Financial derivatives pricing using quantum neural networks: state-of-the-art

Aleksandar S. Milinković

***Abstract* — This paper presents the state-of-the-art in financial derivatives pricing using quantum artificial neural networks. Through the presentation of the available literature, it was shown that this type of application is only in its infancy and that there are still many open questions. As an illustration, the use of quantum artificial neural network to solve the option pricing problem, with given values of underlying asset and strike price, is shown. Furthermore, it is shown that Greeks, such as delta and gamma, which are important measures in risk management, can be computed analytically with this neural network.**

***Key words* — Derivatives pricing, quantum machine learning, quantum neural networks.**

I. Introduction

DERIVATIVE contract is a financial asset whose value is based on (or derived from) the price of one or more underlying assets. Examples of these underlying assets include stocks, currencies, commodities, etc. A derivative contract is typically issued between an issuer and a holder, and is valid until its expiration date. Each derivative defines a payoff that quantifies what the holder stands to gain. Generically, payoffs depend on the value of the underlying assets across the duration of the contract. Derivative contracts are ubiquitous in finance with various uses from hedging risk to speculation, and currently have an estimated global gross market value of approximately \$18.3 trillion dollars [1].

The goal of derivative pricing is to determine the value of entering a derivative contract today, given uncertainty about future values of the underlying assets and consequently the ultimate payoff. Financial derivatives

A. S. Milinković, Union Universty – School of Computing, Belgrade, Knez Mihailova 6, 11000 Belgrade, Srebia; (e-mail: amilinkovic2721m@raf.rs).

pricing methods are mathematical models used to calculate the fair value or price of financial derivatives. More about financial derivatives and their pricing can be found in [2] and [3].

Some commonly used pricing methods are Black-Scholes model, Binomial model, Monte Carlo simulation, Finite Difference methods, Variance Swaps and Volatility model, Credit Risk models, etc. It's important to note that each pricing method has its assumptions, limitations, and applicability to different types of derivatives. However, all of them consume significant computational resources for financial institutions. Therefore, finding a quantum advantage for this application would be very valuable to the financial sector as a whole.

II. QUANTUM COMPUTING FOR DERIVATIVES PRICING

The use of quantum computing for derivative pricing has a relatively long history [4]-[8]. Nowadays, due to the progress in developing of real (physical) quantum computers, the research in the field of industrial applications has intensified. In the field of derivative pricing, the article [9] is considered to be one of the first influential works. It presents a way to price plain vanilla and Asian options in a Black–Scholes framework using quantum computing. On the other hand, focus the research in [10] is on the possibilities and performance of the currently available noisy intermediate-scale quantum (NISQ, c.f. [11]) hardware to price multi-asset and path-dependent options. Likewise, in a Black–Scholes framework, article [12] presents an algorithm which precomputes standard normal distributions into quantum states and uses affine transformations to obtain the asset’s path-dependent return distributions. In addition to this new method, they also derive bounds on required resources for established quantum algorithms in order to reach a practical quantum advantage. Another approach, followed by [13], is to solve partial derivative equations for option prices by using the finite difference method. Pricing multi-dimensional derivatives with the help of discretizing their price partial differential equations is a method used as well in [14]. The novelty was that a variational quantum algorithm was used. Authors of [15] use a Heath–Jarrow–Morton framework for modeling forward rates and, in doing so, show how the model is adapted to the quantum computer. The application of a quantum gradient estimation algorithm to calculate financial indicators of sensitivity (Greeks) is illustrated by [16]. They present different approaches and investigate their complexity with respect to the requirements of quantum computers. Authors of [17] show an efficient way to prepare quantum states when employing amplitude estimation and present their findings on a Heston model for option pricing using real hardware.

