IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 2 August 2024, accepted 15 August 2024, date of publication 19 August 2024, date of current version 30 August 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3446176

RESEARCH ARTICLE

Quantum Subroutine for Efficient Matrix Multiplication

ANNA BERNASCONI[1], ALESSANDRO BERTI[1,2], GIANNA MARIA DEL CORSO[1], AND ALESSANDRO POGGIALI[1], (Graduate Student Member, IEEE)

[1]Department of Computer Science, University of Pisa, 56127 Pisa, Italy
[2]Department of Physics, University of Pisa, 56127 Pisa, Italy

Corresponding author: Alessandro Berti (alessandro.berti@df.unipi.it)

This work was supported in part by European Union Next-GenerationEU—National Recovery and Resilience Plan (NRRP)—MISSION 4 COMPONENT 2, INVESTMENT N. 1.4—CUP N under Grant I53C22000690001; and in part by the Gruppo Nazionale Calcolo Scientifico–Istituto Nazionale di Alta Matematica (GNCS–INdAM).

ABSTRACT We propose an efficient quantum subroutine for matrix multiplication that computes a state vector encoding the entries of the product of two matrices in superposition. The subroutine exploits efficient state preparation techniques and shows a potential speed-up with respect to classical methods. The most important benefit of our subroutine is that it encodes the entries of the matrix product directly in the state vector, which can be used for further computations within the same quantum circuit. All scenarios involving the computation of non-homomorphic functions of the product of two matrices can benefit from our technique. As a possible application, we discuss the computation of the variance of the entries of a matrix product, which can be a useful tool for some machine learning algorithms.

INDEX TERMS Quantum circuit, quantum matrix multiplication, computation of non-homomorphic functions, state preparation.

I. INTRODUCTION

Matrix multiplication is a fundamental operation in linear algebra with a wide range of applications across various scientific and engineering disciplines. It lies at the heart of many scientific and high-performance computing problems, from solving systems of linear equations with direct and iterative methods to data compression, machine learning, and computer vision problems. Thus, efficient computation of matrix multiplication is still a topic of intense research and optimization. Furthermore, with the resurgence of interest in matrix multiplication fueled by the rapid growth of data-centric applications and the increasing need for improved computing capabilities, recent years have seen the emergence of new algorithms and methodologies that harness modern hardware architectures such as multicore processors, GPUs, distributed computing environments, and even quantum computers.

The associate editor coordinating the review of this manuscript and approving it for publication was Abdullah Iliyasu.

In this work, we delve into the problem of matrix multiplication from the quantum computing perspective, proposing an efficient quantum algorithm for matrix multiplication that computes a state vector encoding the entries of the product in superposition.

The algorithm achieves, in principle, an exponential speed-up over the best classical fast method thanks to the use of efficient state preparation techniques, similar to the block-encoding procedure for matrix multiplication [12].

Actually, it is worth noting that the proposed quantum algorithm achieves something different from the classical matrix multiplication one, as it does not return all entries of the matrix product; rather, it provides them in superposition as amplitudes of a state. If one needs to retrieve all entries with accuracy ε, the algorithm should be executed and measured $O(N^2/\varepsilon^2)$ times, losing the exponential speed up. However, similarly to what happens with the Quantum Fourier transform, our approach could still be very effective: since the entries of the matrix product are directly encoded in the state vector, although in superposition, they can be

exploited for further computations within the same quantum circuit, without performing any measurement. This makes our approach suitable for all scenarios involving the computation of non-homomorphic functions depending on the product of two matrices, such as trace, mean value, variance, or eigenvalues, which are crucial for many applications.

Other quantum algorithms for matrix multiplication proposed in the literature and reviewed in Section II, are either computationally more expensive or present a comparable cost but are less suitable for further performing non-linear operations depending on all entries of the matrix product. We refer to Section IV for a discussion on this aspect. Moreover, to highlight the novelty of our approach, we also provide an application for the computation of the variance of the entries of the product of two matrices. This computation finds many important applications within the machine learning area.

As already observed, the cost of the proposed quantum algorithm mainly depends on the cost of the encoding of the two matrices to be multiplied within a quantum state, and a super-polynomial speed-up can be guaranteed only by exploiting efficient state preparation techniques, i.e., techniques that can be implemented with quantum circuits of depth polylogarithmic in the input size [23].

The work is organized as follows. In Section II, we briefly discuss the recent progress in algorithms for linear algebra problems, particularly matrix multiplication, both from a classical and quantum perspective. In Section III, we first recall a quantum procedure described in [24] for inner product computation that inspired our contribution, and then we present and analyze our new algorithm for matrix multiplication. Section IV discusses the relation, advantages, and disadvantages of our proposal with respect to the method based on block-encoding. Finally, Section V describes how to exploit our algorithm for computing the variance of the entries of a matrix product, and Section VI draws the conclusions.

II. PRELIMINARY AND RELATED WORK

The historical significance of matrix multiplication dates back to the pioneering work of Strassen [29], who introduced the concept of recursive matrix multiplication, reducing the number of multiplicative operations from eight to seven for two matrices of size 2. Since then, numerous algorithms and optimization techniques have been proposed to accelerate matrix multiplication. Prominent among them are the method by Bini et al. [3] based on the concept of border rank of tensors, the Coppersmith-Winograd algorithm [10] and subsequent methods based on the laser method [34], and the more recent developments [19], [35] which seems to have a more practical interest.

The inception of quantum algorithms for linear algebra can be attributed to the pioneering work of Harrow et al. [15]. The HHL algorithm, a significant breakthrough, is designed to handle sparse and well-conditioned systems of linear equations as input. Remarkably, it accomplishes this in a polylogarithmic time complexity in the system's dimension. Although the HHL algorithm does not directly output the classical solution, the quantum state it produces empowers sampling from the solution vector. This capability has had a profound impact, inspiring subsequent works [26], [33] in the field of quantum algorithms for machine learning problems.

However, it is essential to exercise caution when considering the applications of these algorithms. Two critical factors warrant careful consideration: firstly, the assumptions concerning the input data that are necessary to achieve efficient running times. For instance, the HHL algorithm's polylogarithmic time complexity is contingent on the matrix being well-conditioned (i.e., the minimum singular value is sufficiently large) and sparse. Secondly, one must assess whether the quantum algorithm resolves the original classical problem or a modified variant, accounting for the fact that the classical solution is not explicitly provided but rather encoded within a quantum state.

Another interesting consideration is that while in the classical framework, problems such as matrix inversion, computation of the determinant, and linear system solution via Gaussian elimination are asymptotically equivalent to matrix multiplication (see Chapter 16 of [6]), to the best of our knowledge there is not a similar result in the quantum computing framework. The equivalence among these problems in the classical setting has been proved through polynomial reductions of cost quadratic in the dimension of the matrix (i.e., linear in the input size). However, these reductions cannot be immediately adapted to quantum computation as HHL does not use Gaussian elimination to solve a linear system, which was a key step for the classical reduction definition.

Regarding the specific problem of matrix multiplication, let us briefly recall some quantum algorithms proposed in the recent literature. In [28], three different algorithms have been presented for dealing with matrix multiplication. The first technique is inspired by the swap test [5], extended to a more general form suitable for dealing with quantum data in parallel, and has a cost that is dominated by the time for preparing the input at a given precision ε. The other two techniques are based on SVE [20] (singular value estimation) and HHL [15], respectively. The method based on the swap test proved better than the other two, achieving the lowest complexity $\tilde{O}(N^2/\varepsilon)$ to multiply two $N \times N$ matrices with classical data, with precision ε. The quantum algorithms obtained from SVE and HHL depend on the condition number of the given matrices. If the condition number is bounded by $O(\text{polylog}N)$, the two algorithms also achieve the same efficiency. However, these algorithms use amplitudes to store the classical input data and require a measurement to get the results. Hence, they are not immediately suitable to implement matrix multiplication as an intermediate step within a quantum computation.

