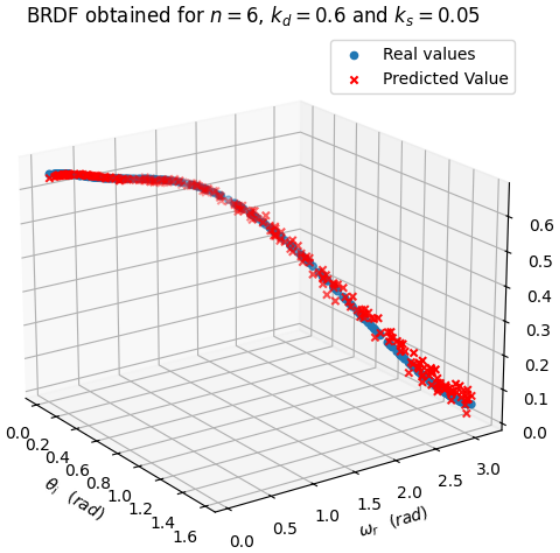
Performing these evaluations for multiple values of n, we obtained the following results:



(a) Predicted BRDF values for $n = 3$



(b) Predicted BRDF values for $n = 4$



(c) Predicted BRDF values for $n = 6$

Figure 17: Results of the full BRDF values using Phong's model

We can also check the values of the MSE and MAE for the derived values:

Values of k_d	Values of n	Average BRDF Value	Average predicted BRDF Value	Score (MAE)	Score (MSE)
$k_d = 0.6$	3	0.444	0.447	1.23E-2	1.52E-4
	4	0.460	0.487	3.13E-2	9.35E-5
	6	0.458	0.462	1.39E-2	1.93E-4
$k_d = 0.2$	3	0.144	0.144	5.46E-3	2.98E-5
	4	0.164	0.167	7.17E-3	5.14E-5
	6	0.160	0.161	6.27E-3	3.94E-5

Table 4: Scores obtained for the multiple circuits with diverse values of n and k_d, with $k_s = 0.05$

From this results we can see that, diving Phong's model into two different circuits and then merging the values obtained from both, resulted in accurate results for the Phong's BRDF model. However, this results only hold if the circuit responsible for predicting the values for the specular term is trained with values of the corresponding n values, since, if fed with different values, the circuit won't generalise and the results won't be accurate. On the other hand, the circuit responsible for the diffuse term, since it's the same one used for Lambert's model, can generalize for any value of θ_i and k_d, with it's only requirement being scaling the output by the value of k_d.

7 Conclusions and future work

In this work we've studied a model of QML based on re-uploading data into the circuit, in order to train and optimize a circuit capable of deriving BRDFs values from the angles θ_i, ω_r and the shininess constant n.

Our regression on datasets with values of BRDF and it's according angles and constants demonstrated the ability of QML to learn simple models of cosines and scaling of cosines, with the difficulty to generalize those learnt models to higher order multiplications and exponentiation of such values.

For the Lambert's model and for the diffuse term of the Phong's model, we've seen that, while it's possible to train a 2-qubit circuit to learn how to generalize and fit a scaled cosine, the simpler way to do such a thing is to optimize a 1-qubit circuit to fit a cosine, and then scale the output for the desired scaling, since it requires less qubits, and will lead to less noise and a higher chance to avoid barren plateaus when performing such calculations on a real quantum computer.

As for the specular term of the Phong's model, we've reached the conclusion that it's hard to generalize a circuit to learn how to give a proper result for any given shininess constant, being the best approach to such problem to train and model a circuit for a specific shininess constant. In that case, the circuit will be able to give accurate results for specular term value, given that the shininess constant will be fixed at the value used to train the circuit.

We've also seen that, while it's indeed possible to train a circuit with multiple values of n, it will lead to inaccurate and imprecise results, which, for the low values of BRDF, will have implications on real world applications.

With the two circuits previously described, and taking into consideration the conditions for accurate results, it's possible to obtain accurate values for the Phong's model BRDFs by scaling the outputs of both circuits by the corresponding factors (k_d for the diffuse term and k_s for the specular one) followed by the sum of the scaled values.

Looking towards the future, we envision to create and optimize a circuit that, given the values of angles involved in BRDFs calculations, outputs the value of the corresponding ω_r angle, like shown in the next figure.

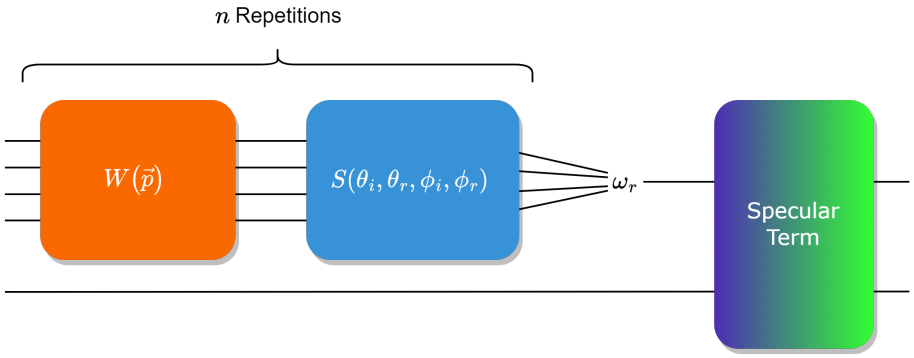


Figure 18: Scheme for a future circuit

Another objective is to try to optimize these models in a real quantum computer, which would provide extra challenges, like dealing with noise and, consequently, harsher barren plateaus.

Finally one must wonder beyond the analytical models and try to simulate real-world materials which can't be easily described through analytical functions but may find a fitting approximation through the QML techniques explored throughout this work.

By doing this we can further explore the power of QMC in function fitting, allowing it to tackle a wider range of real-world problems, further solidifying its position as a transformative technology.

References

[1] M. Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C. Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R. McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, and Patrick J. Coles. Variational quantum algorithms. *Nature Reviews Physics*, 3(9):625–644, August 2021.

[2] Simultaneous perturbation stochastic approximation (2024). Spsa algorithm. available at: https://www.jhuapl.edu/spsa/ (accessed: 15 june 2024).

[3] Maria Schuld, Ryan Sweke, and Johannes Jakob Meyer. Effect of data encoding on the expressive power of variational quantum-machine-learning models. *Physical Review A*, 103(3), March 2021.

[4] A V Uvarov and J D Biamonte. On barren plateaus and cost function locality in variational quantum algorithms. *Journal of Physics A: Mathematical and Theoretical*, 54(24):245301, May 2021.