Quantum computing utilizes the superposition property to achieve much higher computational efficiency than classical computers. For example, quantum supremacy was achieved in 2019. Article [18] reported that Google succeed in computing a problem in only 200 seconds using a quantum computer that would take 200 million years to solve with a classical supercomputer. In classical computers, the basic component of information is called a bit, which is expressed by 0 or 1 deterministically. On the other hand, quantum computers use two basis quantum states denoted as $|0\rangle$ and $|1\rangle$, analogously to a classical bit, to express their basis component of information, which is called a qubit and determined probabilistically. A single qubit state can be represented as a normalized two-dimensional complex vector, i.e.,

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \;\; \|\alpha\|^2 + \|\beta\|^2 = 1 \qquad (1)$$

where α and β are complex numbers and $\|\alpha\|^2$ and $\|\beta\|^2$ are the probabilities of observing $|0\rangle$ and $|1\rangle$ from the qubit, respectively.

We use the vector representation convention introduced by Dirac [19] for Hilbert spaces, assumed and used extensively in quantum mechanics. In this case, a *ket* vector, represented as $|a\rangle$, is a column vector of complex numbers, while a *bra* vector, represented as $\langle a|$, is the conjugate transpose of $|a\rangle$, that is $\langle a| = |a\rangle^\dagger$. The Hilbert space inner product is represented as $(|a\rangle, |b\rangle) = \langle a|b\rangle$. The outer product is, in turn, given by $|a\rangle\langle b|$. A projection operator corresponds to an operator of the form $P_a = |a\rangle\langle a|$ which acts on any ket $|b\rangle$ as $P_a\,|b\rangle = \langle a|b\rangle|a\rangle$.

Equation (1) can be also geometrically represented using polar coordinates θ and φ,

$$|\psi\rangle = \cos(\theta/2)|0\rangle + e^{i\varphi}\sin(\theta/2)|1\rangle \qquad (2)$$

where $0 \leq \theta \leq \pi$ and $0 \leq \varphi \leq \pi$. This representation maps a single qubit state into the surface of 3-dimensional unit sphere, which is called Bloch sphere. A multi qubit system can be represented as the tensor product of n single qubits, which exists as a superposition of 2^n basis states from $|00\ldots00\rangle$ to $|11\ldots11\rangle$. Quantum entanglement appears as a correlation between different qubits in this system. For example, in a 2-qubit system $2^{-1/2}|00\rangle + 2^{-1/2}|11\rangle$, the observation of the first qubit directly determines that of the second qubit. Those systems are controlled by quantum gates - unitary operators mapping a qubit system into another one. Every quantum gate can be factorized into the combination of several basic operators, like rotation operator gates and Controlled Not (CX) gate. Rotation operator gates $R_x(\theta)$, $R_y(\theta)$, $R_z(\theta)$ rotates a qubit state in Bloch sphere around corresponding axis by θ and CX gate entangles two qubits by flipping a qubit state if the other is $|1\rangle$. Those quantum gates utilizes quantum superposition and entanglement to take an advantage over classical computing, and it is well known that quantum algorithms can obtain an exponential computational gain over existing algorithms in certain tasks (e.g. prime factorization [20]).

III. QUANTUM ARTIFICIAL NEURAL NETWORKS

The connection between quantum computer science and artificial neural networks (ANNs) has been object of research since the 1990s, in particular, in what regards quantum associative memory, quantum parallel processing, extension of classical ANN schemes, as well as computational complexity and efficiency of quantum artificial neural networks (QANNs) over classical ANNs [21]-[25].

Mathematically, a classical ANN with a binary firing pattern can be defined as an artificial networked computing system comprised of a directed graph with the following additional structure [26], [27]:

- A binary alphabet $A_2 = \{0,1\}$ associated to each neuron describing the neural activity, with 0 corresponding to a non-firing neural state and 1 to a firing neural state, so that the firing patterns of a neural network with N neurons are expressed by the set of all binary strings of length N: $A_2^N = \{s_1 s_1 \ldots s_N : s_k \in A_2, k = 1,2,\ldots N\}$;
- A real-valued weight associated with each neural link, expressing the strength and type of neural connection;
- A transfer function which determines the state transition of the neuron and that depends upon: the state of its incident neurons, the weight associated with each incoming neural links and an activation threshold that can be specific for each neuron.