In [22], a different approach has been proposed, where basis states are used to store input data. Thus, quantum algorithms that require matrix multiplication as an intermediate step do not have to rely on measurement to get the result and only need one measurement when the overall result of the computation is output. The proposed matrix multiplier exploits superposition and parallelism of quantum computing to reduce the time complexity to multiply two $N \times N$ matrices from classical $O(N^3)$ to quantum $O(N^2 \log^2 N)$. The space complexity also improves from classical $O(N^2)$ to quantum $O(\log N)$.

We finally recall that the *block encoding* technique [8], [12] can be exploited to implement matrix calculations on a quantum computer quite easily and, in principle, enable exponential speed-ups in terms of the dimension of the matrices. Since, in general, an input matrix of data, say A, could be non-unitary and cannot be directly implemented as a quantum operator, the idea of block encoding is to embed A as a block inside a larger unitary matrix $\mathcal{U}$, usually the top left block. Once a matrix has been block-encoded, we can operate on its corresponding unitary operator $\mathcal{U}$ so that all quantum matrix calculations are carried out in an operational way. For instance, as shown in [12], the product of two block-encoded matrices is a block encoding of the product of the two matrices, and the errors add up without introducing additional errors. The cost of implementing the block encoding of dense unstructured matrices has been studied in [9], where a careful analysis of T-gate counts and circuits' depth has been provided. Some efficient schemes to block-encode structured matrices such as Toeplitz, tridiagonal, or matrices with displacement structure, as well as sparse matrices, are known [7], [30], [32].

III. QUANTUM ALGORITHM FOR MATRIX MULTIPLICATION

In this section, we present and discuss a new quantum algorithm for matrix multiplication. We first recall the quantum procedure described in [24] for the computation of the inner product of two vectors that inspired our contribution. We then describe and analyze the proposed new algorithm, prove its correctness, and evaluate its computational cost.

A. INNER PRODUCT COMPUTATION

Let us briefly review the quantum algorithm proposed in [24] for computing the inner product of two vectors whose values are encoded in the amplitudes of two quantum states. Recall that the inner product of the two quantum states $|a\rangle = \frac{1}{\|a\|_2}\sum_{i=0}^{N-1} a_i |i\rangle$ and $|b\rangle = \frac{1}{\|b\|_2}\sum_{i=0}^{N-1} b_i |i\rangle$, representing two N-dimensional vectors, is defined as

$$\langle a|b\rangle = \frac{1}{\|a\|_2\|b\|_2}\sum_{i=0}^{N-1} \bar{a}_i b_i$$

where $\bar{a}_i$ is the complex conjugate of a_i, for all $i \in [0, N-1]$. Suppose that the two quantum states can be prepared by applying two oracles U_a and U_b, i.e., $|a\rangle = U_a|0\rangle$ and $|b\rangle = U_b|0\rangle$, where the input register $|0\rangle$ is a $n = \lceil \log N \rceil$-qubit register sufficiently large to store all indexes $i = 0, 1, \ldots, N-1$ in superposition. Since $\langle a| = \langle 0| U_a^\dagger$, and $|b\rangle = U_b|0\rangle$, we immediately get

$$\langle a|b\rangle = \langle 0|U_a^\dagger U_b|0\rangle .$$

Now, if we consider the quantum state

$$|\gamma\rangle = U_a^\dagger U_b|0\rangle = \sum_{i=0}^{N-1} \gamma_i |i\rangle$$

derived with the quantum circuit depicted in Figure 1, we can observe that the amplitude γ_0 of the state $|0\rangle$ is precisely the inner product of the two states $|a\rangle$ and $|b\rangle$, i.e.,

$$\gamma_0 = \langle 0|U_a^\dagger U_b|0\rangle = \langle a|b\rangle .$$

Thus, to compute the inner product, we just need to read the value γ_0, for instance, applying the Amplitude Estimation algorithm [4], [14], and multiply back by the normalization factor $\|a\|_2\|b\|_2$.

The overall cost of the method is directly connected to the state preparation cost for $|a\rangle$ and $|b\rangle$, i.e., to the cost of implementing the oracles U_a and U_b. Assuming efficient methods for encoding classical data [23]), the cost of the algorithm becomes $O(\text{polylog}\, N)$.

This approach might be more convenient than the standard one based on the swap test [5], even if the circuit costs (width, size, and depth) of the two methods are comparable. The cost of both algorithms is indeed dominated by the cost of the state preparation step. Recall that the swap test method uses an ancilla qubit in state $|0\rangle$, which is *(i)* put into a uniform superposition with a Hadamard gate, *(ii)* used to control a swap between the two states $|a\rangle$ and $|b\rangle$, and *(iii)* measured after the application of a second Hadamard gate. Since the result of the measurement of the ancilla is 1 with probability $\frac{1}{2} - \frac{1}{2}|\langle a|b\rangle|^2$, it is possible to estimate $\langle a|b\rangle$ through this probability. The downside of the swap test is that the value of the inner product can be estimated only after a measurement. Instead, in the previous method, the inner product is directly encoded into the amplitude γ_0 and can, therefore, be exploited for further computations within the same quantum circuit without the need for a measurement.

B. MATRIX MULTIPLICATION

Let us now consider the more general problem of matrix multiplication.

Let $A \in \mathbb{R}^{M\times K}$ and $B \in \mathbb{R}^{K\times N}$, assuming w.l.o.g. that M, K, N are powers of 2, and let $C \in \mathbb{R}^{M\times N}$ denote their product $C = AB$. Observe that each entry in C is the result of an inner product computation, i.e., C_{ij} is given by the inner product between the i-th row of A and the j-th column of B.

Our proposal consists of a generalization of the inner product procedure described above: we encode all columns of B, appropriately normalized, in a weighted superposition, and we then apply an operator encoding all rows of A. The entries of the matrix $C = AB$ will then be found in some specific amplitudes of the final state. Some care is needed

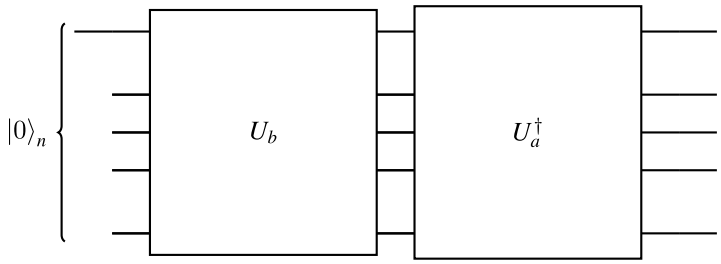

FIGURE 1. Quantum circuit for inner product computation.

to deal with the normalization factors properly. In particular, we need to amplitude-encode the norms of the rows of A and of the column of B (i.e., the rows of $B^\dagger$).

The quantum circuit for matrix multiplication (QMM) that we propose makes use of four unitary operators for encoding entries and norms of the matrices. In particular, U_A and V_A denote the oracles used to encode the rows of A and their norms in the amplitudes of $k = \log_2 K$ and $m = \log_2 M$ qubits, respectively:

$$\begin{aligned} U_A\,|i\rangle_m\,|0\rangle_k &= \frac{1}{\|A(i,:)\|_2}\,|i\rangle_m \sum_{j=0}^{K-1} A_{ij}\,|j\rangle_k \\ &= |i\rangle_m\,|A(i,:)\rangle_k\ ,\ \text{for } 0 \le i < M, \\ V_A\,|0\rangle_m\,|j\rangle_k &= \frac{1}{\|A\|_F} \sum_{i=0}^{M-1} \|A(i,:)\|_2\,|i\rangle_m\,|j\rangle_k \\ &= |\tilde{A}\rangle_m\,|j\rangle_k\ \text{for } 0 \le j < K. \end{aligned} \tag{1}$$

Similarly, we use the two oracles U_{B^T} and V_{B^T} to encode the columns of the matrix B, i.e., the rows of the matrix B^T, and their norms in the amplitudes of $k = \log_2 K$ and $n = \log_2 N$ qubits, respectively:

$$\begin{aligned} U_{B^T}\,|j\rangle_n\,|0\rangle_k &= \frac{1}{\|B(:,j)\|_2}\,|j\rangle_n \sum_{s=0}^{K-1} B_{sj}\,|s\rangle_k \\ &= |j\rangle_n\,|B(:,j)\rangle_k\ ,\ \text{for } 0 \le j < N,, \\ V_{B^T}\,|0\rangle_n\,|s\rangle_k &= \frac{1}{\|B\|_F} \sum_{j=0}^{N-1} \|B(:,j)\|_2\,|j\rangle_n\,|s\rangle_k \\ &= |\tilde{B^T}\rangle_n\,|s\rangle_k\ ,\ \text{for } 0 \le s < K. \end{aligned} \tag{2}$$

Note that the above relations specify the behavior of the two operators V_A and V_{B^T}, whose role is to recover the normalization factors of the columns and rows of the two matrices. However, these operators are defined only when the first register is in the $|0\rangle$ state. No specifications are provided for other configurations, potentially allowing more flexibility in the implementation of the operators within a quantum circuit. Moreover, we can observe that we are replicating the same information $|\tilde{A}\rangle$ and $|\tilde{B^T}\rangle$ on the first K columns of V_A and V_{B^T} in order to facilitate the removal of the normalization factors in U_A and U_{B^T}.

The circuit also requires a register-swap operator, denoted by RS, that is used to swap the position of two registers during the computation. The swap of two registers can be performed using several single swap gates,[1] each acting on a pair of qubits. In particular, if the two registers have the same number t of qubits, they can be swapped in constant depth applying exactly t swap gates, acting in parallel on t distinct pairs of qubits: the i-th qubits of the two registers, $1 \le i \le t$. Otherwise, if the registers contain a different number of qubits, their swap can be realized, for example, implementing the rotation by inversion algorithm (see [11] for details). This algorithm requires two steps, each of constant depth, consisting of the parallel application of swap gates on disjoint pairs of qubits. First, each register is reversed simultaneously, swapping its first qubit with its last one, its second qubit with the second last one, and so on until the middle is reached and the entire register is reversed. Then, the concatenation of the two reversed registers is reversed as well, with the same technique, thus realizing the interchange of the two original registers (see Figure 2). Note that each qubit is swapped twice.

The main steps of the quantum algorithm QMM for performing matrix multiplication according to our proposal are depicted in Figure 3, and detailed below.

Step 1: The initial state $|\varphi_0\rangle$ consists of three registers initialized to 0, of n, m, and k qubits respectively:

$$|\varphi_0\rangle = |0\rangle_n\,|0\rangle_m\,|0\rangle_k\,.$$

Step 2: We encode the norms of the rows of A in the second register with m qubits applying V_A to $|0\rangle_m\,|0\rangle_k$. Then, we swap the first two registers using the register-swap operator RS, and we finally apply V_{B^T} to encode the norms of the columns of B in the first register with n qubits. The state becomes

$$|\varphi_1\rangle = \frac{1}{\|A\|_F\,\|B\|_F} \sum_{i=0}^{M-1} \|A(i,:)\|_2\,|i\rangle_m \sum_{j=0}^{N-1} \|B(:,j)\|_2\,|j\rangle_n\,|0\rangle_k\,.$$

Step 3: We then create a weighted superposition of the columns of B using the unitary U_{B^T} on the second and third

[1] A swap gate is a 2-qubit gate used to interchange the state of two qubits. It can be implemented with three C-NOTs [25].

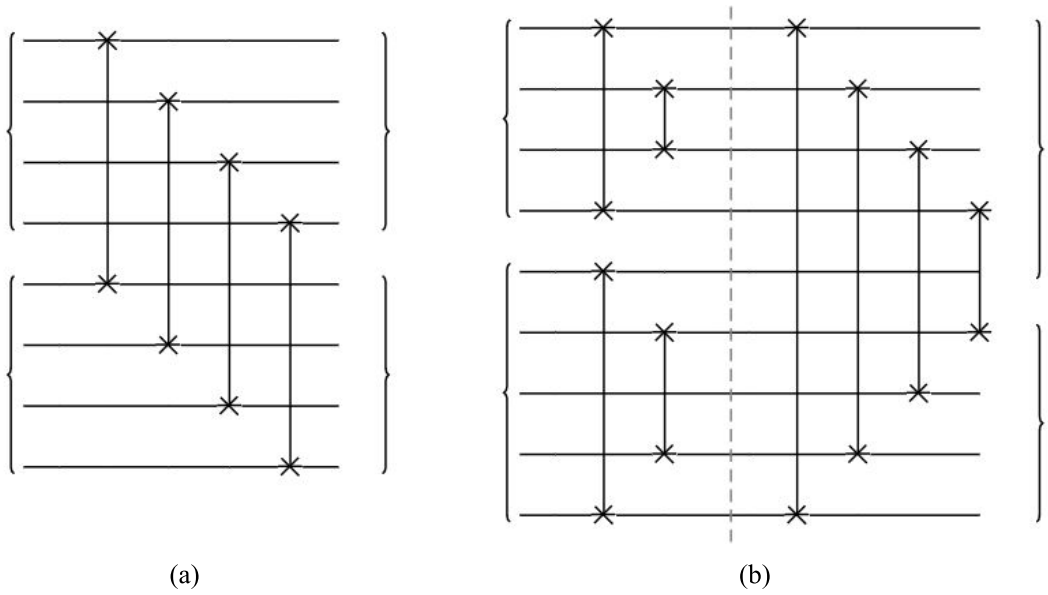

FIGURE 2. Operator *RS* for the swap of registers of equal size (a) and of different size (b).

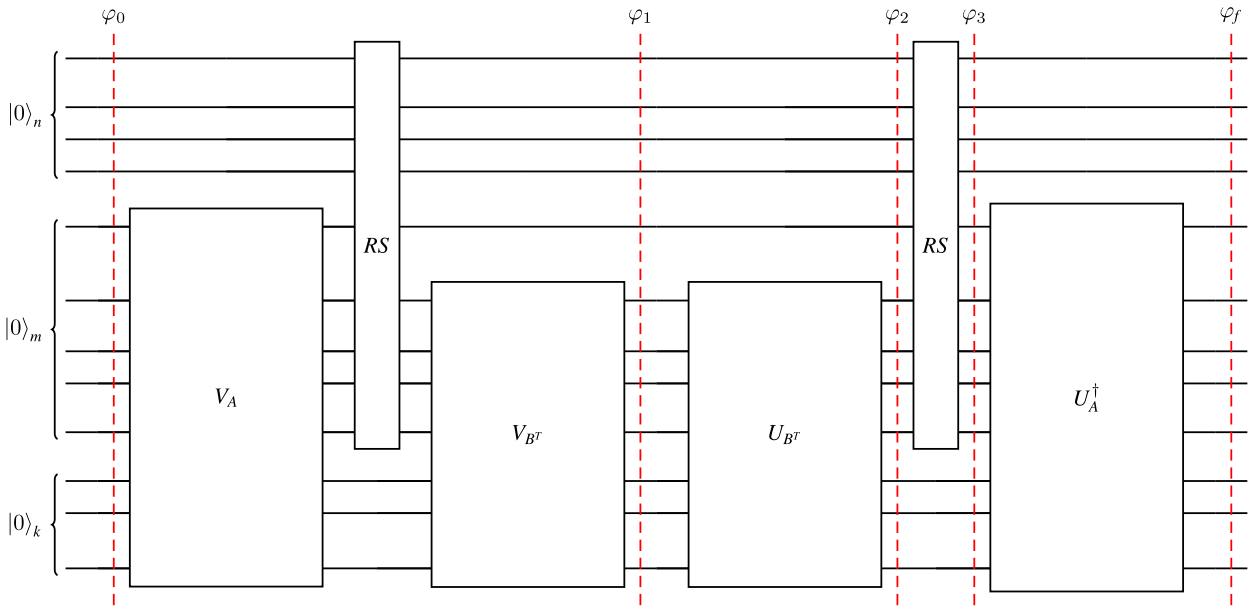

FIGURE 3. QMM: a Quantum Circuit for Matrix Multiplication.