A quantum version of ANNs, on the other hand, can be defined as a directed graph with a networked quantum computing structure, such that [25]:

- To each neuron is associated a two-dimensional Hilbert space H_2 spanned by the computational basis $B_2 = \{|0\rangle, |1\rangle\}$, where $|0\rangle$ encodes a non-firing neural dynamics and $|1\rangle$ encodes a firing neural dynamics;
- To a neural network, comprised of N neurons, is associated the tensor product of N copies of H_2, so that the neural network's Hilbert space is

the space $\mathrm{H}_2^{\otimes N}$ spanned by the basis $\mathrm{B}_2^{\otimes N} = \left\{ |s\rangle : s \in \mathrm{A}_2^N \right\}$ which encodes all the alternative firing patterns of the neurons;

- The general neural configuration state of the neural network is characterized by a normalized ket vector $|\psi\rangle \in \mathrm{H}_2^{\otimes N}$ expanded in the neural firing patterns' basis $\mathrm{B}_2^{\otimes N}$:

$$|\psi\rangle = \sum_{s \in \mathrm{A}_2^N} \psi(s) \, |s\rangle \tag{3}$$

with the normalization condition:

$$\sum_{s \in \mathrm{A}_2^N} |\psi(s)|^2 = 1 \tag{4}$$

The neural network has an associated neural links state transition operator U such that, given an input neural state $|\psi^{in}\rangle$, the operator transforms the input state for the neural network in an output state $|\psi^{out}\rangle$, reflecting, in this operation, the neural links for the neural network, so that each neuron has an associated structure of unitary operators that is conditional on its input neurons:

$$|\psi^{out}\rangle = U \, |\psi^{in}\rangle \tag{5}$$

The output state of a QANN shows, in general, complex quantum correlations so that the quantum dynamics of a single neuron may depend in a complex way on the entire neural network's configuration [25]. Considering the neurons n_1, ..., n_N for a N-neuron neural network, the U operator can be expressed as a product of each neuron's neural links operator following the ordered sequence n_1, ..., n_N, where neuron n_1 is the first to be updated and n_N the last (that is, following the activation sequence):

$$U = U_N \dots U_2 U_1 \tag{6}$$

Note. For some QANNs it is possible to consider the action of the operators conjointly and to introduce, in one single neural links operator, a transformation of multiple neurons' states, taking advantage of parallel quantum computation [25].

Each neuron's neural links operator is a quantum generalization of an activation function, with the following structure for the k-th neuron:

$$U_k = \sum_{sum} |s\rangle\langle s| \otimes U_k(s_{in}) \otimes |s'\rangle\langle s'| \tag{7}$$

where *sum* reads as $s \in \mathrm{A}_2^{k\text{-}1}, s' \in \mathrm{A}_2^{N-k}$ and where s*in* is a substring, taken from the binary word *ss′*, that matches in *ss′* the activation pattern for the input neurons of n_k, under the neural network's architecture, in the same order and binary sequence as it appears in ss′. U_k(s*in*) is a neural links function that maps the input substring to a unitary operator on the two-dimensional Hilbert space H_2. This means that, for different configurations of the neural network, the neural links operator for the *k*-th neuron U_k assigns a corresponding unitary operator that depends upon the activation pattern of the input neurons. The neural links operators incorporate the local structure of neural connections so that there is a unitary state transition for the neuron (a quantum computation) conditional upon the firing pattern of its input neurons.

Suppose some quantum states $|\psi^{in}\rangle$ and $|\psi^{out}\rangle$ satisfy (5), where *U* is an unknown unitary operator which is to be learned by the QANN using *N* training pairs $(|\psi_n^{in}\rangle, |\psi_n^{out}\rangle)$ for *n* = 1, .., *N*.

The architecture of the QANN is analogous to that of a classical deep neural network. The QANN is composed of an input layer, an output layer, and *L* hidden layers. The main issue is how to achieve a feedforward propagation of information in the QANN because the no-cloning theorem [28] states that the quantum state of a qubit cannot be copied to another qubit.