register, leading to the state

$$|\varphi_2\rangle = \frac{1}{\|A\|_F \, \|B\|_F} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \|A(i,:)\|_2 \|B(:,j)\|_2 \, |i\rangle_m \left(\frac{1}{\|B(:,j)\|_2} \, |j\rangle_n \sum_{s=0}^{K-1} B_{sj} \, |s\rangle_k \right)$$

$$= \frac{1}{\|A\|_F \, \|B\|_F} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \sum_{s=0}^{K-1} \|A(i,:)\|_2 B_{sj} \, |i\rangle_m \, |j\rangle_n \, |s\rangle_k \, .$$

Step 4: Swapping back the first two registers with the operator *RS*, we get

$$|\varphi_3\rangle = \frac{1}{\|A\|_F \, \|B\|_F} \sum_{j=0}^{N-1} \sum_{i=0}^{M-1} \sum_{s=0}^{K-1} \|A(i,:)\|_2 B_{sj} \, |j\rangle_n \, |i\rangle_m \, |s\rangle_k \, .$$

Step 5: Finally, we apply the oracle $U_A^\dagger$ to the last two registers and obtain the final quantum state

$$|\varphi_f\rangle = \frac{1}{\|A\|_F \, \|B\|_F} \sum_{j=0}^{N-1} \sum_{i=0}^{M-1} \sum_{s=0}^{K-1} \|A(i,:)\|_2 B_{sj} \, |j\rangle_n \, U_A^\dagger \, |i\rangle_m \, |s\rangle_k .$$

In the following theorem, we prove that this final state $|\varphi_f\rangle$ encodes all the entries of $C = AB$ scaled by the Frobenius norms of A and B.

Theorem 1: Let $A \in \mathbb{R}^{M\times K}$, $B \in \mathbb{R}^{K\times N}$, and let $C \in \mathbb{R}^{M\times N}$ denote their product $C = AB$. Suppose that M, K, N are powers of 2, and that the following oracles are provided:

$$\begin{aligned} U_A\,|i\rangle_m\,|0\rangle_k &= |i\rangle_m\,|A(i,:)\rangle_k\,, \text{ for } 0 \le i < M,\\ U_{B^T}\,|j\rangle_n\,|0\rangle_k &= |j\rangle_n\,|B(:,j)\rangle_k\,, \text{ for } 0 \le j < N,\\ V_A\,|0\rangle_m\,|j\rangle_k &= |\tilde{A}\rangle_m\,|j\rangle_k\,, \text{ for } 0 \le j < K,\\ V_{B^T}\,|0\rangle_n\,|s\rangle_k &= |\tilde{B}^T\rangle_n\,|s\rangle_k\,, \text{ for } 0 \le s < K, \end{aligned}$$

where $n = \log_2 N$, $m = \log_2 M$, and $k = \log_2 K$.

Then, the quantum circuit QMM, starting from the state $|\varphi_0\rangle = |0\rangle_n\,|0\rangle_m\,|0\rangle_k$ and with one use of V_A, V_B, U_B^T and $U_A^\dagger$, and two applications of the Register-Swap operator RS, computes the state

$$|\varphi_f\rangle = \frac{1}{\|A\|_F\|B\|_F}\sum_{j=0}^{N-1}\sum_{i=0}^{M-1}\sum_{s=0}^{K-1}\|A(i,:)\|_2 B_{sj}\,|j\rangle_n\,U_A^\dagger\,|i\rangle_m\,|s\rangle_k$$

that encodes all entries of $C = AB$. In particular, for all $0 \le p < M$ and $0 \le q < N$, it holds that

$$C_{pq} = \|A\|_F\|B\|_F\,{}_n\langle q|\,{}_m\langle p|\,{}_k\langle 0|\varphi_f\rangle.$$

Proof: As detailed before, applying once the four state preparation oracles V_A, V_B, U_B^T, $U_A^\dagger$, and twice the register-swap operator RS according to the quantum circuit QMM, the starting state $|\varphi_0\rangle$ is transformed into the final state $|\varphi_f\rangle$. Now consider the oracle U_A used to encode the rows of A, and observe that

$$\begin{aligned} &{}_m\langle p|_k\langle 0|U_A^\dagger\,|i\rangle_m\,|s\rangle_k\\ &= {}_m\langle p|_k\langle A(p,:)|i\rangle_m\,|s\rangle_k\\ &= {}_m\langle p|i\rangle_m\;{}_k\langle A(p,:)|s\rangle_k\\ &= {}_m\langle p|i\rangle_m\;\frac{1}{\|A(p,:)\|_2}\sum_{j=0}^{K-1}A_{pj}\,{}_k\langle j|s\rangle_k\\ &= {}_m\langle p|i\rangle_m\;\frac{1}{\|A(p,:)\|_2}A_{ps}\\ &= \begin{cases}\frac{1}{\|A(p,:)\|_2}A_{ps}, & i = p\\ 0 & i \ne q\,.\end{cases} \end{aligned}$$

$$\begin{aligned} &\|A\|_F\|B\|_F\,{}_n\langle q|\,{}_m\langle p|\,{}_k\langle 0|\,|\varphi_f\rangle\\ &= {}_n\langle q|\,{}_m\langle p|\,{}_k\langle 0|\sum_{j=0}^{N-1}\sum_{i=0}^{M-1}\sum_{s=0}^{K-1}\|A(i,:)\|_2 B_{sj}\;|j\rangle_n\,U_A^\dagger\,|i\rangle_m\,|s\rangle_k\\ &= \sum_{j=0}^{N-1}\sum_{i=0}^{M-1}\sum_{s=0}^{K-1}\|A(i,:)\|_2 B_{sj}\,{}_n\langle q|\,{}_m\langle p|\,{}_k\langle 0|\,|j\rangle_n\,U_A^\dagger\,|i\rangle_m\,|s\rangle_k\\ &= \sum_{j=0}^{N-1}\sum_{i=0}^{M-1}\sum_{s=0}^{K-1}\|A(i,:)\|_2 B_{sj}\,{}_n\langle q|j\rangle_n\;{}_m\langle p|\,{}_k\langle 0|U_A^\dagger\,|i\rangle_m\,|s\rangle_k\\ &= \sum_{i=0}^{M-1}\sum_{s=0}^{K-1}\|A(i,:)\|_2 B_{sq}\,{}_m\langle p|\,{}_k\langle 0|U_A^\dagger\,|i\rangle_m\,|s\rangle_k\\ &= \sum_{s=0}^{K-1}\|A(p,:)\|_2 B_{sq}\,\frac{1}{\|A(p,:)\|_2}A_{ps} = \sum_{s=0}^{K-1}A_{ps}B_{sq} = C_{pq}\,, \end{aligned}$$

and the thesis follows. ∎

Hence, the state $|\varphi_f\rangle$ encodes all the normalized entries of the product C. In particular, the entry C_{pq}, scaled by $\|A\|_F\|B\|_F$, corresponds to the entry of the vector state $|\varphi_f\rangle$ of index $(qM + p)K$. More precisely, the final vector state corresponds to the vectorization by column of the matrix C (in all entries congruent to 0 modulo K), interlaced with garbage entries.

The characterization of the content of the state vector might be exploited to design a generalization of the proposed matrix product to the product of more than two matrices. Since the memorization of the matrix product in the vector state is by column, the generalized quantum algorithm for computing the product $P = A_1A_2A_3$ should execute the product from right to left, i.e., $P = A_1(A_2A_3)$. However, this generalization is not straightforward, especially if we need to maintain the number of ancillary qubits and the depth of the circuit polylogarithmic in the matrix dimensions.