The first property of such a QANN is that the density matrix [28] of output quantum state ρ^{out} may be expressed as the composition of a sequence of completely positive layer-to-layer transition maps ε^l:

$$\rho^{out} = \varepsilon^{out}(\varepsilon^L(\ldots(\varepsilon^2(\varepsilon^1(\rho^{in})))\ldots)), \tag{8}$$

where

$$\varepsilon^l(Q^{l-1}) = \mathrm{tr}_{l-1}\left(\prod_{j=m_l}^{l} U_j^l (Q^{l-1} \otimes |0\ldots0\rangle_l \langle 0\ldots0| \prod_{j=1}^{m_l} U_j^{l\dagger}\right), \tag{9}$$

Q^{l-1} is any operator on the (*l* – 1)th layer, m_l is the number of neurons in layer *l*, and U_j^l is the *j*-th unitary matrix acting on the layers (*l* – 1) and *l*. This characterization of the output of a QANN highlights a key structural characteristic: information propagates from input to output and hence naturally implements a quantum feedforward neural network. Furthermore,

(9) can be expressed using the Kraus (measurement) operator A_α [29] (we have omitted index *l* for the Kraus operators to make the notation clearer):

$$\varepsilon^l(Q^{l-1}) = \sum_\alpha \mathrm{A}_\alpha Q^{l-1} \mathrm{A}_\alpha^\dagger. \tag{10}$$

Similar to a classical deep network, we want to estimate *U* ("weights") from the training pairs using some cost function. Operationally, there is an essentially unique measure of closeness for (pure) quantum states, namely the fidelity, and it is for this reason that we define our cost function to be the fidelity between the QANN output and the desired output averaged over the training data:

$$C = \frac{1}{N}\sum_{n=1}^{N} \langle \psi_n^{out} \mid \rho_n^{out} \mid \psi_n^{out} \rangle, \tag{11}$$

which varies between 0 (worst) and 1 (best).

Parameter updating is proposed in [30] according to:

$$U \to e^{iK_d\zeta} U \tag{12}$$

where K_d represents the parameterized matrix chosen such that *C* increases the most rapidly, and where ζ is the learning step size. The change in *C* is given by:

$$\Delta C = \frac{\zeta}{N}\sum_{n=1}^{N}\sum_{i=1}^{L+1} \mathrm{tr}((\sigma_n^l)(\Delta\varepsilon^l)(\rho_n^{l-1})), \tag{13}$$

$\rho_n^l = \varepsilon^l(\ldots(\varepsilon^2(\varepsilon^1(\rho_n^{in}))\ldots)$, $\sigma_n^l = \mathcal{F}^{\ell+1}(\ldots\mathcal{F}^{out}(|\psi_n^{out}\rangle\langle\psi_n^{out}|)\ldots)$, and $\mathcal{F}$ is the adjoint channel of ε given by

$$\mathcal{F}^\ell(Q^{l-1}) = \sum_\alpha \mathrm{A}_\alpha Q^{l-1} \mathrm{A}_\alpha^\dagger. \tag{14}$$

It is derived in [30] that the elements of K_d are represented as follows:

$$K_{d,j}^l = \eta \frac{2^{m_{l-1}}}{N} \sum_{n=1}^{N} \mathrm{tr}_{rest} M_j^l, \tag{15}$$

where

$$M_j^l = [\prod_{\alpha=j}^{l} U_\alpha^l (\rho_n^{l-1} \otimes |0\ldots0\rangle_l \langle 0\ldots0| \prod_{\alpha=1}^{j} U_\alpha^{l\,\dagger}, \prod_{\alpha=j+1}^{m_l} U_\alpha^{l\,\dagger} (\mathrm{I}_{l-1} \otimes \sigma_n^l) \prod_{\alpha=m_1}^{j+1} U_\alpha^l],$$

η is the learning rate and *rest* in (15) means that M_j^l is traced out over all the qubits independent of unitary U_α^l.