C. COMPLEXITY ANALYSIS

First of all, observe that the depth and the size of the proposed quantum circuit QMM for matrix multiplication mainly depend on the cost of implementing the oracles U_A, U_{B^T}, V_A, and V_{B^T} for the state preparation, as the register swap operator RS has constant depth and size linear in the number of qubits.

In general, the preparation of a quantum state is a crucial step that can affect the efficiency of quantum algorithms. Specifically, we refer to an *efficient state preparation* when the depth of the oracle implementing the state preparation is polylogarithmic in the input size.

For our application, the cost of the state preparation for encoding the two matrices A and B depends on their structural properties. For example, if the matrices are sparse, data sparse, or present regular patterns [7], it is possible to implement the oracles U_A, U_{B^T}, V_A, and V_{B^T} in polylogarithmic depth in their dimensions.

In the literature, some efficient state preparation techniques for general matrices have been proposed. For instance, we recall here the approach described in [20] that combines a Quantum Random Access Memory [13] with the classical data structure known as KP-trees, to achieve efficient encoding of vectors in the amplitudes. We refer the reader to [9] and [23] for a comprehensive description of this framework. Exploiting QRAM together with KP-trees, the oracles U_A, U_{B^T}, V_A, and V_{B^T} can be implemented with ε-precision with circuits of depth $O(\text{polylog}(MK/\varepsilon))$,

$O(\text{polylog}(NK/\varepsilon))$, $O(\text{polylog}(M/\varepsilon))$, and $O(\text{polylog}(N/\varepsilon))$, respectively (see Theorem 5.1 of [20] and [8]), where ε denotes an upper bound for the norm of the difference between the normalized original matrix and the encoded one. Thus, the overall depth of the QMM circuit becomes of order $O(\text{polylog}(\max\{M,N\}K/\varepsilon))$, providing an exponential speed-up over the best classical fast algorithms.

From a numerical perspective, the computation of the inner products defining the entries of the matrix product is backward stable, meaning that the computed entries are a tiny perturbation of the exact ones [16]. However, forward errors can be high because cancellations may occur while computing the inner products. The overall accuracy of the whole computation also depends on the specific quantum hardware used to physically implement quantum gates.

However, we should mention that quantum hardware that supports data storage and access in superposition in polylogarithmic time is currently still unavailable, and its feasibility is debated [17], [31].

D. A NUMERICAL EXAMPLE

In this section, we detail the steps of our quantum algorithm on a small numerical example. In particular, we will explicitly construct the unitary matrices involved in the computation. Let $A \in \mathbb{R}^{4\times 2}$ and $B \in \mathbb{R}^{2\times 4}$ be defined as

$$A = \begin{bmatrix} -1 & 2 \\ -1 & -5 \\ 3 & -4 \\ 0 & -2 \end{bmatrix}, \quad B = \begin{bmatrix} 2 & -2 & -0 & 3 \\ -4 & -3 & 1 & -1 \end{bmatrix}.$$

Step 1: The initial state $|\varphi_0\rangle$ consists of three registers initialized to 0, of 2, 2, and 1 qubits, respectively:

$$|\varphi_0\rangle = |00\rangle\,|00\rangle\,|0\rangle\,.$$

Thus, $|\varphi_0\rangle \in \mathbb{R}^{32}$, with the first entry equal to 1 and all the others equal to zero.

Step 2: We encode the norms of the rows of A in the second register with 2 qubits applying V_A to $|00\rangle\,|0\rangle$. Since $\|A\|_F = \sqrt{\sum i,jA_{ij}^2} = \sqrt{60}$, we have that $|\tilde{A}\rangle = \frac{1}{\sqrt{60}}[\sqrt{5};\sqrt{26};\sqrt{25},\sqrt{4}]$. A possible form for matrix V_A is[2]

$$V_A = \begin{bmatrix} \sqrt{5/60} & 0.658 & 0.645 & 0.258 \\ \sqrt{26/60} & 0.289 & -0.637 & -0.279 \\ \sqrt{26/60} & -0.258 & 0.667 & -0.267 \\ \sqrt{4/60} & 0.645 & 0.267 & 0.667 \end{bmatrix} \otimes I_2,$$

where $I_2 = I_k$ is the identity matrix of dimension 2, acting on the last register. Similarly a possible form for V_{B^T} is

$$V_{B^T} = \begin{bmatrix} \sqrt{20/44} & -0.544 & -0.389 & 0.314 \\ \sqrt{13/44} & 0.674 & -0.314 & -0.389 \\ \sqrt{1/44} & -0.477 & 0.261 & -0.826 \\ \sqrt{10/44} & 0.151 & 0.826 & 0.261 \end{bmatrix} \otimes I_2.$$

Note that these are not the only possible forms for V_A and V_{B^T} since equation (1) and (2) define the behavior of the operators only on the first K columns, and there are many different ways to complete the matrix to a unitary basis. Indeed, there might be some completion to unitary which are more convenient for the circuital implementation such as the CMV form [2].

At the end of this step, considering also the register swap executed before the application of the operator V_{B^T}, the state becomes

$$|\varphi_1\rangle = \frac{1}{\sqrt{60}\sqrt{44}} \begin{bmatrix} \sqrt{5} \\ \sqrt{26} \\ \sqrt{25} \\ \sqrt{4} \end{bmatrix} \otimes \begin{bmatrix} \sqrt{20} \\ \sqrt{13} \\ \sqrt{1} \\ \sqrt{10} \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

Step 3-5: Matrix U_{B^T} can be constructed as a block diagonal matrix as follows

$$\begin{bmatrix} \frac{2}{\sqrt{20}} & -\frac{4}{\sqrt{20}} & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{4}{\sqrt{20}} & -\frac{2}{\sqrt{20}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{2}{\sqrt{13}} & \frac{3}{\sqrt{13}} & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{3}{\sqrt{13}} & -\frac{2}{\sqrt{13}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{3}{\sqrt{10}} & -\frac{1}{\sqrt{10}} \\ 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{\sqrt{10}} & -\frac{3}{\sqrt{10}} \end{bmatrix}$$

and similarly U_A is

$$\begin{bmatrix} -\frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{5}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{\sqrt{26}} & \frac{5}{\sqrt{26}} & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{5}{\sqrt{26}} & -\frac{1}{\sqrt{26}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{3}{\sqrt{25}} & -\frac{4}{\sqrt{25}} & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{4}{\sqrt{25}} & -\frac{3}{\sqrt{25}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \end{bmatrix}$$

The final state, multiplied by the product of the Frobenius norm of A and B, becomes

$$\begin{aligned} &\|A\|_F\|B\|_F\,|\phi_f\rangle \\ &= [-\mathbf{10},\ 0,\ \mathbf{18},\ 14,\ \mathbf{22},\ 4,\ \mathbf{8},\ -4,\ -\mathbf{4},\ -7,\ \mathbf{17},\ -7,\ \mathbf{6},\\ &\quad 17,\ \mathbf{6},\ 4,\ \mathbf{2},\ 1,\ -\mathbf{5},\ -1,\ -\mathbf{4},\ -3,\ -\mathbf{2},\ 0,\ -\mathbf{5},\ 5,\ \mathbf{2},\ 16,\\ &\quad \mathbf{13},\ -9,\ \mathbf{2},\ -6]. \end{aligned}$$

where, for clarity, almost zero entries have been replaced with zero.