To evaluate $K_{d,j}^l$ for a specific neuron, we only need the output state of the previous layer, ρ_n^{l-1} (which is obtained by applying the layer-to-layer channels ε^1, $\varepsilon^2 \ldots \varepsilon^{l-1}$ to the input state), and the state of the following layer σ^l obtained from applying the adjoint channels to the desired output state up to the current layer. An exceptional feature of this algorithm is that the parameter matrices may be calculated layer-by-layer without ever having to apply the unitary operator corresponding to the full quantum circuit on all the constituent qubits of the QANN in one go. In other words, we need only access two layers at any given time, which greatly reduces the memory requirements of the algorithm. Hence, the size of the matrices in our calculation only scales with the width of the network, enabling us to train very deep QANNs.

IV. QANN FOR LEARNING OPTION PRICES AND COMPUTING GREEKS

The following results are taken from [32] and slightly modified. Suppose V is the value of a financial derivative, x is an underlying asset that affects V, and K is a strike (the price at which the underlying asset can be either bought or sold once exercised). The estimation of not only V but also the differentials (Greeks), such as dV/dx and d^2V/dx^2, is an important task for financial risk management. Differential machine learning (DML) proposed in [31] has received great attention in the financial industry because DML estimates these differentials efficiently using a twin network scheme. After using a conventional feedforward neural network for predicting V, DML applies backpropagation via automatic adjoint differentiation (AAD) to compute the differential of the output V with respect to the input x. Weights are updated to minimize a cost function consisting of a weighted summation of the prediction errors of V and the differentials. Another aspect of DML is that it can approximate the shape of V with respect to x via the differentials, which greatly improves the estimation accuracy of V.

The following normalization function is used to convert market data to numbers between 0 and 1, and to fuse them into Bloch sphere representations:

$$P(a,b,c) = \frac{1}{1+\exp(-(a/b)c)} \cdot \frac{\pi}{2}. \tag{17}$$

Given *x*, *V*, and strike *K*, we consider the following input and output:

$$|\psi^{in}\rangle = \psi_1^{in}\,|0\rangle + \psi_2^{in}\,|1\rangle, \quad |\psi^{out}\rangle = \psi_1^{out}\,|0\rangle + \psi_2^{out}\,|1\rangle,$$

where

$$\psi_1^{in} = \cos(P(x,K,\beta)), \quad \psi_2^{in} = \sin(P(x,K,\beta)),$$

and

$$\psi_1^{out} = \cos(P(V,K,\gamma)), \quad \psi_2^{out} = \sin(P(V,K,\gamma)).$$

and where β and γ are fixed parameters.

Then density matrices ρ^{in} and ρ^{out} are given as

$$\rho^{in} = \begin{bmatrix} (\psi_1^{in})^2 & \psi_1^{in}\psi_2^{in} \\ \psi_1^{in}\psi_2^{in} & (\psi_2^{in})^2 \end{bmatrix}$$

$$\rho^{out} = \begin{bmatrix} (\psi_1^{out})^2 & \psi_1^{out}\psi_2^{out} \\ \psi_1^{out}\psi_2^{out} & (\psi_2^{out})^2 \end{bmatrix}$$

so, the QANN gives

$$\rho^{out} = \mathrm{tr}_{in,hid}(U(\rho^{in} \otimes |0\ldots0\rangle_{hid,out}\langle 0\ldots0|)U^{\dagger}). \tag{18}$$

The right-hand side of (18) means that ρ^{in} is tensorized with qubits $|0...0\rangle$ in hidden layers, and the output layer, unitary operators *U* are applied to the tensorized quantum state, and finally the resulting quantum state is traced out over qubits in input layer and hidden layers. Furthermore, given (18), it holds that

$$\frac{d\rho^{out}}{dx} = tr_{in,hid}(U((d\rho^{in}/dx) \otimes |0\ldots0\rangle_{hid,out}\langle 0\ldots0|)U^{\dagger}). \tag{19}$$

Equation (19) implies that given the QANN and $d\rho^{in}/dx$, we can compute $d\rho^{out}/dx$. Therefore, the QANN is highly effective for learning differentials. However, if the QANN is implemented on a quantum computer, the main issue is that $d\rho^{in}/dx$ and $d\rho^{out}/dx$ are not quantum states (the traces are zero [32]). The traces must be one, the eigenvalues must be positive real numbers and we therefore modify them as follows:

$$\frac{d\rho^{in,m}}{dx} = \frac{\mu_{in}}{2}\frac{d\rho^{in}}{dx} + \frac{I}{2} + \frac{1}{2}(r_{in}M_{\rho,x} + I),$$

where $r_{in} = \mu_{in}k_x$, and

$$\frac{d\rho^{out,m}}{dx} = \frac{\mu_{out}}{2}\frac{d\rho^{out}}{dx} + \frac{I}{2} + \frac{1}{2}(r_{out}M_{\rho,V} + I),$$

where $r_{out} = \mu_{out}\frac{dV}{dx}k_V$, I is the identity matrix and μ_{in}, μ_{out} are scaling parameters. Note that $d\rho^{in,m}/dx$ and $d\rho^{out,m}/dx$ are quantum states since the traces are equal to one and the eigenvalues are positive real as long as

$$0 \le r_{in} \le \frac{1}{2},\, 0 \le r_{out} \le \frac{1}{2}.$$

Then the following output from the QANN is also a quantum state:

$$\frac{d\rho^{out,in}}{dx} = \mathrm{tr}_{in,hid}(U((d\rho^{in,m}/dx)\otimes|0\ldots0\rangle_{hid,out}\langle0\ldots0|)U^{\dagger}).$$

We can define $d^2\rho^{in,m}/dx^2$ and $d^2\rho^{in,m}/dx^2$ similarly. Then, the cost function to be maximized is:

$$C_d = \frac{1}{N}\sum_{n=1}^{N}\langle\psi_n^{out}|\rho_n^{out}|\psi_n^{out}\rangle + \nu_d\frac{1}{N}\sum_{n=1}^{N} f_d(d\rho_n^{out,m}/dx, d\sigma_n^{out,m}/dx) +$$
$$+\nu_g\frac{1}{N}\sum_{n=1}^{N} f_d(d^2\rho_n^{out,m}/dx^2, d^2\sigma_n^{out,m}/dx^2)$$

where $f_d(\cdot,\cdot)$ represents the fidelity for mixed states, and ν_d and ν_g control the relative influence of the differential fidelity on the cost function. The unitary layers are updated according to (12). The supplementary information of [30] gives the elements of K_d as follows:

$$K^l_{d,j} = \eta \frac{2^{m_{l-1}}}{N}\left(\sum_{n=1}^{N} \mathrm{tr}_{rest} M^l_j + \nu_d \sum_{n=1}^{N} \mathrm{tr}_{rest} M^l_{d,j} + \nu_g \sum_{n=1}^{N} \mathrm{tr}_{rest} M^l_{g,j} \right),$$

where corresponding matrices are calculated using (16).

Finally, after denormalization, the predicted option price is given as:

$$V_p = K\left(-\log\left(\frac{\pi}{2\arccos(\sqrt{X}\,)} - 1 \right) \right)^{1/\gamma},$$

where X is the (1,1) element of matrix ρ^{out}. If we implement the QANN using a quantum computer, we can calculate $\cos(P(V, K, \gamma))$ by computing an approximation of the probability of observing $|0\rangle$.

Similarly, the predicted delta, $\Delta = dV/dx$, is given as:

$$\Delta = \frac{Y - Z}{\frac{\mu_{out}}{2} \cdot k_{V_p} \cdot \sin\left(\frac{1}{1 + \exp(-V_p / K)^{\gamma}} \cdot \pi \right)},$$

where Y is the (1,1) element of matrix $d\rho^{out,m}/dx$, Z is the (1,1) element of $\mathrm{tr}_{in,hid}(U(1/2 \cdot I \otimes |0\ldots0\rangle_{hid,out}\langle 0\ldots0|)U^{\dagger})$, V_p is the option price predicted by the QANN, and

$$k_{V_p} = \frac{\pi}{2} \cdot \frac{\gamma}{K}(V_p / K)^{\gamma - 1} \cdot \frac{1}{1 + \exp(-V_p / K)^{\gamma}} \left[1 - \frac{1}{1 + \exp(-V_p / K)^{\gamma}} \right].$$

We can compute the predicted gamma $\Gamma = d^2V/dx^2$ similarly.