We can easily recognize the entries of C, in boldface, listed by columns and some garbage entries. Indeed

$$C = \begin{bmatrix} -10 & -4 & 2 & -5 \\ 18 & 17 & -5 & 2 \\ 22 & 6 & -4 & 13 \\ 8 & 6 & -2 & 2 \end{bmatrix},$$

and the final state can be rewritten as

$$|\phi_f\rangle = \frac{1}{\sqrt{60}\sqrt{44}} \left(|\phi_C\rangle\,|0\rangle_k + |\phi_G\rangle\,|1\rangle_k\right),$$

[2]We rounded the entries of V_A and V_{B^T} to a three digits precision.

where the unnormalized states $|\phi_C\rangle$ and $|\phi_G\rangle$ are as follows

$$|\phi_C\rangle = [-10,\ 18,\ 22,\ 8,\ -4,\ 17,\ 6,\ 6,\ 2,\ ,-5,\ -4,\ -2,\ -5,\ 2,\ 13,\ 2];$$
$$|\phi_G\rangle = [0,\ 14,\ 4,\ -4,\ -7,\ -7,\ 17,\ 4,\ 1,\ -1-3,\ 0,\ 5,\ 16,\ -9,\ -6].$$

We also report the numerical absolute and relative error measured in the simulation of the algorithm with the IBM Qiskit[3] framework, using a noise-free simulator. Denoting by $\tilde{C}$ the reshaped form of vector $|\phi_C\rangle$, we get

$$\|C - \tilde{C}\|_2 = 1.889\text{e-}15$$

and

$$\frac{\|C - \tilde{C}\|_2}{\|C\|_2} = 9.754\text{e-}14,$$

reaching almost the machine precision (2.2204e-16). Simulating the execution of the circuit on real noisy quantum hardware, using the simulator Fake20QV1 provided by the Qiskit framework, we obtained a density matrix leading to a very low fidelity of about 0.24 in an all-to-all qubit connection. This shows that our algorithm can be run on real quantum machines, but the noise makes the result unreliable.

IV. DISCUSSION

This section compares our algorithm with the block-encoding approach proposed in [12]. First of all, recall that the *block encoding* technique consists of embedding a scaled, non-unitary $N \times N$ square matrix into the top left block of a larger unitary one that can then be used within a quantum circuit. Once matrices have been block-encoded, we can operate on their corresponding unitary operators so that all quantum matrix calculations are carried out in an operational way. In particular, the computation of the matrix product $C = AB$ can be easily carried out starting from the block-encoding of A and B.

The cost of implementing the block encoding of dense unstructured matrices has been analyzed in [9], where a careful evaluation of the performance of the encoding in terms of T-gate counts and circuits' depth has been provided.

However, we can observe that the block-encoding approach does not immediately provide a final state vector encoding all entries of C, as all entries reside in the top-left block of the unitary describing the circuit. Of course, by exploiting particular configurations of the initial state, it is possible to retrieve any selected column of the product. Otherwise, by adding some ancillary qubits, we can exploit the quantum parallelism and retrieve all columns of the matrix product C in the final state.

The construction of the block-encoding requires efficient state preparation techniques similar to the ones briefly discussed in Section III-C. Therefore, this technique is similar to our proposal, but it requires the additional state preparation of an appropriate sparse initial state in order to retrieve all the columns of the matrix product.

A different approach, again based on the block-encoding technique, requires to *(i)* block-encode the first matrix A into a unitary $\mathcal{U}$, *(ii)* add $\log N$ ancillary qubits to get the operator $I_N \otimes \mathcal{U}$, and *(iii)* prepare a state encoding the vectorized by column version of B. This technique exploits the possibility offered by the block encoding of computing matrix-vector products easily.

Despite its similarity to the more general block-encoding methodology, our quantum algorithm for matrix multiplication offers some advantages. First of all, it allows the manipulation of even rectangular matrices without padding with zeros to get square matrices, causing a potential increase in the number of qubits. Moreover, it provides by construction an encoding of all entries of the product directly in the final state vector without requiring additional manipulations, or a measurement. The cost, in terms of width, size, and depth of the circuits, appears to be equivalent. Still, our approach guarantees that each entry is already individually encoded and indexed within the state vector.

V. APPLICATION: VARIANCE OF THE PRODUCT OF TWO MATRICES

As already observed, the quantum algorithm QMM for matrix multiplication can be exploited in all scenarios involving the computation of non-homomorphic functions of the product of two matrices, for instance, trace, mean value, variance, or eigenvalues. In this regard, we now show how QMM can be exploited to compute the variance of the entries of a matrix product.

The variance is a pervasive measure that finds application across a vast number of domains [18], including the field of Artificial Intelligence. Some applications within this area require the computation of the variance of the entries of a matrix product. For instance, the classical Angle-Based Outlier Detection (ABOD) algorithm [21] for the Outlier Detection problem, exploits the variance of $A^\dagger A$, where A is the matrix encoding the dataset.

Let $A \in \mathbb{R}^{M\times K}$ and $B \in \mathbb{R}^{K\times N}$ be two matrices, let $C \in \mathbb{R}^{M\times N}$ be their product, and let $Var(C)$ denote the variance of the elements of C. First of all, observe that $Var(C)$ cannot be derived from the variances of A and B, but requires the computation of all entries of C, i.e., the variance is a non-homomorphic function.

We can implement a quantum circuit for computing $Var(C)$ combining the circuit QMM with the quantum subroutine QVAR proposed in [1] and [27] to compute the variance of a quantum superposition of values indexed by proper register qubits.

The overall circuit, depicted in Figure 4, requires five registers and an ancilla qubit. The three last registers $|\hat{n}\rangle$, $|\hat{m}\rangle$, and $|\hat{k}\rangle$ are the input of the QMM circuit and are registers composed of $n = \log_2 N$, $m = \log_2 M$, and $k = \log_2 K$

[3] https://qiskit.org/

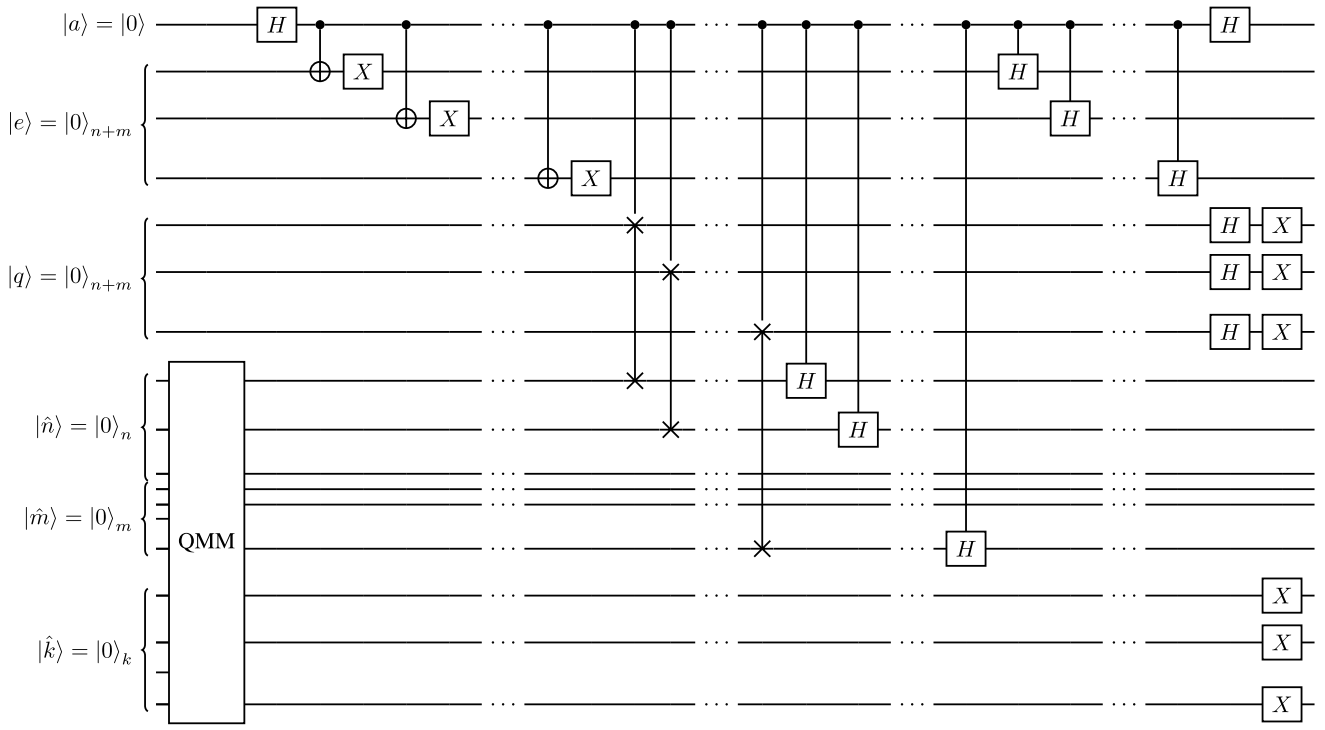

FIGURE 4. QVAR oracle for the computation of the variance.

qubits, respectively. These registers are used to amplitude encode the entries of the matrices A and B and to compute their product in superposition. The ancilla qubit $|a\rangle$ and the other two registers $|e\rangle$ and $|q\rangle$, each with $(n+m)$ qubits, are used to compute the variance of the amplitudes of the state vector $|\varphi_f\rangle$ computed by QMM.