Note. Higher order differentials depend on the value of lower order differentials, so the accuracy of lower order differentials is especially important. On the other hand, an increase of ν_d or ν_f does not affect the accuracy of delta or gamma, respectively, if the difference between the training price and predicted price is not small [32].

The term $d\rho^{out,m}/dx$ can be expressed as:

$$\begin{aligned}\frac{d\rho^{out,m}}{dx} &= \frac{1}{2}(r_{out}M_{\rho,V} + I)\\ &= \frac{1}{2}I + r_{out}\cos(2P(V,K,\gamma)X - r_{out}\sin(2P(V,K,\gamma)Z,\end{aligned}$$

where X and Z are Pauli matrices

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

This implies that the Bloch vector of $d\rho^{out,m}/dx$ is:

$$\begin{bmatrix} r_{out}\cos(2P(V,K,\gamma) & 0 & -r_{out}\sin(2P(V,K,\gamma) \end{bmatrix}$$

and if we implement the QANN using a quantum computer, we can compute $-\ r_{out}$ sin((2P(V, K, γ)) and the predicted delta by measuring the expected value of the qubit in the output layer in the Z-basis.

V. Conclusion

Classical methods for financial derivatives pricing, such as Monte Carlo and partial differential equations, have certain limitations in terms of computational complexity and performance. Quantum methods and quantum machine learning using quantum artificial neural networks, offer potential speedups and improvements in performance. However, the term quantum supremacy may lead to the illusion that quantum algorithms are always better than classical algorithms performing the same function. Given the inherent limitations of quantum computing, quantum computing benefits can only be realized through well-thought-out algorithms under certain circumstances and assumptions. Therefore, in designing a quantum-based deep learning algorithm, it is necessary to justify it by articulating its advantages over the corresponding classical models.

The example shown is a truly quantum analogue of classical neurons, which form quantum feedforward neural network capable of universal quantum computation. Training of this network for option pricing utilizes the fidelity as a cost function. This method allows for fast optimization with reduced memory requirements. The number of qubits required scales with only the width, allowing to largely avoid the vanishing gradient problem in this particular application. However, it is unavoidable to deal with this problem when designing large-scale QANN. This is still an open problem for which a solution is not yet clear.

REFERENCES

[1] Bank for International Settlements, *OTC derivatives statistics at end-June 2022*. 30 November 2022. [Online]. Available: www.bis.org/publ/otc_hy2211.pdf

[2] J. Baz and G. Chacko, *Financial derivatives: Pricing, applications, and mathematics*, Cambridge, UK: Cambridge University Press, 2008.

[3] D. Sevcovic, B. Stehlíková, and K. Mikula, *Analytical and numerical methods for pricing financial derivatives*, Hauppauge, NY, USA: Nova Science Publishers, 2011.

[4] W. Segal and I. Segal, “The Black– Scholes pricing formula in the quantum context,” *Proc. Natl. Acad. Sci. U.S.A,* vol. 95 no. 7, pp. 4072–4075, Mar. 1998.

[5] B. Baaquie, L. Kwek, and M. Srikant, “Simulation of stochastic volatility using path integration: Smiles and frowns,” 2000. [Online]. Available: arXiv:cond-mat/0008327.

[6] B. Baaquie and S. Marakani, S. (2001), “Empirical investigation of a quantum field theory of forward rates,” 2001. [Online]. Available: arXiv:cond-mat/0106317.

[7] B. Baaquie, *Quantum finance: Path integrals and Hamiltonians for options and interest rates*, Cambridge, UK: Cambridge University Press, 2004.

[8] B. Baaquie and T. Pan, “Simulation of coupon bond European and barrier options in quantum finance,” *Physica A*, vol. 390, no. 2, pp. 263–289, Jan. 2011.