We refer the reader to [1] for a detailed description of the quantum variance computation. Here, we only recall that the main idea of the QVAR subroutine is to use the ancilla qubit $|a\rangle$ to create an equal superposition of two branches through a Hadamard gate. In the branch where the ancilla is in state $|0\rangle$, the circuit maintains the superposition of the amplitudes of the state $|\varphi_f\rangle$ computed by QMM, while in the branch where the ancilla is in state $|1\rangle$, the circuit computes their mean value. Eventually, another Hadamard gate on $|a\rangle$ is applied to make the two branches collide and create the superposition containing the component of the variance, namely the differences between each amplitude and the mean value.

As explained in [1], all these differences are the amplitudes of the final state vector related to specific configurations of the qubits. Moreover, to compute $Var(C)$ we must select from $|\varphi_f\rangle$ only the entries of the matrix C, encoded in the amplitudes of the configurations where $|\hat{k}\rangle = |0^{\otimes k}\rangle$, as proved in Theorem 1. Therefore, the Amplitude Estimation algorithm [4] is used to estimate the variance value as the sum of squares of these target amplitudes only. In particular, for the computation of $Var(C)$, Amplitude Estimation is employed measuring s additional qubits to estimate the amplitude of the target configuration where $|a\rangle|e\rangle|q\rangle = |1\rangle|1^{\otimes(n+m)}\rangle|0^{\otimes(n+m)}\rangle$ and $|\hat{k}\rangle = |0^{\otimes k}\rangle$. This measurement output represents an approximation of the variance of C in the computational basis.

The overall circuit for computing $Var(C)$ requires in total $3(m+n)+k+s+1$ qubits, where $s = O(\log \frac{1}{\varepsilon})$ is the number of additional qubits required by Amplitude Estimation to get an estimate of the variance with (absolute) error ε with respect to the classical variance.

Concerning the complexity of this method, we first observe that the QVAR circuit exhibits a logarithmic complexity both in the circuit depth and width, excluding the state preparation cost, as discussed in [1]. Moreover, for the specific computation of $Var(C)$, no additional state preparation is required beyond the encoding of A and B in QMM. Indeed, the entries of C are already encoded in the state on which the QVAR oracle acts. The complexity of the overall circuit is, therefore, mainly due to the complexity of QMM.

Eventually, since the cost of Amplitude Estimation is $O(\delta\frac{1}{\varepsilon} + \log\log\frac{1}{\varepsilon})$ [14] where δ is the depth of the circuit in Figure 4, the overall complexity for the computation of $Var(C)$ becomes $O(\frac{1}{\varepsilon}\text{polylog}(\max\{M,N\}K/\varepsilon))$, assuming efficient state preparation techniques for matrix encoding.

VI. CONCLUSION

In this work, we designed and analyzed a quantum algorithm for matrix multiplication that computes a state

vector encoding the entries of the product in superposition, in time polylogarithmic in the matrix dimensions, assuming efficient state preparation techniques. This algorithm enables immediate composition with other quantum circuits, implementing computations that involve possibly all entries of the product of the two matrices. As a possible application, we discussed the computation of the variance of the entries of a matrix product, a useful tool for some machine learning algorithms.

As future work, we plan to generalize the algorithm to the multiplication of more than two matrices, according to the intuition mentioned at the end of Section III-B. Moreover, we plan to extend it to deal with matrices over $\mathbb{C}$. The easy approach would consist of a repetition of the algorithm for real and imaginary parts of the two matrices. A more efficient approach would require the definition of state preparation techniques capable of dealing with complex entries. We also plan to identify other applications that could benefit from this approach.

ACKNOWLEDGMENT

This study was carried out within the National Centre on HPC, Big Data and Quantum Computing-SPOKE 10 (Quantum Computing).

REFERENCES

[1] A. Bernasconi, A. Berti, G. M. D. Corso, R. Guidotti, and A. Poggiali, "Quantum subroutine for variance estimation: Algorithmic design and applications," 2024, *arXiv:2403.14655*.

[2] R. Bevilacqua, G. M. Del Corso, and L. Gemignani, "Compression of unitary rank-structured matrices to CMV-like shape with an application to polynomial rootfinding," *J. Comput. Appl. Math.*, vol. 278, pp. 326–335, Apr. 2015.

[3] D. Bini, M. Capovani, F. Romani, and G. Lotti, "$o(n^{2.7799})$ complexity for n × n approximate matrix multiplication," *Inf. Process. Lett.*, vol. 8, no. 5, pp. 234–235, 1979. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0020019079901133

[4] G. Brassard, P. Hoyer, M. Mosca, and A. Tapp, "Quantum amplitude amplification and estimation," *Contemp. Math.*, vol. 305, pp. 53–74, Oct. 2002.

[5] H. Buhrman, R. Cleve, J. Watrous, and R. de Wolf, "Quantum fingerprinting," *Phys. Rev. Lett.*, vol. 87, no. 16, Sep. 2001, Art. no. 167902, doi: 10.1103/physrevlett.87.167902.

[6] P. Bürgisser, M. Clausen, and M. A. Shokrollahi, *Algebraic Complexity Theory*, 1st ed., Cham, Switzerland: Springer, 1996.

[7] D. Camps, L. Lin, R. Van Beeumen, and C. Yang, "Explicit quantum circuits for block encodings of certain sparse matrices," *SIAM J. Matrix Anal. Appl.*, vol. 45, no. 1, pp. 801–827, Mar. 2024.

[8] S. Chakraborty, A. Gilyén, and S. Jeffery, "The power of block-encoded matrix powers: Improved regression techniques via faster Hamiltonian simulation," in *Proc. 46th Int. Colloquium Automata, Lang., Program. (ICALP)*, vol. 132, C. Baier, I. Chatzigiannakis, P. Flocchini, and S. Leonardi, Eds., Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum Für Informatik, 2019, pp. 33:1–33:14, doi: 10.4230/LIPIcs.ICALP.2019.33.

[9] B. D. Clader, A. M. Dalzell, N. Stamatopoulos, G. Salton, M. Berta, and W. J. Zeng, "Quantum resources required to block-encode a matrix of classical data," *IEEE Trans. Quantum Eng.*, vol. 3, pp. 1–23, 2022, doi: 10.1109/TQE.2022.3231194.

[10] D. Coppersmith and S. Winograd, "Matrix multiplication via arithmetic progressions," *J. Symbolic Comput.*, vol. 9, no. 3, pp. 251–280, Mar. 1990, doi: 10.1016/s0747-7171(08)80013-2.

[11] C. A. Furia, "Rotation of sequences: Algorithms and proofs," 2014, *arXiv:1406.5453*.

[12] A. Gilyén, Y. Su, G. H. Low, and N. Wiebe, "Quantum singular value transformation and beyond: Exponential improvements for quantum matrix arithmetics," in *Proc. 51st Annu. ACM SIGACT Symp. Theory Comput.*, Jun. 2019, pp. 193–204.