[9] P. Rebentrost, B. Gupt, and T. Bromley, "Quantum computational finance: Monte Carlo pricing of financial derivatives," *Phys. Rev. A*, vol. 98, Aug. 2018. Art. no. 022321.

[10] N. Stamatopoulos *et al.*, "Option pricing using quantum computers," 2020. [Online]. Available: arXiv:1905.02666v5.

[11] J. Preskill, "Quantum Computing in the NISQ era and beyond," Jul. 2018. [Online]. Available: arXiv:1801.00862v3.

[12] S. Chakrabarti, R. Krishnakumar, G. Mazzola, N. Stamatopoulos, S. Woerner, and W. Zeng, "A threshold for quantum advantage in derivative pricing," May 2021. [Online]. Available: arXiv:2012.03819v3.

[13] K. Miyamoto and K. Kubo, "Pricing multi-asset derivatives by finite difference method on a quantum computer," Sep. 2021. [Online]. Available: arXiv:2109.12896.

[14] K. Kubo, K. Miyamoto, K. Mitarai, and K. Fujii, "Pricing multi-asset derivatives by variational quantum algorithms," Jul. 2022. [Online]. Available: arXiv:2207.01277.

[15] A. Martin *et al.*, "Toward pricing financial derivatives with an IBM quantum computer," *Phys. Rev. Res.*, vol. 3, Feb. 2021. Art. no. 013167.

[16] N. Stamatopoulos, G. Mazzola, S. Woerner, and W. Zeng, "Towards quantum advantage in financial market risk using quantum gradient algorithms," Jul. 2022. [Online]. Available: arXiv:2111.12509v2.

[17] A. C. Vazquez and S. Woerner, "Efficient state preparation for quantum amplitude estimation," *Phys. Rev. Appl.*, vol. 15, Mar. 2021. Art. no. 034027.

[18] F. Arute *et al.* "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, pp. 505-510, 2019.

[19] P. Dirac, *The principles of quantum mechanics*, Oxford, UK: Claredon Press, 1967.

[20] P. Shor, “Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer,” *SIAM Rev. Soc. Ind. Appl. Math*., vol. 41, no. 2, pp. 303–332, 1999.

[21] S. Kak, “Quantum neural computing”, *Adv. Imaging Electron Phys.*, vol. 94, pp. 259–313. 1995.

[22] T. Menneer and A. Narayanan, “Quantum-inspired neural networks", Dept. Comp. Sci., Univ. Exeter, Exeter, UK, Tech. Rep. R329, 1995.

[23] T. Menneer, "Quantum artificial neural networks," Ph.D. dissertation, Univ. Exeter, UK, 1998.

[24] V. Ivancevic and T. Ivancevic, *Quantum neural computation*, Dordrecht, NL: Springer, 2010.

[25] C. Gonçalves, “Quantum cybernetics and complex quantum systems science: A quantum connectionist exploration”, *Neuroquantology*, vol. 13, no. 1, pp. 35–48, 2015.

[26] W. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” *Bull. Math. Biophys*., vol. 7, pp. 115–133, Dec. 1943.

[27] B. Müller, J. Reinhardt, and M. Strickland, *Neural networks: An introduction*, Berlin, DE: Springer-Verlag, 1995.

[28] M. Nielsen and I. Chuang, *Quantum computation and quantum information*, Cambridge, UK: Cambridge University Press, 2011.

[29] K. Kraus. *States, effects, and operations: Fundamental notions of quantum theory* (Lecture Notes in Physics, vol. 190), Berlin, DE: Springer-Verlag, 1983.

[30] K. Beer *et al.*, "Training deep quantum neural networks," *Nat. Commun.*, vol. 11, Feb. 2020. Art. no. 808.

[31] B. Huge and A. Savine, "Differential machine learning," Sep. 2020. [Online]. Available: arXiv:2005.02347.

[32] T. Sakuma, "Application of deep quantum neural networks to finance,", May 2022. [Online]. Available: arXiv:2011.07319.