[13] V. Giovannetti, S. Lloyd, and L. Maccone, "Quantum random access memory," *Phys. Rev. Lett.*, vol. 100, no. 16, Apr. 2008, Art. no. 160501.

[14] T. Giurgica-Tiron, I. Kerenidis, F. Labib, A. Prakash, and W. Zeng, "Low depth algorithms for quantum amplitude estimation," *Quantum*, vol. 6, p. 745, Jun. 2022.

[15] A. W. Harrow, A. Hassidim, and S. Lloyd, "Quantum algorithm for linear systems of equations," *Phys. Rev. Lett.*, vol. 103, no. 15, Oct. 2009, Art. no. 150502, doi: 10.1103/physrevlett.103.150502.

[16] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed., Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2002, doi: 10.1137/1.9780898718027.

[17] S. Jaques and A. G. Rattew, "Qram: A survey and critique," 2023, *arXiv:2305.10310*.

[18] A. Jovic, K. Brkic, and N. Bogunovic, "A review of feature selection methods with applications," in *Proc. 38th Int. Conv. Inf. Commun. Technol., Electron. Microelectron. (MIPRO)*, May 2015, pp. 1200–1205.

[19] E. Karstadt and O. Schwartz, "Matrix multiplication, a little faster," *J. ACM*, vol. 67, no. 1, pp. 1–31, Jan. 2020, doi: 10.1145/3364504.

[20] I. Kerenidis and A. Prakash, "Quantum recommendation systems," in *Proc. 8th Innov. Theor. Comput. Sci. Conf. (ITCS)*, vol. 67, C. H. Papadimitriou, Ed., Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum Für Informatik, 2017, pp. 49:1–49:21, doi: 10.4230/LIPIcs.ITCS.2017.49.

[21] H.-P. Kriegel, M. Schubert, and A. Zimek, "Angle-based outlier detection in high-dimensional data," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2008, pp. 444–452.

[22] H. Li, N. Jiang, Z. Wang, J. Wang, and R. Zhou, "Quantum matrix multiplier," *Int. J. Theor. Phys.*, vol. 60, no. 6, pp. 2037–2048, Jun. 2021.

[23] A. Luongo. *Classical Data in Quantum Computers*. Accessed: Feb. 13, 2024. [Online]. Available: https://quantumalgorithms.org/chap-classical-data-quantum-computers.html

[24] V. Markov, C. Stefanski, A. Rao, and C. Gonciulea, "A generalized quantum inner product and applications to financial engineering," 2022, *arXiv:2201.09845*.

[25] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information (10th Anniversary Edition)*. Cambridge, U.K.: Cambridge Univ. Press, 2016.

[26] J. Pan, Y. Cao, X. Yao, Z. Li, C. Ju, H. Chen, X. Peng, S. Kais, and J. Du, "Experimental realization of quantum algorithm for solving linear systems of equations," *Phys. Rev. A, Gen. Phys.*, vol. 89, no. 2, Feb. 2014, Art. no. 022313.

[27] A. Poggiali, A. Bernasconi, A. Berti, G. M. Del Corso, and R. Guidotti, "Quantum feature selection with variance estimation," in *Proc. Eur. Symp. Artif. Neural Netw., Comput. Intell. Mach. Learn. (ESANN)*, Bruges, Belgium, 2023, pp. 245–250, doi: 10.14428/esann/2023.ES2023-99.

[28] C. Shao, "Quantum algorithms to matrix multiplication," 2018, *arXiv:1803.01601*.

[29] V. Strassen, "Gaussian elimination is not optimal," *Numerische Math.*, vol. 13, no. 4, pp. 354–356, Aug. 1969. [Online]. Available: http://eudml.org/doc/131927

[30] C. Sünderhauf, E. Campbell, and J. Camps, "Block-encoding structured matrices for data input in quantum computing," *Quantum*, vol. 8, p. 1226, Jan. 2024, doi: 10.22331/q-2024-01-11-1226.

[31] E. Tang, "Quantum machine learning without any quantum," Ph.D. dissertation, Dept. Comput. Sci. Eng., Univ. Washington, Seattle, WA, USA, 2023.

[32] L.-C. Wan, C.-H. Yu, S.-J. Pan, S.-J. Qin, F. Gao, and Q.-Y. Wen, "Block-encoding-based quantum algorithm for linear systems with displacement structures," *Phys. Rev. A, Gen. Phys.*, vol. 104, no. 6, Dec. 2021, Art. no. 062414, doi: 10.1103/physreva.104.062414.

[33] N. Wiebe, D. Braun, and S. Lloyd, "Quantum algorithm for data fitting," *Phys. Rev. Lett.*, vol. 109, no. 5, Aug. 2012, Art. no. 050505.

[34] V. V. Williams, "Multiplying matrices faster than coppersmith-Winograd," in *Proc. 44th Annu. ACM Symp. Theory Comput.* New York, NY, USA: Association for Computing Machinery, May 2012, pp. 887–898, doi: 10.1145/2213977.2214056.

[35] V. V. Williams, Y. Xu, Z. Xu, and R. Zhou, "New bounds for matrix multiplication: From Alpha to Omega," in *Proc. Annu. ACM-SIAM Symp. Discrete Algorithms (SODA)*, 2024, pp. 3792–3835.

ANNA BERNASCONI received the Laurea degree in physics from the University of Pavia in 1992, and the Ph.D. in computer science from the University of Pisa in 1998. She is currently an Associate Professor with the Department of Computer Science, University of Pisa, where she teaches fundamental courses in the computer science program (algorithms and data structures, cryptography, and quantum computing). She has authored or co-authored more than 90 research papers published in international journals, conference proceedings, books, and book chapters. Her research interests include algorithms and complexity, Boolean function complexity, and logic synthesis for emerging technologies. Recently, she got interested in quantum computing, and in particular in quantum machine learning techniques and in the synthesis of quantum logic circuits. Her dissertation ''Mathematical Techniques for the Analysis of Boolean Functions'' received the Doctoral Dissertation Thesis Award 1998 from Italian Chapter of the European Association for Theoretical Computer Science (EATCS).

ALESSANDRO BERTI received the bachelor's, master's, and Ph.D. degrees in computer science from the University of Pisa, Italy, in 2017, 2020, and 2024, respectively. He was a Visiting Ph.D. Student with the Superconducting Quantum Materials and Systems (SQMS) Center, Fermilab, Batavia, IL, USA, from 2022 to 2023. He is currently a Postdoctoral Researcher with the Department of Physics, University of Pisa. His research interests include quantum computing applications in artificial intelligence, variational circuits, and efficient quantum state preparation techniques. In addition, he is an Advocate of scientific dissemination and has contributed to various education and outreach activities.

GIANNA MARIA DEL CORSO received the Ph.D. degree from the University of Milano. She has been a Visiting Scholar with the Computer Science Department, Columbia University, New York, and a Visiting Researcher with the Department of Computer Science, Johns Hopkins University, Baltimore. She joined the Computer Science Department, University of Pisa, in June 2000, where she is currently an Associate Professor of numerical analysis. Her main scientific interests range from the study of algorithms for eigenvalue computation of low-rank perturbation of unitary matrices to the use of spectral techniques for data classification. In the quantum computing field, she is especially fascinated by the study of state preparation, quantum machine learning techniques, and discrete-time quantum walks.

ALESSANDRO POGGIALI (Graduate Student Member, IEEE) received the B.S. degree in computer science from the University of Pisa, in 2019, with a thesis on Boolean functions, and the M.S. degree in computer science from the University of Pisa, in April 2022, with a thesis on quantum algorithms, where he is currently pursuing the Ph.D. degree in artificial intelligence. From April to November 2022, he was a Research Assistant with the University of Piemonte Orientale. From 2023 to 2024, he spent seven months at the Fraunhofer Institute for Cognitive Systems, Munich, Germany, as a part of the Quantum-Enhanced AI Team. His current research interest includes hybrid quantum algorithms for artificial intelligence.

Open Access funding provided by 'Università di Pisa' within the CRUI CARE Agreement